Using Connections to Make Predictions on Dynamic Networks

Rebecca Dorff Jones
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Physical Sciences and Mathematics Commons

BYU ScholarsArchive Citation
Jones, Rebecca Dorff, "Using Connections to Make Predictions on Dynamic Networks" (2022). Theses and Dissertations. 9388.
https://scholarsarchive.byu.edu/etd/9388

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
Using Connections to Make Predictions on Dynamic Networks

Rebecca Dorff Jones

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Emily Evans, Chair
Amanda Francis
Tyler Jarvis
Mark Kempton
Benjamin Webb

Department of Mathematics
Brigham Young University

Copyright © 2022 Rebecca Dorff Jones
All Rights Reserved
ABSTRACT

Using Connections to Make Predictions on Dynamic Networks

Rebecca Dorff Jones
Department of Mathematics, BYU
Doctor of Philosophy

Networks are sets of objects that are connected in some way and appear abundantly in nature, sociology, and technology. For many centuries, network theory focused on static networks, which are networks that do not change. However, since all networks transform over time, static networks have limited applications. By comparison, dynamic networks model how connections between objects change over time. In this work, we will explore how connections in dynamic networks change and how we can leverage these changes to make predictions about future iterations of networks. We will do this by first considering the link prediction problem, using either Katz distance or effective resistance to predict future connections, and relate these two metrics. Then we will look at using bipartite network connections to predict group transitions in professional sports teams. Lastly, we will investigate how to use network connections to identify and predict roles in social networks.

Keywords: network theory, machine learning, Katz distance, link prediction, role identification, dynamic networks
ACKNOWLEDGEMENTS

This work would not have been possible without the support and unfailing unconditional love of my husband Preston, the support of my parents Michael and Sarah, and the flexibility and understanding of my adviser Emily.

Chapter 6 was supported by the U.S. Department of Energy’s NNSA; and PNNL, operated by Battelle for the U.S. Department of Energy under Contract DE-AC05–76RL01830.
CONTENTS

Contents iv

List of Tables vii

List of Figures ix

1 Introduction 1

2 Network Theory 3

2.1 Introduction 3

2.2 Types of Networks 5

2.3 Terminology and Definitions 6

2.4 Special Graphs 9

2.5 Matrix Representations 13

2.6 Metrics 16

2.7 Similarity 23

2.8 Network Theory Problems 25

3 Machine Learning 27

3.1 Machine Learning Concepts 27

3.2 The Machine Learning Process 35

3.3 Regression 38

3.4 Classification 41

3.5 Clustering 46

3.6 Neural Networks 47

3.7 Time Series 49

4 Katz 52
### List of Tables

3.1 Classification Result Matrix. ........................................... 32
3.2 Machine Learning Metrics. ............................................. 45
3.3 Input matrix for convolution. ......................................... 48
3.4 Kernel for convolution. ................................................ 48
3.5 Output matrix of convolution. ........................................ 48

5.1 Sample of MLB data. ................................................... 92
5.2 Number of MLB players per year. .................................... 92
5.3 Sample NBA data. ........................................................ 93
5.4 Number of NBA players per year. .................................... 95
5.5 Three years of collected team data for a representative basketball player. .................................................. 98
5.6 Final MLB and NBA dataset totals. ................................. 100
5.7 Basketball Predictions Results. ...................................... 102
5.8 MLB Results. ............................................................. 103
5.9 Basketball Inference and Prediction Results. .................... 104
5.10 Baseball Inference Results. .......................................... 106
5.11 Basketball Inference and Prediction Accuracy. .................. 107
5.12 Baseball Inference Accuracy. ........................................ 108
5.13 Basic statistics for the MLB and NBA Twitter networks. .... 108
7.1 Results of Johanson cointegration test on House of Representative Roles. .............................................. 127
7.2 Accuracy of Congress Role Detection using KMeans. ............ 132
A.1 Summary of Algorithm Accuracy for Baseball data from 2001-2019. ................................................. 136
A.2 Summary of Algorithm F1 for Baseball data for 2002-2019. ...................................................... 137
A.3 Summary of confidence intervals for Baseball data for 2002-2019. .................................................. 137
A.4 Summary of Algorithm Accuracy for Baseball data from 2020. ...................................................... 138
## List of Figures

2.1 Diagram of the seven bridges of Königsberg. .................................. 3
2.2 $K_5$ graph with 5 nodes each connected to every other node. ............... 5
2.3 Examples of undirected and directed graphs. .................................. 7
2.4 A directed multigraph with self loops and edge weights. ....................... 8
2.5 Examples of trees .................................. 9
2.6 A fork with 7 nodes that splits on node 5. .................................... 10
2.7 $K_8$. .................................. 10
2.8 A regular graph of degree 3. .................................. 10
2.9 A cycle graph with 8 nodes. .................................. 11
2.10 Bipartite graph example. .................................. 12
2.11 Random graph. .................................. 13
2.12 Graph with 0 eigenvector centrality. .................................. 18
2.13 Examples of equivalence classes of nodes. .................................. 23

3.1 Sample confusion matrix .................................. 34

4.1 Plots of $R(j, l, \alpha)$ .................................. 61
4.2 A path graph with 6 nodes. .................................. 62
4.3 A fork with 7 nodes that splits on node 5. .................................... 69
4.4 A cycle graph with 8 nodes. .................................. 75

5.1 Percent of MLB/NBA players with public, active Twitter accounts. ........ 94
5.2 Histograms of MLB degree centrality and NBA college alumni. ............ 96
5.3 MLB/NBA affinity network examples. .................................. 97
5.4 NBA alma mater histograms. .................................. 105
5.5 Degree centrality histograms for MLB and NBA. ................................ 110
5.6 MLB and NBA centrality histograms. .................................. 110
6.1 RMSE for different ranks of NMF. ........................................ 118
6.2 The six Twitter roles described by the features used. .............. 120
6.3 Percentage of each role over time. .................................... 121
6.4 Probability of transitioning from a role (row) to a role (column). . 122

7.1 The 4 roles identified for the House of Representatives. ............. 125
7.2 The 4 roles identified for the Senate. .................................. 125
7.3 The distributions of the 4 roles identified for the House of Representatives. 126
7.4 The distributions of the 4 roles identified for the Senate. ............ 126
7.5 Prediction of the distribution of the four roles in the House of Representatives.
   The top graph represents the Typical roles, and the second represents State
   Loyalist. The third graph represents the Collaborator. The bottom graph
   represents Party Loyalist. .................................................. 128
7.6 Three time periods of a dynamic network. .............................. 130
7.7 The 5th House of Representatives network ............................... 132
Chapter 1. Introduction

Networks have been studied for hundreds of years in many different fields, including mathematics, computation, information technology, biology, anatomy, sociology, neurology, epidemiology, technology, and transportation. With so many applications, they offer a wide variety of interesting problems to analyze and solve, especially with the rise in machine learning and computational capability. These advances have allowed researchers to study larger and more complex networks including dynamic networks.

One dynamic network problem of particular interest in online social networks and online shopping is the prediction of new connections. Called link prediction, it asks if given a dynamic network, what nodes are going to be connected in the next time period? In Facebook or Twitter for example, it is one way of choosing who to recommend as a potential friend or someone to follow. There are several different metrics used to make these predictions. One is called effective resistance, or resistance distance, and it is based off of electrical circuits. Another one, Katz distance, is based off of the idea that who an object is immediately connected to, its neighbors, is important and will affect the prediction. Katz distance is difficult to study on networks, so very little work has been done on understanding its consequences. Being different metrics, effective resistance and Katz distance can yield different predictions. Effective resistance is only dependent on the network connections, while Katz distance requires a weighting parameter $\alpha$. A natural question then is: by tweaking $\alpha$, is it possible to guarantee that Katz distance and effective distance predict the same new connections? Understanding this boundary can help determine which metric to use in certain cases and in what scenarios they are the same.

A similar problem to link prediction is to use the current connections to predict group transitions. This related idea takes a bipartite network, which has two disjoint sets of objects such that objects in one set are only connected to objects in the other set. Instead of predicting the new connections, we aim to predict which group objects will be in the
following time period. Using professional sports data, we can create a bipartite network of players and teams. There is a connection between a player and a team if the player is friends with someone on that team. Using these connections, we can then try to predict which team a player will be on in the next season. Just using the connections from the previous time period, like in link prediction, is not going to be very effective, so we add more data using machine learning techniques to increase the accuracy of this group transition problem.

In addition to classifying network nodes into traditional communities, we can also group them by role. Roles are often not explicitly defined, even if they seem obvious based on domain knowledge. For example in a social network, there may be influencers whose goal is to reach as many followers as possible, or in a pandemic network there may be super spreaders, who infect a large number of people. We understand these roles from experience but finding them based off of network structure is difficult. Predicting when an object’s role will change is a modification of the group transition problem above in which the groups are hidden, and we want to identify which nodes are in which groups and predict how the groups change. This work will build off some recent research and explore possible predictive solutions in dynamic social networks.

We proceed with the following format: Chapter 2 will discuss the history of networks as well as important terminology, definitions, metrics and representations. In Chapter 3, we will introduce machine learning, focusing on the machine learning process and algorithms that will be used in this work. Chapter 4 will look at link prediction and prove some claims about Katz distance on two simple networks: paths, forks, and cycles. Chapter 5 will investigate how connections in professional sports affect team transitions. In Chapter 6, we will discuss a method of identifying roles in dynamic cyber social networks, and Chapter 7 will expand on those techniques to make role predictions in social networks. Chapter 8 will conclude the work.
Chapter 2. Network Theory

2.1 Introduction

The study of networks formally began in 1736, when Leonhard Euler solved the Seven Bridges of Königsberg problem [1]. At that time, the Pregel River ran through the city of Königsberg, now Kaliningrad, Russia, splitting it and encircling two islands. Connecting the islands and the main city were seven bridges. Figure 2.1 shows the landmasses and bridges in Königsberg.

Figure 2.1: Diagram of the seven bridges of Königsberg. ¹

The problem asked how to walk through the city crossing each of the seven bridges once and only once. Euler proved that there is no solution. He observed that in order to cross all of the bridges only once, only two of the land masses can have an odd number of bridges and these must be the starting and ending points. However, in this situation, all of the land masses have an odd number of bridges; one island has five bridges, while the other island and mainlands have three. Thus no matter where a person started, they would get stuck

¹Image by Chris Martin, distributed under a GNU Free Documentation License https://commons.wikimedia.org/wiki/File:7_bridges.svg.
after crossing the last bridge onto a landmass. This idea of crossing every section in a path once and only once is named an Eulerian path after Euler [2].

Network theory kept developing in the 1800s with a few contributions of note. Thomas Kirkman and William Hamilton defined a Hamiltonian network which is similar to a Eulerian network. Instead of crossing each segment in the path exactly once, every vertex or point is visited exactly once [2]. Another important definition, a tree, was developed by Cayley, Sylvester, and Polya while they enumerated chemical isomers [3]. Many of the original network theory problems were posed as puzzles, including the question by Euler above and one presented by A. F. Mobius around 1840 in one of his geometry lectures at the University of Leipzig [4]:

Once upon a time, there was a king with five sons. In his will, he stated that after his death the sons should divide the kingdom into five provinces so that the boundary of each province should have a frontiers line in common with each of the other four provinces. The king further stated that all five brothers should join the provincial capital by roads so that no two roads intersect.

This can be thought of as five different points (the provincial capitals) that are all connected to each other, known as the $K_5$ network, shown in Figure 2.2. The problem asks if the $K_5$ network is planar, whether each edge can be drawn without intersecting. Determining if a network is planar or not has been a highly researched question in graph theory.

Another puzzle arose a few years later in 1852, Frederick Guthrie, on behalf of his brother Francis Guthrie, asked Augustus De Morgan, his professor of mathematics at University College in London why the following question was true [4]:

If a figure be any how divided and the compartments differently coloured so that figures with any portion of common boundary lines are differently coloured—four colours may be wanted, but not more.

This wasn’t proved until 1976 by Kenneth Appel and Wolfgang Haken who used 1200 hours of computational time on a computer [5]. With the development of computational re-
sources, network theory has expanded into many different areas that require large processing power, including genome sequencing, community detection, neurology, and link predictions on social networks. The rest of this chapter consists of definitions and terminology important to understanding network theory, and unless otherwise referenced, can be found in M. E. J Newman’s seminal work “Networks: An Introduction” [2] as well as Estrada’s and Knight’s “A First Course in Network Theory” [3].

2.2 TYPES OF NETWORKS

Networks are found everywhere, including biology, sociology, neuroscience, and public health. Social networks are perhaps the most recognizable networks. They comprise people who are connected in some way. Families and social groups are natural networks that arise from our daily interactions. Social media such as Facebook, Twitter, and Instagram are also social networks where people are connected by followers and friends. Sometimes considered its own category of network, disease spread can also be considered a social network.

Biological networks are very common as well. The neurons in our bodies form a neurological network that helps us function. Metabolic networks, the way food is broken down into energy, have been gaining popularity as genetic technology has improved. Other biological
networks are the protein-protein interaction networks, which describe how proteins interact in different species. Species of animals are related through predator-prey and food-web networks, as well as mutualistic networks, networks of mutually beneficial interactions.

Technological networks play a large role in our modern daily lives. These networks, which include transportation networks like roads and planes, electrical networks, internet networks, and delivery and supply chain networks, determine how long our commute is, how constant the electricity in our home is, and how long it takes for us to receive purchases. The circuits that run electronics are also networks.

Citation and coauthor networks are types of information networks. Citation networks connect journal articles or books that cite each other, and coauthor networks connect authors who publish together. The most famous information network is the World Wide Web where web pages are connected through links. Search engines like Google and Bing use network properties based on these links to determine what results to display when you search them.

2.3 Terminology and Definitions

Definition 2.1. Let $V$ be a finite set of elements, called nodes or vertices. Define

$$E \subset \{(x, y) \in V \times V \mid x \neq y\},$$

to be the set of edges, each of which connects two elements in $V$. Then the pair $G = (V, E)$ is called a graph or network.

While the above definition holds for both graphs and networks, in mathematics, networks are typically represented by graphs so graph is the term most often used. In other fields, the term network is more common. However, they are interchangeable.

Definition 2.2. If $E$ is not ordered, $G$ is undirected. If $E$ is a set of ordered pairs, then $G$ is called a directed graph, or a digraph for short.
Figure 2.3a is an example of an undirected graph in which edges go both directions. Figure 2.3b is the same graph as Figure 2.3a except that it is directed. The edges go in the direction the arrows point. Note that in Figure 2.3b there are two edges between nodes 0 and 6, $(0, 6)$ and $(6, 0)$.

![Figure 2.3: Examples of undirected and directed graphs.](image)

**Definition 2.3.** If $E$ is allowed to have duplicates, $G$ is called a *multigraph*. Figure 2.4 has multiple edges between nodes 1 and 4 and 2 and 4. In a directed graph, $(x, y) \neq (y, x)$ so they are not considered duplicates.

**Definition 2.4.** In the above definition, an edge cannot connect a single vertex to itself. $G$ can be extended to include loops, where

$$E \subset \{x, y \in V\}.$$  

$G$ is then said to be a *graph (or multigraph) with loops*. There is a loop at node 2 in Figure 2.4.

**Definition 2.5.** A *weighted graph* $G$ is a graph in which each edge is associated with a weight, given by a weight function $w : E \rightarrow \mathbb{R}$. The graph in Figure 2.4 has edge weights on some of the edges.

As an example of these terms, consider the road system in a city. It is typically an undirected graph; cars can travel on roads in both directions. However, a downtown area
which is comprised of only one way streets would be a directed network. Usually there aren’t two different roads connecting two intersections, an example of multiple edges, but an exception is where a frontage road connects adjoining highway on-ramps. A weighted graph of a road system could use the length of each road segment as the weight.

**Definition 2.6.** A node $x$ is a *neighbor* to node $y \in V$ if there is an edge $(x, y)$ or $(y, x) \in E$. The neighbors of $x$ is the set of nodes that share an edge with $x$,

$$N_x = \{y \in V \mid (x, y) \text{ or } (y, x) \in E\}.$$

**Definition 2.7.** The *degree* of a node is the number of its edges, or neighbors in an graph without multiple edges,

$$\deg(u) = |\{(u, v), (v, u) \in E\}|.$$

For a directed graph, the *in-degree* is the number of incoming edges, $|\{(v, u) \in E\}|$ while the *out-degree* is the number of outbound edges $|\{(u, v) \in E\}|$

**Definition 2.8.** A *subgraph* of a graph $G$ is a graph formed from a subset of nodes and edges of $G$. All endpoints of the edges must be in the subset of nodes.
Definition 2.9. An ordered sequence of edges, \((x, x_1), (x_1, x_2) \cdots (x_n, y) \in E\), is called a path between \(x\) and \(y\) if each edge starts at the same node where the previous node ends. A path that does not visit a node twice is called a self-avoiding path.

Definition 2.10. A random walk is a sequence of random steps along the edges of a graph. It starts at an initial node. Then one of its edges is uniformly chosen at random. The chosen edge is traversed, reaching a new node. At the new node, the process is repeated, creating a path.

Definition 2.11. A graph \(G\) is connected if there exists a path between each pair of \(x, y \in V\).

A directed graph \(G\) is called strongly connected if for any two vertices \(x\) and \(y\), there is a directed path from \(x\) to \(y\) and a directed path from \(y\) to \(x\). A directed graph \(G\) is weakly connected if there is a directed path from \(x\) to \(y\) or from \(y\) to \(x\).

A maximal subgraph of \(G\) that is connected is called a connected component. In this case, the subgraph contains all the vertices that can be reached from a any given vertex.

2.4 Special Graphs

Definition 2.12. A simple graph is an unweighted, undirected graph without multiple edges and loops.

Definition 2.13. A tree is an simple graph in which any two nodes are connected by exactly one path. See Figure 2.5a for an example.

Figure 2.5: Examples of trees
Definition 2.14. A *path graph* is a graph that has one path through all of its nodes. All nodes except the terminal nodes have two edges. See Figure 2.5b for an example.

Definition 2.15. A *fork* is a graph that has a subgraph of $n - 1$ nodes that is equivalent to a path graph, and then the $n - 2$ node is connected to the $n$th node. All nodes except the terminal nodes and the splitting node have two edges. See Figure 2.6 for an example.

![Figure 2.6: A fork with 7 nodes that splits on node 5.](image)

Definition 2.16. A *complete graph* is a simple graph in which there is an edge between each pair of nodes. It is denoted $K_n$, where $n$ is the number of nodes. The $K_5$ graph in Figure 2.2 is complete, as is the graph in Figure 2.7.

Definition 2.17. A *regular graph* is a simple graph where each node has the same degree. See Figures 2.7 and 2.8 for two examples.

![Figure 2.7: $K_8$.](image)  ![Figure 2.8: A regular graph of degree 3.](image)
Definition 2.18. A planar graph is a graph that can be drawn on the plane so that no edges intersect. Some planar networks include road networks, river graphs, and trees, like in Figure 2.5. $K_5$ in Figure 2.2 is not planar, as discussed in Section 1.1.

Definition 2.19. A cycle is a path where the first and last nodes are equal. A cycle graph is a graph that consists of a single cycle. Graphs that do not contain any cycles are called acyclic.

![Figure 2.9: A cycle graph with 8 nodes.](image)

Definition 2.20. A hypergraph is a network in which edges, called hyperedges, can connect more than two nodes.

Many networks can be described as hypergraphs; the edges represent groups in the network. Examples include coauthor networks where there is a hyperedge between all coauthors of a paper, or movie-actor networks with edges between all actors in a specific film. An important type of hypergraph is the bipartite graph. In fact, all hypergraphs can be represented as a bipartite graph.

Definition 2.21. A bipartite graph consists of two sets of nodes with edges between the two sets only. If $U$ and $V$ are sets with edges $E = \{u, v \mid u \in U, v \in V\}$, then $G = (U, V, E)$
One example is a movie-actor network, where movies comprise one set of nodes and actors the other. There is an edge between an actor and movie if the actor appeared in that movie. This is another representation of the film hypergraph mentioned above. See Figure 2.10 for another example.

![Bipartite Graph](image)

Figure 2.10: An example of a bipartite graph. The nodes on the left belong to one set (numbers) while the nodes on the right belong to a different set (letters). The two sets are connected by the edges.

**Definition 2.22.** A *dynamic graph* is a graph that changes over time. For analysis purposes, the time period is split into discrete-time intervals, called *time periods* or *snapshots*, and the graph for that snapshot is the graph created from the nodes and edges present at the beginning of the snapshot. A graph that does not change over time, or is represented by one time period, is called *static*.

When studying dynamic networks, we can study the “dynamics on” the network or the “dynamics of” the network. The first looks at quantities that move around the networks, such as rumors over a social network, information through the internet or traffic over a transportation network. The second is the topological evolution of the network as nodes and edges are added or removed.
2.5 Matrix Representations

2.5.1 Adjacency Matrix. Graphs can be represented mathematically in many ways. One important representation is the adjacency matrix. Consider a simple undirected network with nodes $N$ and edges $E$, $|N| = n, |E| = m$. Each row and column in the $n \times n$ adjacency matrix represents one of the nodes. The entries represent the edges. So if there is an edge $(i, j) \in E$ between nodes $i$ and $j$, then the $ij^{th}$ entry of the matrix is 1. Specifically,

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Summing each column or row in the adjacency matrix of an undirected unweighted graph gives the degree, or number of connecting edges, for that node. Note that $A$ is symmetric when $G$ is undirected. The powers of adjacency matrices also represents the number of paths of length $t$. So to find the number of paths of length $t$ between nodes $i$ and $j$, we simply find the $i^{th}$ $j^{th}$ entry of $A^t$, $A^t_{i,j}$.

![Figure 2.11: Random graph.](image)

Consider the graph $G$ show in Figure 2.11. The adjacency matrix and the square of the adjacency matrix for $G$ are
\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{bmatrix}
\]

\[
A^2 = \begin{bmatrix}
3 & 1 & 1 & 3 & 1 & 1 & 1 & 2 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
3 & 1 & 1 & 3 & 1 & 1 & 1 & 2 \\
1 & 1 & 0 & 1 & 4 & 3 & 0 & 2 \\
1 & 1 & 0 & 1 & 3 & 4 & 0 & 2 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
2 & 0 & 1 & 2 & 2 & 2 & 1 & 5
\end{bmatrix}
\]

Then the number of paths of length two between any two nodes is found in \(A^2\). So there are three paths of length two between nodes 0 and 3: 0 − 5 − 3, 0 − 4 − 5, and 0 − 7 − 3.

Adjacency matrices can be adapted for weighted graphs by using the weights in the nonzero entries instead of 1s and directed graphs by using a 1 if the directed edge \((i, j)\) \(\in E\). In the unweighted directed graph case, the adjacency matrix of a directed graph is no guaranteed to be symmetric. Multigraphs can be represented by using the number of edges between the nodes.

When \(G\) is undirected or simple, \(A\) has some nice properties. One is that the eigenvalues of the \(A\) are real and bounded above by the maximum degree of \(G\). We can arrange the eigenvalues in decreasing order, so \(\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n\). Then we define the spectral radius \(\rho(A)\) to be the largest eigenvalue,

\[
\rho(A) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}.
\]

For a bipartite graph where \(U\) and \(V\) are the distinct sets of nodes and \(|U| = n_1\) and \(|V| = n_2\), we can create the adjacency matrix in the following manner:

\[
A = \begin{bmatrix}
0_{n_1 \times n_1}, & B \\
B^T, & 0_{n_2 \times n_2}
\end{bmatrix}
\]
where $B$ is the biadjacency matrix, defined as

$$B_{ij} = \begin{cases} 
1, & \text{iff } (i, j) \in E \\
0, & \text{else.} 
\end{cases}$$

Note the similarity of $B$ to the adjacency matrix for a simple graph. The only difference is that here, the columns and rows represent different nodes.

2.5.2 Laplacian Matrix. The degree matrix is a diagonal matrix that has the degree of the nodes along the diagonal.

$$D = \begin{bmatrix} 
k_1 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots \\
0 & \cdots & k_{n-1} & 0 \\
0 & \cdots & 0 & k_n \end{bmatrix}$$

where $k_i$ is the degree of node $i$.

$D$ is used to define the Laplacian matrix $L$, another important representation of a graph:

$$L = D - A.$$ 

Note that

$$L_{ij} = \begin{cases} 
k_i & \text{if } i = j \\
-1 & \text{if } i \neq j \text{ and there is an edge from } i \text{ to } j \\
0 & \text{else.} 
\end{cases}$$

The eigenvalues of $L$ are important in a number of applications. For an undirected graph, the eigenvalues are real since $L$ is symmetric, and they are also nonnegative. By definition, every row and column in $L$ sums to 0. As a consequence, $1$ is an eigenvector with 0 as an eigenvalue. This means that $L$ does not have an inverse.
2.6 Metrics

Network metrics have been studied extensively. One large group of measures that have been studied extensively are centrality measures. They attempt to measure how important nodes are in a graph. The different centralities arose from different definitions of importance. Other metrics measure distance between nodes in a network.

2.6.1 Distance Metrics. Similar to a geodesic on a manifold, a geodesic path on a network is the shortest path through the network between two vertices, and $d_{ij}$ is the length of that path. Note that the geodesic path may not be unique. The mean geodesic distance of a vertex $i$ is the sum of the geodesic lengths between $i$ and all other vertices divided by the number of vertices in the network and is defined to be

$$l_i = \frac{1}{n} \sum_j d_{ij}.$$  

Notice that $l_i$ will be low for more centrally located vertices since the central vertices are more connected and high for vertices that are on the periphery of the network. The average distance over all connected nodes is the mean distance of the connected component.

Instead of finding the shortest path between two nodes, Katz distance \cite{6, 7, 8} considers all of the paths between two nodes. It measures the distance of two nodes by looking at the number of paths between them as well as their lengths:

$$K_{ij} = \sum_{l=1}^{\infty} \alpha^l \cdot |p^l_{ij}|,$$

where $p^l_{ij}$ is the set of paths between $i$ and $j$ of length $l$ and $0 \leq \alpha \leq 1$ is a decay parameter. In the case that $\alpha < \frac{1}{\rho(A)}$, we can calculate the Katz distances between all nodes by calculating the matrix

$$K = (I - \alpha A)^{-1} - I.$$  

Alternatively effective resistance is an extension of the concept of resistance in electrical
engineering into networks [9, 10]. It is a distance measure, hence it is also known as resistance
distance. The idea behind effective resistance is to consider the network as a circuit with
each edge as a resistor. The resistance of each resistor is the reciprocal of the weight of
the edge, or 1, if there is no weight. For example, in a path graph like in Figure 2.5b, the
effective resistance between any two nodes is the length of the path connecting them.

The effective resistance matrix is

\[ R_{ij} = L_{i,i}^+ + L_{j,j}^+ - L_{i,j}^+ - L_{j,i}^+, \]

where \( L \) is the Laplacian and \( M^+ \) is the Moore-Penrose inverse [9, 11]. If \( G \) is undirected,
then \( A \) and thus \( L \) are symmetric so we can simplify to get

\[ R_{ij} = L_{i,i}^+ + L_{j,j}^+ - 2L_{i,j}^+. \]

The effective resistance \( R_{ij} \) corresponds to the expected number of times a random walk
starting at node \( i \) passes through node \( j \).

2.6.2 Importance Metrics. One way of measuring importance is by degree. This is a
natural measure in a social network; people who are connected to lots of other people could
be more influential, have more access to information, or be more popular. For example, in
a citation network, papers that have a high in-degree have been cited more often, meaning
that they are more influential. Degree centrality can be represented by the degree of the
node, or it can be normalized by taking the number of neighbors divided by the number of
nodes in the graph. With a directed network, this can be split into in-degree centrality and
out-degree centrality, the number of edges coming into the node and leaving the node.

Eigenvector centrality extends degree centrality by considering the importance of a node’s
neighbors as well as the degree of a node. For example, if node \( v \) has 6 neighbors, 5 of which
are only connected to \( v \), then node \( v \) is probably not as important as node \( w \) who only has
3 neighbors who all have degree 3 as well.

The eigenvector centrality $x$ is the vector of centralities that satisfies

$$Ax = \rho(A)x.$$ 

Equivalently, it is also the sum of centralities of its neighbors.

$$x_i = \rho(A)^{-1} \sum_j A_{ij}x_j.$$

For a directed network, we specify that $x$ is the right leading eigenvector, so that the eigenvector centrality for a vertex is based on incoming edges.

One issue with eigenvector centrality is that in a directed network, a node can have 0 eigenvector centrality, even though it has an incoming edge. Consider the directed network shown in Figure 2.12, where A has three outgoing edges to B, C, and D and no incoming edges. B also has three outgoing edges, as well as one incoming edge.

![Figure 2.12: A directed network where nodes A, B, C, and F have an eigenvector centrality of 0.](image)

The eigenvector centrality for $B$ is given by

$$x_B = \kappa_1^{-1} \sum A_{Bj}x_j$$
\( A_{Bj} \) is zero for all \( j \) except \( A \), so \( x_B = \kappa^{-1}_1 x_A \). However, \( A \) has no incoming edges, and thus \( x_A = 0 \), meaning \( x_B = 0 \).

*Katz centrality* addresses this by adding a small amount of free centrality to node so that no node has 0 centrality [6]. The Katz centrality for node \( i \) is defined to be

\[
x_i = \alpha \sum_j A_{ij} x_j + \beta
\]

where \( \alpha \) is a decay parameter and \( \beta \) is a vector that controls the initial centrality.

We can calculate the Katz centrality of each node at one time by writing

\[
K = \alpha Ax + \mathbf{\beta}
\]

Through rearranging and setting \( \beta = 1 \) for convenience, we get the Katz vector

\[
K = (I - \alpha A)^{-1} I
\]

where \( 1 \) is the vector of ones. Note the similarity to the Katz distance,

\[
(I - \alpha A)^{-1} - I.
\]

To calculate Katz centrality, one must choose a value for \( \alpha \). If \( \alpha \) goes to 0, \( K \) converges to \( \beta = 1 \), with each node having the same centrality. When \( \alpha \geq \frac{1}{\rho(A)} \), \( K \) diverges because \( \det(I - \alpha A) \) passes through 0.

One effect of Katz centrality is that a vertex with high centrality passes on high centrality to all of its outgoing neighbors. This is especially noticeable when a node has an extremely large number of outgoing edges that are not particularly important; for instance, Barack Obama at one point had the most followers on Twitter (over 130 million), and most likely has a high Katz centrality. However, he also follows around 590000 people, most of which only have a few hundred followers. Using the Katz centrality, they all would have a high
centrality, even if they are not relatively important.

One way to mitigate this is to divide the Katz centrality from a neighbor by their out-degree, $d_j^{\text{out}}$.

$$x_i = \alpha \sum A_{ij} \frac{x_j}{d_j^{\text{out}}} + \beta$$

If $d_j^{\text{out}}$ is 0, then we set $\frac{x_j}{d_j^{\text{out}}}$ to 0. This centrality is called Pagerank, made famous by Google.

*Betweenness centrality* measures how important a vertex is by looking at how many geodesics in the network it lies on. Consider an information network, where information is passed between the vertices along the edges. Assume that during every time period, each pair of connected vertices exchanges a message with equal probability and that the message travels along a geodesic. After a sufficient amount of time, we can estimate how many messages have passed through each node and thus the number of geodesic paths the vertex is in. This number is the betweenness centrality. Formally, define

$$n_{st}^i = \begin{cases} 1 & \text{if } i \text{ is on the geodesic between } s \text{ and } t \\ 0 & \text{otherwise.} \end{cases}$$

Then the betweenness centrality of vertex $i$ is

$$x_i = \sum_{s,t} n_{st}^i.$$ 

In certain directed networks, a node can be considered important if it points to other important nodes. For example, in the world wide web, there are web pages whose content consists mostly of links to other important sites. The pages by themselves are not important, but they point to important pages. We call the important pages that are linked to *authorities* and the pages that point to the authorities *hubs*. The hub and authority centralities, $x$ and $y$ respectively, are found using the hyperlink-induced topic search algorithm, HITS, and satisfy the following equations:

$$x = \alpha Ax, y = \beta A^T y$$
\[ AA^T x = \lambda x, \quad A^T A y = \lambda y \]

where \( \alpha \) and \( \beta \) are constants and \( \lambda = (\alpha \beta)^{-1} \). Both conditions are satisfied only if \( AA^T \) and \( A^T A \) have the same leading eigenvalue \( \lambda \). This is the case, as we can see by multiplying \( AA^T x = \lambda x \) by \( A^T \).

\[ A^T A (A^T x) = \lambda (A^T x) \]

so \( A^T x \) is an eigenvector of \( A^T A \) with eigenvalue \( \lambda \).

Closeness centrality deviates from the above centralities and focuses on the distance between vertices instead of degrees. Recalling that the mean geodesic distance of a node is

\[ l_i = \frac{1}{n} \sum_j d_{ij}, \]

closeness centrality is the inverse of the mean geodesic distance, \( C_i = \frac{1}{l_i} = \frac{n}{\sum_j d_{ij}} \). When a node is more central (or connected) in the network, \( l_i \) is low. This means that a lower mean distance corresponds to more important vertices, i.e., vertices that are more influential since they can reach more vertices quickly.

2.6.3 Other Metrics. Transitivity is the notion that a neighbor of my neighbor is also my neighbor. Perfect transitivity, where a vertex is connected to all of its neighbors’ neighbors, only occurs in a network in which each connected component is complete. However, partial transitivity is useful, especially in social networks; the chance that I know my friend’s friend is much higher than the chance that I know a random person. Transitivity is manifested in networks by three nodes that are connected, or triangles. In a social network, this is sometimes referred to as a closed triad.

Since triangles are closed paths of length three, we can measure the amount of transitivity in the network by counting the ratio of closed paths of length two over all paths of length...
two. This measure of transitivity is called the clustering coefficient:

\[ C = \frac{\text{number of paths of length two that are closed}}{\text{number of paths of length two}}. \]

\( C = 1 \) implies the network is made up of complete subgraph components, while \( C = 0 \) means that there are no triangles, such as in a tree. Unlike the other metrics in this section, transitivity is calculated for \( G \) and not for individual nodes, although there are formulas for calculating the transitivity, or local clustering coefficient, of individual nodes.

Another metric of interest in directed networks is how often a node’s neighbors also connect to them. For example, on social media, if node A follows node B, then does node B also follow node A? If so, then the edge is reciprocated. Unsurprisingly, when two nodes are connected by an edge in one direction, it is much more likely that they will be connected the other direction than if they were not connected to begin with. We can calculate the average reciprocity of a network using the number of edges in the graph, \( m \), as follows:

\[ r = \frac{1}{m} \sum_{ij} A_{ij} A_{ji} = \frac{1}{m} \text{Tr}(A^2). \]

Most social networks display assortative mixing, where people associate with others who are similar. Think of high school and the cliché cliques. High school students hang out with their respective group. There is some overlap between certain groups, but they are mostly isolated. The assortativity coefficient captures this behavior by using either node attributes or the network structure. It is defined here where \( c_i \) is the class of of node \( i \) and \( m \) is again the number of edges:

\[ Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j). \]

We note that the assortative coefficient is interchangeable with modularity, which is more often used.
2.7 Similarity

Similarity in a graph is the study of how nodes are similar within a network as well as how similar two networks are. There are many ways to define similar nodes, and we can identify several categories of similarities.

2.7.1 Similar Networks. Network analysis involves comparing and contrasting networks. We can identify similarities and differences by using the global metrics described in Section 2.6 and the adjacency matrices. Further, two networks $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are isometric if there is a bijective mapping $F : V_G \rightarrow V_H$ that preserves the edges: $(v, w) \in E_G$ if and only if $(f(v), f(w)) \in E_H$. Identifying isometric networks gained popularity as a computational problem in the 1970’s [12].

2.7.2 Structural Equivalence. Structural equivalence partitions a graph’s nodes into equivalence classes of neighbors; two nodes are structurally equivalent if and only if they share the same neighbors. This definition is very strict, so it is often relaxed to sharing many of the same neighbors. Consider the graphs in Figure 2.13. Figure 2.13a shows strict structural equivalence classes. The two red nodes have exactly the same neighbors, while the blue nodes also have the same neighbors. In Figure 2.13b, the red and blue nodes are strict equivalence classes, while the black and yellow nodes are relaxed equivalence classes.

Regular equivalence further relaxes the definition. Two nodes are regularly equivalent if they have neighbors who are similar. For instance, two actors might not have any friends
in common, but they are similar in that they have worked with many people in the film
industry, including directors, producers, costume designers, and so forth.

One measure of structure equivalence is the number of common neighbors. Cosine simi-
larlity normalizes the number of same neighbors by adjusting for the degrees of the nodes.
It is based off of the inner product of two vectors and defined by

$$
cos(\theta) = \frac{x \cdot y}{|x||y|}.
$$

To apply this to a graph, the $i$th and $j$th columns of the adjacency matrix are used as the
$x$ and $y$ vectors. Then the cosine similarity becomes

$$
\sigma_{ij} = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A^2_{ik}} \sqrt{\sum_k A^2_{jk}}},
$$

For an unweighted simple graph, we can further simplify this to

$$
\sigma_{ij} = \frac{n_{ij}}{\sqrt{d_i d_j}},
$$

where $n_{ij}$ is the number of common neighbors and $d_i$ is the degree of node $i$. When the
cosine similarity is 1, the nodes have exactly the same neighbors. A cosine similarity of 0
means the nodes do not have any neighbors in common.

2.7.3 Node Attribute Similarity. We can also use node attributes to identify similar
nodes. Node attributes are features given to each node that describe the node in a non-
structural way. For instance, in a social network, each node may have an attribute with the
number of minutes the person spends on the site each day or the number of times they’ve
edited posts in the last month. Attributes like this can provide some insight into the nodes,
but aren’t essential to the network’s structure. Including them in similarity analysis can
help make more meaningful comparisons [13, 14].
2.8 Network Theory Problems

2.8.1 Weight Matching. Weight matching is the network theory version of the assignment problem. Examples include the famous stable marriage problem [15] and the rank-maximal allocation problem [16]. The assignment problem assumes that there are two sets of possibly different sizes, and items in the set can be matched according to cost constraints. The goal is to match every item from the smaller set to a unique item from the larger set while minimizing, or maximizing, the cost. In network theory terms, this is equivalent to minimizing (maximizing) the sum of the edge weights in a weighted bipartite graph matching. If the two sets are the same size, we say the problem is balanced. Otherwise, it is unbalanced. When the sum of the cost of the assignments for the items in both sets are equal, the problem is linear and is called linear assignment. Usually, the assignment is balanced and linear, unless otherwise indicated. There are several algorithms to solve the weight matching problem, including linear programming, the Hungarian algorithm, also known as the Kuhn-Munkres algorithm or Munkres assignment algorithm [17], and the Jonker-Volgenant algorithm [18].

2.8.2 Link Prediction. One common area of study for networks is link prediction [7]. Link prediction is split into two main questions: predicting the existence of an edge between two nodes, and predicting which edges are going to be created in the next snapshot. As an example of the former, on social media a user might get recommendations for new friends. An example of the latter is predicting future coauthorships in a citation network. Predicting the existence of an edge is a type of recommendation algorithm. Link prediction is also used in bioinformatics, e-commerce, and database duplication. Techniques for solving link prediction problems include using random walks, similarity algorithms such as Katz and effective resistance, and matrix factorization [19].

2.8.3 Community Detection. Sometimes with a network, we want to identify groups of nodes that are more connected within the group than to nodes in other groups. Com-
Community detection, or graph partitioning, is the process of splitting the nodes of a network into such groups, also called communities. This is similar to clustering algorithms, discussed in Section 3.5, in which node attributes and other data are used to cluster the nodes into groups. Community detection however, is a network specific subset of clustering algorithms that focuses almost exclusively on the structure of the graph. A few algorithms for community detection include the Kernighan-Lin algorithm, spectral partitioning, modularity maximization, and neural network clustering techniques.

2.8.4 Role Discovery. While community detection focuses on finding groups that are more connected inside the group than to nodes outside the group, role detection looks at identifying nodes that are similar. Similarity in this case can mean that they have the same connections, same type of connections, or function the same way in the network. Some interesting questions in this area are “When will a node transition roles?” and “What role will a node have in a future time period?”

Roles in networks originated in sociology as a way to cluster similar groups inside a social network [14, 20]. For instance, in a family tree, roles might include mother, father, and child, and in a pandemic, superspreader, isolater, and health care worker could be roles. As technology has advanced, role discovery has expanded to large, complex networks like online social networks like Twitter and Facebook [21]. While role discovery was first focused on identifying similar roles based on structural equivalence, as discussed in Section 2.7, most current research includes node attributes [22, 14]. Using non-negative matrix factorization allows the unsupervised role detection problem to become semi-supervised, and is used in many algorithms [23, 22, 24, 25]. However, these algorithms only address static networks. Since networks exist in a world that is constantly changing, looking at static networks only gives us information about the past and current states of the network. In the past decade, role discovery has been looked at in dynamic networks through some specific algorithms [24, 26, 25].
Chapter 3. Machine Learning

Machine learning is a method of creating a computer model that answers some question by applying an algorithm to data. It does this by identifying and learning patterns in the data that it can apply to new data. Arthur Samuel, an artificial intelligence pioneer said in 1959 that machine learning is the “study that gives computers the ability to learn without being explicitly programmed” [27]. Machine learning affects our lives every day from filtering spam in our email, recognizing when we tell our phone to call a friend, recommending what music to listen to, detecting credit card fraud, forecasting stock prices, and many more. The field of machine learning is huge, so in this chapter, we will only give a brief overview of important concepts and algorithms that will be relevant to subsequent chapters. Most of the information can be found in the highly-rated “Hands-on Machine Learning with Scikit-Learn, Keras, & TensorFlow” by Aurelien Geron [27], the award winning “Machine Learning” by Zhi-Hua Zhou [28] and the comprehensive “Deep Learning” by Ian Goodfellow, Yoshua Bengio and Aaron Courville [29].

3.1 Machine Learning Concepts

3.1.1 Supervised v. Unsupervised Algorithms. Machine learning algorithms are classified into two categories based on the type of data they use. Supervised algorithms use data that includes the inputs, also known as features, and the category that is being outputted or predicted, called the label. For instance, a label in a cancer dataset is whether each image of a cell is cancerous or not, or a label in a dataset about houses sold could be the selling price. These algorithms attempt to find the function that relates the inputs to the labels and predicts what the label should be. They are called supervised algorithms because by using the labels, which we can think of as answers, they get feedback on how good the model is at making predictions.

Algorithms that do not use labels are called unsupervised. Instead of trying to predict
an output, they try to identify patterns from the inputs. As a result, they are more difficult
to evaluate and can be less accurate. However, as many datasets do not have labels, they
have grown in use and there are some very accurate unsupervised algorithms.

3.1.2 Types of Machine Learning Problems. There are four types of problems that
machine learning algorithms solve: classification, regression, reinforcement learning, and time
series predictions. Algorithms that solve classification problems take datasets comprised of
discrete classes and learn the traits of the classes so that they can take new data and sort it
into the correct class. Binary classification algorithms have only two outcomes, usually true
or false or positive and negative. Multinomial classification algorithms have more than two
outcomes. Use of classification algorithms include facial recognition, cancer detection, fraud
detection and labeling documents by topic.

Regression algorithms are used when the data is continuous. These algorithms are com-
mon when trying to identify a price or cost, such as the cost of a house given certain
parameters like number of bedrooms, square feet, and so on. It can also be used to measure
a stock’s volatility, return on investment (ROI) in businesses, and to make predictions in
weather and sea levels.

Reinforcement learning attempts to learn the optimal policy, a set of actions that max-
imizes reward, given certain constraints, over a set amount of time. It is typically framed
as a Markov Decision Process and can sometimes be solved using traditional dynamic pro-
gramming approaches. A famous reinforcement learning problem, The Secretary Problem,
asks, “How many secretaries should you interview before hiring one?” [30]. Another common
reinforcement learning problem is how to navigate a robot around obstacles.

Time series algorithms also deal with time. A time series is a dataset that splits time into
discrete intervals. Unlike reinforcement learning which consists of constraints and rewards,
time series datasets include features that are timestamped. Problems involving these time
series are also either classification or regression problems, and the objective is usually to
predict or forecast the label. Even though classification and regression algorithms can be
applied to time series, algorithms that are specific to time series data often perform more accurately.

3.1.3 Evaluating Models. Determining the validity of a model depends on the type of model created and the data used. When ground truth is available, either in a supervised learning model or an unsupervised model in which the data contains labels that aren’t used, validation is more meaningful and accurate. Machine learning algorithms are optimization algorithms, with the goal of minimizing or maximizing a function, called the loss, or cost, function. In most cases, the loss function measures the error in the algorithm, so the objective is usually to minimize the loss function.

**Gradient Descent.** Gradient descent, also called steepest descent, is one of the most popular algorithms that minimizes loss used in machine learning. Given a multivariable function defined and differentiable in a neighborhood of a point \( x \), then the function will decrease most quickly in the opposite direction of the gradient. By taking a small step in this direction, and repeating the process, we will hopefully converge to a local minimum. This is not guaranteed, and changing the step size can have an impact. One of the main difficulties with gradient descent is that it is slow, especially on large datasets. Speed can be improved using variations, including conjugate gradient and stochastic gradient descent. Conjugate gradient converges in fewer iterations by requiring that the search directions are orthogonal. Stochastic gradient descent uses the idea that the gradient is an expectation, which can be approximated using only one to one hundred samples. Using a small set of samples reduces the time to calculate the derivative so it is much faster than gradient descent, even though it usually takes more iterations to converge. While other optimization methods exist, such as Newton’s method, gradient descent variations are the most common means of minimizing loss functions.

**Loss Functions.** Loss functions measure the difference between a model’s prediction and the correct label. Some loss functions are convex, which means that any local minimum is also a global minimum. Further, a strictly convex function only has one global minimum,
so finding the minimum is efficient using local optimization methods. While it is desirable to use a convex loss function, it is not always possible. Some convex loss functions include the L1 and L2 norms and binary logistic loss. Non-convex loss functions are used in neural nets, and because of the non-convexity, it is possible to have an exploding or vanishing gradient. In the first case, the weights in the network become large, causing the gradient to become very large, while in the second, the gradient of the weights very small.

Mean square error, or MSE, is a common loss function and the default one used in regression algorithms. It handles outliers well, but is not as good when datasets are noisy. It is the Euclidean distance, or L2 norm between two points. Here $y$ is the predicted value and $y'$ is the actual label and $n$ is the number of data points.

$$MSE = \frac{1}{n} \sum_{0}^{m} (y - y')^2$$

Root mean square error is a variation of MSE in which the square root of the MSE is taken and is the average distance of of a data point from the predicted value.

A similar loss function is the mean absolute error, MAE, which is average of the absolute value of the differences

$$MAE = \frac{1}{n} \sum_{0}^{m} |y - y'|.$$ 

This error, also known as the L1 norm, is not as good at dealing with outliers and can occasionally cause very poor predictions. It is often used when dealing with time series because it estimates how much error we can expect from a forecast on average.

Logistic loss, used mainly in logistic regression, is

$$\sum_{i=1}^{n} -y_i \log(y'_i) - (1 - y_i) \log(y'_i).$$

In logistic regression, the nonlinear sigmoid function,

$$S(x) = \frac{1}{1 - e^{-x}}$$
is used at the final step to convert the output to values between 0 and 1. This makes the model non-convex, so using MSE as the loss function makes it very difficult to find the global minimum. Additionally, the error values would always be between 0 and 1, making it difficult to reduce error.

An extension of logistic loss, cross-entropy loss comes from information theory and is used for classification problems, especially in neural networks. In information theory, the more surprising an event is, the more information is needed to describe it. The information of an event $x$ is the negative log of the probability of the event occurring:

$$h(x) = -\log(p(x)).$$

The entropy measures the information from a probability distribution. In a distribution where the likely events happen more often, the information needed to describe the distribution is lower, and thus the entropy will be lower. Cross entropy is the difference between the entropy of two distributions, the true distribution and the estimated distribution

$$CE = - \sum_{i=1}^{n} y_i \log(p(y'_i)).$$

When there are only two classes, this becomes the binary cross entropy loss:

$$BCE = - \frac{1}{n} \sum_{i=1}^{n} y_i \log(p(y'_i)) - (1 - y_1) \log(1 - p(y'_1))$$

since $y'_2 = 1 - y'_1$ and $y_2 = 1 - y_2$.

**Classification.** While some classification machine learning models minimize loss functions, like logistic regression, others like random forest do not. These algorithms need a different way of determining how accurate a model is since the difference between the predicted label and actual label is ambiguous. For instance, how would you numerically measure the difference between a cat and dog? When analyzing a binary classification model, there
are four possible outcomes. True positives are the items that were correctly classified as positive. False positives are the items that the model incorrectly identified as being positive, meaning the ground truth was negative, but the model predicted it was positive. True negatives are correctly classified as having a negative label, and finally false negatives are incorrectly predicted to be negative.

The basic metric used is *accuracy*, which is the percentage of correct predictions the model made.

\[
\text{Accuracy} = \frac{\text{true positives}}{\text{true positives + false positives + false negatives + false positives}}
\]

However, sometimes accuracy can be misleading. Consider the following scenario in which we want to predict if an airline traveler is a terrorist. We could create a basic model that classifies every traveler as not a terrorist. This model will have a very high accuracy since over 99% of airline flyers are not terrorists. However, this model is not useful as it doesn’t stop the very few terrorists that do fly. In this case we only have a few terrorists compared to the millions of people who fly every year in our dataset, meaning that the data is imbalanced. Imbalanced classification datasets are not suited toward accuracy as a measure of goodness for this reason.

Instead of looking at accuracy, we could look at *recall*, the number of items the model correctly predicted was positive over the number of items with an actual positive label.

\[
\text{Recall} = \frac{\text{true positives}}{\text{true positives + false negatives}}
\]

In the terrorist scenario, this would be the number of terrorists the model correctly predicted
over the total number of terrorists. The basic model classifying all travelers as non-terrorists has a recall of 0, which is not very good. We could adjust our model to another simple model: classifying each traveler as a terrorist. This would give us very low accuracy, less than .01 and a recall of 1, the maximum possible recall.

Once again, this is not a very enlightening model, and another metric, precision, illustrates that. Precision is the number of true positives over the number of items the model predicted were positive.

\[ \text{Precision} = \frac{\text{true positives}}{\text{true positives + false positives}} \]

In the first model where no travelers were classified as terrorists, the precision was 0. In the second model, the precision is very low, less than .01 since the denominator is the size of the dataset.

There is a trade-off between precision and recall; as precision increases, recall decreases and vice versa. We can measure this with the F-score, also called the F1 score:

\[ F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

This balanced F-score is the harmonic mean of the precision and recall. However, there may be situations where recall or precision is a more important metric. Then we can weight the F-score with \( \beta > 0 \) to get the general \( F_\beta \) score

\[ F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}} \]

Here \( \beta \) represents how much more important recall is than precision. So if recall is 2 times as important than precision, \( \beta = 2 \), and if precision is twice as important than recall, \( \beta = .5 \).

This method can be extended to multinomial classifications. The true positives and true negatives become the true classes, for however many classes are in the dataset. Instead of
false positive and false negatives, we have false class 1, false class 2, etc. We can visualize this easily with a confusion matrix. A confusion matrix shows the number of correct predictions for each class along the diagonal. The rows indicate how many of a class were classified as the other classes and the columns show how many of each class the model predicted. Figure 3.1 is an example of a confusion matrix. From the diagonal, we can see the model correctly predicted a high number in each class. Then we can see that the model predicted 4 class 0 items as class 1 and none as class 2. Similarly, we can extend the precision, recall, and F-score by calculating them globally (summing rows for recall and columns for precision), calculating the metrics for each label individually and then averaging them for a global metric, or calculating the metrics for each label and averaging a weight to account for imbalanced data.

![Confusion matrix for a Random Forest model on the Sklearn Wine Dataset.](image)

Figure 3.1: Confusion matrix for a Random Forest model on the Sklearn Wine Dataset.

### 3.1.4 Bias Variance Trade-off

The error measured by a loss function can be interpreted as the sum of irreducible and reducible error. Irreducible error is the error that is always present due to the nature of machine learning. It can be caused by noise in the dataset, randomization in an algorithm, or rounding error. Per its name, it cannot be mini-
Reducible error can be minimized. It is the sum of bias squared and variance. Bias is the error that occurs due to incorrect assumptions and measures the average error of the model. When bias is high, the model makes more assumptions about the target function, simplifying the model. Regression algorithms and linear neural networks have a high bias, while nonlinear neural networks and decision trees have low bias.

Models with low bias have high variance, and vice versa. Variance is a measure of how well a model generalizes by looking at the variability of predictions. High variance models do not generalize well because they focus heavily on the training data. This is called overfitting because the model focuses too much on the seen data and does not perform well on new data. Underfitting is when a model fails to identify patterns in the training data and occurs when the bias is high and variance low.

Since

\[ \text{error} = \text{variance} + \text{bias}^2 + \text{irreducible error}, \]

reducing error equates to reducing variance and bias. However, due to the opposing definitions of bias and variance, reducing one comes at the cost of increasing the other. Therefore, minimizing error will minimize some combination of bias and variance.

### 3.2 The Machine Learning Process

Building a machine learning model consists of several steps. The gist of these steps is the same, whether you’re training a chatbot, recommendation system, facial recognition, or a Go model. Each of these steps is important to create the most accurate model while mitigating bias.

#### 3.2.1 Collecting and Cleaning Data.

In some ways, data can be the most important aspect of building a machine learning model. Without it, the model obviously doesn’t exist. If the data is incomplete, inaccurate, or unrepresentative, the trained model will be too.

In some instances, such as working for a company that wants you to analyze a specific
dataset, the data might be given to you. For research, there are many datasets online, and
some public data can be scraped. Even in the first instance, cleaning the data, or making
sure the data is complete and makes sense, is vital. The following is a list of steps that
should be taken with each dataset before using it to train a model.

- Handle missing data: Remove data points or fill in the missing data.
- Correct consistency in the data: Ensure data has the same format, such as with dates,
names and capitalization.
- Handle invalid data: Verify that data is the type and in the right ranges. For instance,
an *age* column measured in years should only have nonnegative integers less than 120.
- Handle duplicated, irrelevant or unwanted data: Before removing such data, one should
verify that it is not needed and was not duplicated or irrelevant for a certain reason
that may impact the model.
- Handle outliers: Generally, outliers are not removed unless there is a significant reason
to remove them.
- Handle misencoded and ordinal data: Ordinal data is categorical data that has a
meaningful order which doesn’t have an obvious numerical equivalent. For example,
data that can take the values of *beginner*, *intermediate*, *advanced* is ordinal. Since
data must be numerical to use in an algorithm, it is tempting to assign these categories
the values of 1, 2, and 3 respectively. This is an example of misencoded data, in which
nonnumerical data is assigned numerical values. However, this system has biases that
can impact the results. A better approach is to one-hot encode the variables. This
is done by creating three new binary columns, one for each category, with a 1 in the
column that represents the value for each data point.
- Remove unnecessary features: Determine if there are any features that are not relevant
to your problem. This can be done using feature selection algorithms.
• Create new features: Called feature importance, this involves aggregating multiple features to create a new feature that represents all of the other features.

• Augment data: Generate new data to increase the number and variability of the data points.

3.2.2 Creating the Model. After the data is collected, an algorithm is chosen based on the type of problem as described in Section 3.1.2, the question being asked, the data, and the resources available. All machine learning algorithms require hyperparameters, variables in the machine learning model that are independent of the data. One important hyperparameter is the learning rate. It determines how big a step to take when doing gradient descent. As discussed in Section 3.1.3, this can have a large impact on the model. Since hyperparameters affect the outcome, good hyperparameters are found by taking a small sample of the data and running the algorithm with different values of these variables in a method called a grid search. The variation with the best results are selected to use on the actual training.

Once hyperparameters are identified, the model can be trained. The purpose of the model affects how it is trained. Deep learning models that take large amounts of computing power and time to train usually only train Typically, machine learning models are trained multiple times to account for randomness in the algorithms. Large deep learning models may only train a few times due to the large amount of processing power and time needed, while smaller models can be run more times. This gives a sense of how typical the models are and ensures the model is not an outlier. With a few exceptions, data is split into training and validation sets each iteration for consistency and replication. The data is trained on the training set and the validation set is used for analyzing, ensuring that the model is given data it hasn’t seen before when determining how good it is. If the data is labeled, the model is compared against the test data the error is calculated. If not, then the models are compared against some metric or against other baseline models or datasets. Models are generally considered successful if they increased accuracy, decreased loss, or improved computational time or
3.3 Regression

Regression is a predictive technique that attempts to find a relationship between a set of independent (predictor) variables and a dependent (response) variable. It can be used to forecast the effects of a single change, like predicting total revenue if the price of a product is changed, forecasting future values like stock prices, and determining how much an independent variable affects the dependent variable. An example of the last use case is determining which marketing type affects total sales most. There are two main types of regression: logistic regression and linear regression.

3.3.1 Linear Regression. Linear regression solves the least squares problem

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \cdots \beta_n x_n \]

by minimizing the sum of squared residuals

\[ \sum (y_i - \hat{y}_i)^2 \]

where each \( x_i \) represents the data values for a specific feature and \( y = (y_1, \ldots, y_n) \) is the actual value and \( \hat{y} = (\hat{y}_1, \ldots, \hat{y}_n) \) is the predicted value. In order to apply linear regression, several assumptions must be met.

- Dependent variable is continuous and numeric: This is necessary to formulate a least squares problem.

- Linearity: The relationship between the dependent and independent variables is linear i.e. the dependent variable is linear combination of the independent variables. If this is not true, due to the data or feature engineering, linear regression can still be use by
applying a nonlinear transformation to the data.

- **Homoscedasticity:** The variance of the residuals is constant, meaning that there are not big outliers in the dataset.

- **Independence of Residuals:** The residuals are uncorrelated. If violated, the accuracy of the model decreases.

- **No multicollinearity** - The least squares problem requires X to have full column rank, otherwise there is a linear relationship between some of the independent variables.

- **Weak exogeneity:** Independent variables are fixed, not random variables. This means that we assume there is no measurement error. This is rarely realistic but greatly simplifies the problem.

If the assumptions are met, then the least squares problem can be solved iteratively, using gradient descent. Linear regression tends towards overfitting, so mitigation techniques, especially regularization, should be evaluated.

**Ridge Regression.** Ridge regression, a regularization of linear regression, is used when independent variables are multicollinear. It minimizes sum of squared residuals with a regularization term, called an L2 penalty:

\[
\sum (y_i - \hat{y}_i)^2 + \lambda \sum (\beta_i^2).
\]

Here \(\lambda\) is a shrinkage parameter and will cause the regularization term to go to 0, keeping the model weights small as \(\lambda\) increases. By doing this, the variance decreases more than the bias increases, resulting in a lower loss.

**Lasso Regression.** Another regularization is called LASSO, least absolute shrinkage and selection operator, regression. It adds an L1 penalty to the regularization term:

\[
\sum (y_i - \hat{y}_i)^2 + \lambda \sum (|\beta_i|).
\]
As \( \lambda \) increases, independent variables that aren’t important go to 0 and can be dropped from the model. Thus lasso regression can be used for feature selection. For this reason, Ridge regression performs better when there are a small number of independent variables or in situations with high multicollinearity. Lasso regression is better when there are a lot features and you want a simpler model. One important distinction is that lasso regression cannot be solved using using gradient descent like linear or ridge regression because the absolute values make the loss function undifferentiable at 0. There are two variations, subgradient descent and coordinate descent, that can be used instead.

### 3.3.2 Logistic Regression

Logistic regression is an algorithm that classifies data into binary outcomes such as True and False, or 1 and 0. It is based off of the logit function, defined as the inverse of the standard logistic function where \( p \) is the probability that the outcome is 1, \( p = P(Y = 1) \). In this definition, \( e \) is used as the base, but any base can be used.

\[
\text{logit}(p) = \left( \frac{1}{1 + e^{-x}} \right)^{-1} = \ln \left( \frac{p}{1 - p} \right).
\]

Logistic regression has several assumptions that should be met before assuming a good model.

- No multicollinearity between the independent variables.
- The independent variables, \( \beta_i \)'s, should be linearly related to the log-odds of the dependent variable.
- Large sample size.

The second assumption above is critical and yields the following equation:

\[
\ln \left( \frac{p}{1 - p} \right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n b_n = \beta_0 + \sum_{j=1}^{n} \beta_j x_j = \beta x.
\]
After simplifying, we can write the probability \( p \) in terms of the logistic sigmoid function \( S(x) \),
\[
p = \frac{1}{1 + e^{-\beta x}} = S(\beta x).
\]
Calculating the \( \beta \) coefficients is the bulk of logistic regression and can be done using iterative or optimization methods, such as maximum likelihood estimation or gradient descent to maximize the log likelihood since a closed-form expression is usually impossible to find.

Multiple logistic regression can be used for multiple classification problems by using the softmax function instead of the sigmoid function. First, a softmax score is calculated using parameter vectors \( \beta_k \) for each class \( k \):
\[
s_k(x) = (\beta_k)^T \cdot x.
\]
Then the softmax function estimates the probability \( p_k(x) \) that \( x \) belongs to class \( k \).
\[
p_k(x) = \frac{e^{s_k(x)}}{\sum_{j=1}^{c} e^{s_j(x)}}
\]
where \( c \) is the number of classes. The algorithm returns a probability vector where the \( i \)th entry is the probability that the data point is in the \( i \)th class.

3.4 Classification

Classification is another predictive technique that looks for relationships. In this situation, the data is grouped into different classes, which the algorithm tries to predict based on the data’s features. The classic example is to predict which class an image belongs to. So given a set of pictures with different animals, can a model learn good enough patterns in the images of each animal to correctly identify which animal is in an image it hasn’t seen yet? The similar problem of facial recognition is also classification. A different type of classification problem is identifying emails as either spam or legitimate.
3.4.1 Decision Trees. A decision tree is a supervised learning algorithm that can be used for classification or regression predictions. It is basically a flowchart that asks a series of yes/no questions to determine the prediction. Decision trees are popular because they are easy to understand and can parse non-linear relationships well. The tree takes input data, moves through the various decisions nodes of the tree to a leaf, and returns as output the most common result in that leaf. At each level of a classification tree a decision node is constructed by considering the features not already split on, choosing the best feature to split on, and then choosing the optimal split point [31].

To choose the optimal split, a decision tree calculates either the Gini impurity score or the information gain for the dataset. Due to speed and processing time, Gini impurity is the preferred method of calculating splits. The Gini impurity calculates the probability of incorrectly classifying a data point based on the class distribution:

\[
\text{GI}(X) = \sum_{i=1}^{c} p_i (1 - p_i) = 1 - \sum_{i=1}^{c} p_i^2
\]

where \(p_i\) is the percentage of datapoints in \(X\) that have a class \(i\) label and \(c\) is the number of classes. When all labels are the same, the Gini impurity is 0, meaning it they are homogeneous. Entropy measures the purity of a split; the more homogeneous a set is, the less information is required to describe it. Using the same notation, the entropy of a dataset is

\[
H(X) = -\sum_{i}^{c} p_i \log_2 p_i.
\]

Information gain compares two sets and determines which requires more information, based on entropy.

\[
\text{IG} = 1 - H(X).
\]

To determine the best split, a tree either calculates either the information gain or the Gini impurity of all possible ways of splitting the data in two and chooses the split that minimizes the result, maximizing the homogeneity of the labels. This splits the data into
two sections, known as branches, and the process is repeated on each branch until there is only one data point left or some criteria is met. When the data cannot be split anymore, the branch ends in a leaf.

Decision trees are prone to overfitting; if the data was split until each branch ended in one data point, the tree would be highly correlated to that specific dataset. To compensate, decision trees have several parameters that can help offset the overfitting.

- min_split is the number of datapoints that must be in each split.
- max_depth is the number of layers in the tree, or the maximum number of questions that can be asked before predicting the label. This restricts the tree from becoming too deep and specific.
- min_leaves is the minimum number of datapoints that a leaf can have. This prevents leaves having only one datapoint.

There are several modifications to decision trees that help with overfitting and improve accuracy, and as a result, decision trees as described here are rarely used.

Random decision trees randomize the creation of decision trees in two ways. The first is that a random subset of the data is sampled and from that subset a classification tree is created. A standard classification tree considers all of the remaining features when deciding which feature to split on. However, this often results in trees that are highly correlated. Instead of using all the remaining features at each level of the decision tree, a random decision tree also chooses a random subset of the remaining features to split on. This offers two advantages, the first is a collection of uncorrelated trees, and the second is splitting on fewer features results in faster algorithms.

3.4.2 Random Forests. As mentioned, decision trees are prone to overfitting. Although modifying hyperparameters can help, random forests offer another alternative. Random forests are an ensemble algorithm, in which a collection of algorithms are used to create
a better algorithm than each one individually. It operates by having many decision trees, which are trained on different random parts of the training set, and “vote” on the final classification [32]. A typical random forest consists of thousands of these voting decision trees, and it is typically the case that some of them are actually good models. Random forests rely on Condorcet’s Jury Theorem from political science which guarantees a collection of weak voters will arrive at the correct decision with high probability [33].

A variation on random forests, extremely randomized trees algorithm, instead of looking for an optimal splitting threshold for each feature at each step of the decision tree, thresholds are created at random. The splitting rule for each tree is chosen to be the best of these random thresholds. This results in trees that are created more quickly and with lower variance in the model at the cost of a slight increase in bias. These trees similarly “vote” as in the case of the random forests algorithm.

3.4.3 Boosting Algorithms. Boosting algorithms are a family of algorithms whose aim is to create a strong learner from a weak learner. They work by applying the weak learner sequentially to weighted versions of the data where in each sequential application misclassified data is given additional weight. The weak learner can be any classification or regression model, but the most frequently used learner is a decision tree [31]. However, they can be sensitive to outliers, take longer to train, and are harder to scale up, due the number of individual learners.

AdaBoost. Adaptive boosting [34], Adaboost for short, was the original boosting algorithm and has the characteristic that the decision trees have a single split, sometimes called decision stumps. This gives all data points equal weight and then weighs the incorrectly classified points more than the correctly classified ones. Then it will keep training models until the error increases.

Xtreme Gradient Boosting. XGBoost [35] is a more recent algorithm for boosting and combines decision trees with more splits and sophisticated algorithms to improve the time it takes for the algorithm to converge to the optimal tree. It is extremely popular due
### Metric Formula

<table>
<thead>
<tr>
<th>Metric (L2)</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean (L2)</td>
<td>( \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} )</td>
</tr>
<tr>
<td>Manhattan (L1)</td>
<td>( \sum_{i=1}^{k}</td>
</tr>
<tr>
<td>Minkowski (Lp)</td>
<td>( (\sum_{i=1}^{k}</td>
</tr>
<tr>
<td>Chebyshev</td>
<td>( \max_{i=1}</td>
</tr>
<tr>
<td>Hamming</td>
<td>( \sum_{i=1}^{k} d_i ) where ( d_i = \begin{cases} 0 &amp; \text{if } x_i = y_i \ 1 &amp; \text{else} \end{cases} )</td>
</tr>
</tbody>
</table>

Table 3.2: Machine Learning Metrics.

...to its ease in configuring, its relative speed in running and its high accuracy. The name comes from the method of updating the model. Unlike in Adaboost, the strong learner is optimized using gradient descent.

#### 3.4.4 K-Nearest Neighbors

Like decision trees, K-nearest neighbor is both a classification and regression algorithm. The premise is that it calculates distances between data points, finds the closest neighbors to a new data point, and predicts its label based on its neighbors’ labels. For each new point, the distances between it and all other points are calculated. One important distinction of KNN compared to other algorithms is that it does not learn a relationship function from the data; instead, it memorizes the training set. KNN is simple to implement and can do well enough in practice, but it has a high computational cost, a high storage cost, and there must be a meaning distance function between the data points. The choice of distance can have a large impact on the model’s results. Common distances used are shown in Table 3.2. Another difficulty is a lack of insight into which K to choose. Too small a value of K can lead to overfitting and noise. A large K will have less variance but more bias and be more computationally expensive.
3.5 Clustering

Clustering algorithms are similar to classification algorithms in that they attempt to classify datapoints. However, they are unsupervised, so instead of learning a function that relates the features to the labels, they learn patterns in the features and group the datapoints by these patterns. Applications of clustering include market research, recommender systems, document clustering, signal processing, missing data manipulation, and image compression.

3.5.1 K-means. K-means is an iterative algorithm that divides a dataset into clusters where each datapoint is in the cluster whose mean is closest to it. It requires the number of clusters to divide the data into, hence the $k$ in the name. After randomly initializing centroids, the centers of the clusters, the algorithm calculates the distance between each point and the centroids and assigns the data point to the closest cluster, known as the assignment step. In the update step, new centroids are computed by taking the average of the points in each cluster, and the process is repeated until the clusters are constant. K-means is a special case of a Gaussian Mixture Model in which the data is assumed to have a normal, Gaussian, mean. A GMM creates centroids based on parameters like mean and co-variance instead of distances. Instead of calculating distances as in K-means, it calculates the probability a data point belongs to each cluster, and then updates the clusters based on the parameters.

3.5.2 Non-negative Matrix Factorization. Non-negative matrix factorization (NMF) is clustering and dimensionality reducing algorithm that works by factoring a non-negative matrix into two non-negative matrices [36]. Given a non-negative $n \times m$ matrix $M$ and a rank $r$, find an $n \times r$ matrix $B$ and an $r \times m$ matrix $C$ such that

$$M \approx BC$$
where
\[ \min ||M - BC||_F \text{ subject to } B \geq 0, C \geq 0. \]

In most cases the rank \( r \) of the factorization is needed. However, for some problems and in some software, such as Python’s sklearn, the rank is optimized inside the algorithm.

If \( C \) is constrained to be orthogonal, \( CC^T = I \), then NMF is equivalent to K-means clustering [37]. If the orthogonality of \( C \) is not enforced, then in practice, the clustering property generally holds. The error function can be calculated using the Frobenius norm or the Kullback-Leibler divergence. In the latter case, NMF is identical to probabilistic latent semantic indexing, (PLSI) [37].

NMF can be performed in several ways, though currently there are no algorithms that guarantee to find a global minimum. Some recent methods have been developed that use non-negative least squares and gradient descent [38]. A popular method is the multiplicative update rule, which after initializing the \( B \) and \( C \), updates them in the following manner

\[
C_{ij}^{n+1} = C_{ij}^n \frac{((B^n)^T M)_{ij}}{((B^n)^T B^n C^n)_{ij}}
\]

\[
B_{ij}^{n+1} = C_{ij}^n \frac{(M(C^{n+1})^T)_{ij}}{(B^n C^{n+1} (C^{n+1})^T)_{ij}}
\]

NMF is often used for identifying roles in a network because we can define the roles after the factorization [14, 38], which we will see in Section 6.3 and Section 7.1.

3.6 Neural Networks

Neural networks are an area of machine learning that has exploded in popularity in recent years. They mimic a human brain by creating layers of nodes (neurons) that are connected to each other. If the output of a node at any layers is above a certain value, the node is activated, sending its data to the next layer. These layers identify hidden features in data that is used to make predictions. Neural networks are used in image recognition, voice
Table 3.3: Input matrix for convolution.

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>2</th>
<th>0</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.4: Kernel for convolution.

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.5: Output matrix of convolution.

<table>
<thead>
<tr>
<th>8</th>
<th>9</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

recognition, language translation, language prediction, recommender systems, and many other applications [29].

3.6.1 Convolution Neural Network. A convolution neural network (CNN) is a neural network that uses convolution operator

\[ [f * g](t) = \int_0^t f(x)g(t-x)dx. \]

In a CNN where the input is a matrix, also called a tensor since it may have more than two dimensions, the convolution takes the form of a kernel, a multidimensional array of parameters that are multiplied by the input. Each submatrix of the same size as the kernel is multiplied by the kernel using element-wise multiplication and then summed to get a single number. For example, consider the input matrix \( I \) shown in Table 3.5 and the \( 2 \times 2 \) kernel shown in Table 3.4. There are sixteen \( 2 \times 2 \) submatrices of \( I \) and each of these is multiplied by the kernel to get a \( 4 \times 4 \) matrix. So the top left value of the output is \( 2 \times 1 + 4 \times 1 + 1 \times 0 + 2 \times 1 = 8 \).

CNNs stack multiple convolution layers, along with other possible layers like pooling layers which shrink the size of inputs and activation layers, which determine the data to be sent to the next layer. The final layer in a CNN is a fully connected layer that returns the
output for each class; often this take the form of a probability vector. CNNs have been very successful in solving many machine learning problems such as image classification because they reduce the number of parameters used (through the convolutions and pooling layers) without reducing the quality of the output.

3.6.2 Graph Neural Networks. While CNNs are good for certain problems where the input can be represented as a tensor, they are not capable of handling graphs. We could use the adjacency matrix of a graph, however, these tend to be sparse and only include information about the connectivity of the graph. Graph neural networks (GNN) [39] were developed as an alternative to solving graph-related problems, including node classification, clustering, and link prediction [40]. Like CNNs, GNNs also use convolutions, linear layers, and activation functions. Instead of a matrix, a graph convolution uses polynomials that approximate the Laplacian as the kernel. The coefficients of this polynomial filter are then applied to vectors of node features which results in a combination of features from a node and its neighbors. GNNs have been successful and there is a lot of research being done in this area currently [40, 41, 42].

3.7 Time Series

Time series algorithms are specifically designed for forecasting and making predictions over time. They are often used to predict stock prices, revenue, and weather but can also be used to fill in missing values, a process called imputation. The data consists of sequences of values, where each point in the sequence represents a time period. When there is only one value at each time in the sequence, the time series is called univariate. A time series is multivariate when there is more than one value at a time. They are often applied to financial data to predict stock prices and revenue, and weather, but can also be used to fill in missing values, a process called imputation.
3.7.1 Hidden Markov Models. Hidden Markov models (HMM) are probabilistic models that represent sequences of observations [43]. Given such a sequence, it assumes that there is a some stochastic process consisting of hidden states that influence the observations. Further, it assumes that the stochastic process is a Markov chain of the $n$th order, meaning that the current state is only dependent on the $n$th previous states, called $n$th order HMM. Usually, HMMs are first order so that the next state only depends on the current state [44]. The probability of transitioning between each state is represented in the state transition matrix. Since the states are hidden, an emission probability matrix is used to relate the hidden states to the observed states. HMM can be used to compute the probability of the observations through dynamic programming or find the probability of the hidden state sequence based on the observed sequence with Viterbi’s algorithm [45].

3.7.2 ARIMA. ARIMA, which stands for auto-regressive integrated moving average, is a general class of forecasting algorithms for time-series [46, 47]. Unlike an HMM, it depends on previous time periods which are weighted unequally. It is a linear regression model, which requires that the independent variables are independent of each other. This means that the time-series must be stationary: has constant mean, constant variance, and constant covariance between periods of identical distance. If a time-series is not stationary, a new, stationary time-series can be created by differencing it [48].

3.7.3 Recurrent Neural Networks. Recurrent neural networks (RNN) are another time series algorithm used for financial forecasting, machine translation, and music composition [29]. They accept sequences as the input, where each point in the sequence represents a new time period. The sequence entries can be a single number or a matrix representing multiple characteristics. Similar to HMMs, they have a hidden state, which remembers information about the sequence. An RNN takes a neural network in which the layers are independent and links them by passing the hidden state weights and biases through each layer, making each output dependent on the previous output.
LSTMs, which stand for long short-term memory, are a variation on RNNs [49]. They address exploding and vanishing gradients by creating paths outside of the regular RNN that allow information to pass through to the RNN via gates. If the network decides some information is important, it can move that information onto a path until it decides it is no longer useful. For example, when having a machine generate text, a subject’s name may be used and then referred to with gendered pronouns. An RNN will forget the name once it is out of the sequence, but an LSTM can put the name on path until it is needed later by the network [29].
Link prediction is the process of predicting new connections in a graph. This can be either finding an unobserved edge or predicting which two nodes will most likely be connected next in a dynamic graph [50]. An example of the first case is an online social network that recommends people you may already know but are not connected to on the social network based on your connections in the social network. Another example of the first case is data duplication. Consider a large genealogy network where people may be represented more than once in this network. Can we identify which nodes are actually duplicates based on their connections? An example of predicting new connections in the future is an online e-commerce site that gives recommendations for things to buy based on what you have previously bought. In this case, you actually have a bipartite graph with the users as one set and the products as another. Adding roads to a road network to reduce congestion is another example.

There are many ways we make the predictions using network theory. Such predictions typically involves calculating a similarity measure between pairs of nodes and predicting the $k$ unconnected pairs with the maximum likelihood. Methods of identifying new edges can be split into the following categories with some common calculations [50, 51, 52]:

- local similarity methods
  - common neighbors
  - Jaccard’s coefficient
  - preferential attachment
  - resource allocation
  - Adamic-Adar

- distance (path) methods
  - Katz distance
- resistance distance
- rooted PageRank

- kernel-based similarity measures
  - commute time kernel
  - exponential diffusion kernel

- graph neural networks.

As mentioned, two methods of calculating similarity are the Katz distance and resistance distance. Recall from Section 2.6.1 that the Katz distance matrix is

$$K = (I - \alpha A)^{-1} - I$$

when \(\alpha < \frac{1}{\rho(A)}\), and effective distance between nodes \(i\) and \(j\) is

$$R_{ij} = L_{i,i}^+ + L_{j,j}^+ - 2L_{i,j}^+.$$ 

In the link prediction problem, the goal is to take the pairs with the maximal Katz values or the minimum effective resistance. Katz distance is affected by the decay parameter \(\alpha\) which means that changing \(\alpha\) may yield a different maximal pair ordering.

For a path graph, effective resistance is the distance between two nodes. Thus effective resistance will always predict that nodes that are two apart will be connected next. With Katz distance, we look at the maximum Katz values. We simplify the problem to consider nodes that are two and three apart. Given \(\alpha < \frac{1}{\rho(A)}\), we have a Katz ordering on the pairs of nodes that are two apart and another ordering for three apart. If the minimum Katz value for the nodes that are two apart is greater than the maximum Katz value for the nodes that are three apart, then using Katz distance will align with the effective resistance ordering for those pairs. However, if the minimum Katz value for the nodes that are two apart is less
than the maximum Katz value for the nodes that are three apart, then the Katz ordering will rank a pair of nodes that is three apart above a pair that is two apart, unlike effective resistance. We want to identify the value of $\alpha$ where this shift occurs. This is joint work being done with Emily Evans and Amanda Francis.

In this chapter, we will investigate this question by finding closed formulas for the Katz matrix on some simple graphs, specifically paths, forks, and cycles. These formulas involve binomial coefficients so we will first give a general definition for binomial coefficients and prove several identities needed for later proofs. Then we will prove the closed formula for the Katz matrix of a path graph and show that it converges. Next, we will prove a closed formula for the Katz matrix of a fork. Finally, we will prove a closed formula for the Katz matrix for a cycle, show that converges, and relate the Katz distance to the minimum effective resistance.

4.1 Binomial Background

The path, fork, and cycle graphs all have an equivalent closed form definition for the Katz matrix. In all three cases, the closed form uses binomial coefficients. This section consists of binomial identities and lemmas needed for the proofs in the following sections.

4.1.1 Binomial Identities.

**Definition 4.1.** For $r \in \mathbb{R}$ and $k \in \mathbb{Z}$, we define the *binomial coefficient* to be

$$
\binom{r}{k} = \begin{cases} 
\prod_{j=1}^{k} \frac{r+1-j}{j} & k \geq 0 \\
0 & k < 0.
\end{cases}
$$

Notice that this gives the common factorial definition when $r, k \in \mathbb{N}$:

$$
\binom{r}{k} = \frac{(r)!}{(k)!(r-k)!}.
$$
We observe that for $0 \leq k \leq r$, \( \binom{r}{k} \) achieves a maximum when $k = \lfloor \frac{r}{2} \rfloor$ and that $\binom{r}{k} < \binom{r}{k}$.

When $k \geq 0$,

\[
\binom{r}{k} = \prod_{j=1}^{k} \frac{r+1-j}{j} = \frac{r}{r-k} \prod_{j=1}^{k} \frac{r-j}{j} = \frac{r}{r-k} \binom{r-1}{k}
\]

so if $k \neq r$, we get the following identity [53],

\[
\binom{r}{k} = \frac{r}{r-k} \binom{r-1}{k}.
\] (4.1)

We also have the following identity from [53]:

\[
\binom{r}{k} = (-1)^k \binom{k-r-1}{k}.
\] (4.2)

Finally, the Convolution Identity [53] defined for $m \in \mathbb{Z}$ is

\[
\sum_{k \geq 0} \binom{n-tk}{k} \binom{r-t(m-k)}{m-k} \frac{n}{n-tk} = \binom{n+r-tm}{m}.
\] (4.3)

4.1.2 Binomial Lemmas.

**Lemma 4.2.** Let $l \in \mathbb{N}$, $m, j \in \mathbb{N}_0$. Then

\[
\sum_{k=0}^{m} \binom{l+2k}{k} \binom{j-l-1+2(m-k)}{m-k} \frac{1}{l+k+1} = \frac{1}{l+1} \binom{j+2m}{m}.
\]

**Proof.** First, observe that by Equation 4.1:

\[
\binom{l+1+2k}{k} \left( \frac{j-l-1+2(m-k)}{m-k} \right) = \frac{1}{l+k+1} \binom{l+2k}{k} \left( \frac{j+2m}{m} \right).
\]

Then multiplying by 1, substituting $\binom{l+1+2k}{k}$, and applying Equation 4.3 with $t = -2$, we see that
Lemma 4.3. For $l \in \mathbb{N}$ and $m, j \in \mathbb{N}_0$, 

$$
\sum_{k=0}^{m} \binom{l+2k}{k} \binom{j-l-1+2(m-k)}{m-k} \frac{1}{l+k+1}
$$

$$
= \frac{l+1}{l+1} \sum_{k=0}^{m} \binom{l+2k}{k} \binom{j-l-1+2(m-k)}{m-k} \frac{l+2k+1}{(l+2k+1)(l+k+1)}
$$

$$
= \frac{1}{l+1} \sum_{k=0}^{m} \binom{l+2k+1}{k} \binom{j-l-1+2(m-k)}{m-k} \frac{l+1}{l+2k+1}
$$

$$
= \frac{1}{l+1} \binom{j+2m}{m}.
$$

Proof. An application of partial fractions shows that 

$$
\frac{1}{(l+2k)(l+k+1)} = \frac{2}{(l+2)(l+2k)} - \frac{1}{(l+2)(l+k+1)}.
$$

Applying this to the left hand side of the equation, splitting the sum into two, and applying Equation 4.3 to the first sum with $t = -2$ and Lemma 4.2 to the second yields
\[
\sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{l^2 + l}{(l + 2k)(l + k + 1)} 
\]
\[
= \frac{l^2 + l}{l + 2} \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \left( \frac{2}{l + 2k} - \frac{1}{l + k + 1} \right) 
\]
\[
- \frac{l^2 + l}{l + 2} \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{1}{l + k + 1} 
\]
\[
= \frac{l^2 + l}{l + 2} \left[ \frac{2}{l} \binom{j + 2m - 1}{m} - \frac{1}{l + 1} \binom{j + 2m}{m} \right]. 
\]

Now if \( j = m = 0 \),

\[
\frac{l^2 + l}{l + 2} \left[ \frac{2}{l} \binom{j + 2m - 1}{m} - \frac{1}{l + 1} \binom{j + 2m}{m} \right] 
\]
\[
= \frac{l^2 + l}{l + 2} \left[ \frac{2}{l} (-1) - \frac{1}{l + 1} (0) \right] 
\]
\[
= \frac{l^2 + l}{l + 2} \left[ \frac{2}{l} - \frac{1}{l + 1} \right] 
\]
\[
= \frac{l^2 + l}{l + 2} \frac{2l + 2 - l}{l^2 + l} 
\]
\[
= 1. 
\]
If not, then using Equation 4.1 and simplifying, we get

\[
\frac{l^2 + l}{l + 2} \left[ \frac{2}{l} \left( \begin{array}{c} j + 2m - 1 \\ m \end{array} \right) - \frac{1}{l + 1} \left( \begin{array}{c} j + 2m \\ m \end{array} \right) \right]
= \frac{l^2 + l (j + 2m - 1)}{l + 2} \left[ \frac{2}{l} \frac{j + 2m}{(j + m)(l + 1)} \right]
= \frac{l^2 + l (j + 2m - 1)}{l + 2} \left[ \frac{2(j + m)(l + 1) - l(j + 2m)}{(l^2 + l)(j + m)} \right]
= \left( \begin{array}{c} j + 2m - 1 \\ m \end{array} \right) \frac{2jl + 2j + 2m + 2ml - jl - 2ml}{(l + 2)(m + j)}
= \left( \begin{array}{c} j + 2m - 1 \\ m \end{array} \right) \frac{jl + 2j + 2m}{(l + 2)(j + m)},
\]

concluding the proof.

\[\square\]

**Lemma 4.4.** Let \( l \in \mathbb{N} \) and \( j, m \in \mathbb{N}_0 \). Then

\[
\sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{2k}{(l + 2k)(l + k + 1)} = \begin{cases} 
0 & m = j = 0 \\
\frac{2m}{(l+2)(j+m)} \binom{j+2m-1}{m} & \text{otherwise.}
\end{cases}
\]

**Proof.** A partial fractions decomposition shows that

\[
\frac{k}{(l + 2k)(l + k + 1)} = \frac{l + 1}{(l + 2)(l + k + 1)} - \frac{l}{(l + 2)(l + 2k)}.
\]
Using the same method as the proof for Lemma 4.3,

\[
\sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{2k}{(l + 2k)(l + k + 1)}
\]

\[
= \frac{2}{l + 2} \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \left( \frac{l + 1}{l + k + 1} - \frac{l}{l + 2k} \right)
\]

\[
= \frac{2(l + 1)}{l + 2} \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{1}{l + k + 1}
\]

\[
- \frac{2}{l + 2} \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{l}{l + 2k}
\]

\[
= \frac{2}{l + 2} \left[ \binom{j + 2m}{m} \frac{l + 1}{l + 1} - \binom{j + 2m - 1}{m} \right]
\]

\[
= \frac{2}{l + 2} \left[ \binom{j + 2m}{m} - \binom{j + 2m - 1}{m} \right].
\]

Now if \( m = j = 0 \),

\[
\frac{2}{l + 2} \left[ \binom{j + 2m}{m} - \binom{j + 2m - 1}{m} \right] = \frac{2}{l + 2} [0] = 0.
\]

Otherwise,

\[
\frac{2}{l + 2} \left[ \binom{j + 2m}{m} - \binom{j + 2m - 1}{m} \right]
\]

\[
= \frac{2}{l + 2} \left[ \binom{j + 2m - 1}{m} \frac{j + 2m}{(j + m)} - \binom{j + 2m - 1}{m} \right]
\]

\[
= \frac{2}{l + 2} \left[ \frac{j + 2m}{j + m} \right] \binom{j + 2m - 1}{m}
\]

\[
= \frac{2m}{(l + 2)(j + m)} \binom{j + 2m - 1}{m}.
\]
Lemma 4.5. For \( l \in \mathbb{N} \) and \( m, j \in \mathbb{N}_0 \),

\[
\sum_{k=0}^{m} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{(l^2 + l - 2k)}{(l + 2k)(l + k + 1)} = \left\{ \begin{array}{ll} 1 & m = j = 0 \\ \frac{jl + 2j}{(l+2)(j+m)} \binom{j+2m-1}{m} & \text{otherwise.} \end{array} \right.
\]

Proof. Noting that

\[
\sum_{k=0}^{m} \binom{2(m - k) - l - 1}{m - k} \frac{(l^2 + l - 2k)}{(l + 2k)(l + k + 1)} \binom{l + 2k}{k} = \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{l^2 + l}{(l + 2k)(l + k + 1)} - \sum_{k=0}^{m} \binom{l + 2k}{k} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{2k}{(l + 2k)(l + k + 1)},
\]

we can apply Lemma 4.3 and Lemma 4.4. When \( m = j = 0 \), this is equal to \( 1 - 0 \) and if not,

\[
\frac{jl + 2j + 2m}{(l + 2)(j + m)} \binom{j + 2m - 1}{m} - \frac{2m}{(l + 2)(j + m)} \binom{j + 2m - 1}{m} = \frac{jl + 2j}{(l + 2)(j + m)} \binom{j + 2m - 1}{m}.
\]

\[\Box\]

Finally, we look at the following conjecture.

Conjecture 4.6. For \( j, l \in \mathbb{N}, j < l, \alpha < \frac{1}{2} \) and \( s = \lfloor \frac{l}{2} \rfloor \), define

\[
R(j, l, \alpha) = \sum_{m=s+1}^{2s} \alpha^{2m} \sum_{k=m-s}^{s} \binom{2(m - k) + j - l - 1}{m - k} \frac{(l^2 + l - 2k)}{(l + 2k)(l + k + 1)} \binom{l + 2k}{k}.
\]

Then

\[
\lim_{n \to \infty} R(j, l, \alpha) = 0.
\]

Computationally, \( R \) appears to get very small, converging to 0 as \( l \) gets bigger, as seen in Figure 4.1.
Figure 4.1: Various plots of $R(j, l, \alpha)$ for different values of $j$ and $\alpha$. In every case, $R$ is decreasing to 0.
4.2 A Closed Formula for the Katz Matrix for a Path Graph

Recall that a path graph, shown in Figure 4.2, is a graph that has one path through all of its nodes and can be drawn in a straight line. The adjacency matrix for a path is of the form

\[
A = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
1 & 0 & 1 & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & 1 & 0 & 1 \\
0 & \cdots & 0 & 1 & 0 \\
\end{bmatrix}
\]

\(A\) is a tridiagonal Toeplitz matrix [54, 55] so its eigenvalues are

\[2 \cos \left( \frac{k\pi}{n+1} \right) \text{ for } k = 1, ..., n.\]

Thus the spectral radius of \(A\) is less than 2. Because \(A\) is tridiagonal and therefore \(I - \alpha A\) is tridiagonal, we can use Theorem 4.7 to find \(\det(I - \alpha A)\), which can then be used to find the closed form for the Katz matrix.

\[\text{Theorem 4.7. [56, Equation 5]} \text{ For a path graph } G \text{ consisting of } n \text{ nodes with an adjacency matrix } A, s = \left\lfloor \frac{n}{2} \right\rfloor, \text{ and } \alpha < \frac{1}{2}, \]

\[\det(I - \alpha A) = d_n(\alpha) = \sum_{m=0}^{s} (-1)^m \binom{n-m}{m} \alpha^{2m}.\]

Figure 4.2: A path graph with 6 nodes.
Theorem 4.8. If $G$ is a path graph on $n$ vertices, then for $\alpha < \frac{1}{2}$ and

$$K_{i,j}^n(\alpha) = \frac{\alpha^{j-i}d_{i-1}(\alpha)\alpha^{j-n-j}(\alpha)}{d_n(\alpha)} - 1_{ij},$$

where

$$d_i(\alpha) = \sum_{m=0}^{s} (-1)^m \binom{l-m}{m} \alpha^{2m}$$

for $s = \lfloor \frac{l}{2} \rfloor$ and $1_{ij} = 0$ when $i \neq j$ and $1_{ij} = 1$ otherwise.

Proof. Recall from Section 2.6.1 that a formula for the Katz matrix is

$$K = (I - \alpha A)^{-1} - I.$$

Let $MP_n = (I - \alpha A)$ for a path graph with $n$ nodes. We can find the entries of $MP_n^{-1}$ using Cramer's rule:

$$MP_{nij}^{-1} = (-1)^{i+j} \frac{\det(MP_n(i,j))}{\det(MP_n)},$$

where $MP_n(i,j)$ is the matrix obtained from $MP_n$ by deleting the $i$th row and the $j$th column. We also make use of the formula for block matrix determinants which states that assuming block $E$ is invertible,

$$\det \begin{bmatrix} B & C \\ D & E \end{bmatrix} = \det(B - CE^{-1}D) \det(E).$$

We now consider the determinant of $MP_n(i,j)$ and without loss of generality assume that $i \leq j$ since $MP_n$ is symmetric. We note that $MP_n(i,j)$ is $(n - 1) \times (n - 1)$, and let $B$ be the $(j - 1) \times (j - 1)$ upper left block, $C$ be the $(j - 1) \times (n - j)$ upper right block, $D$ be the $(n - j) \times (j - 1)$ lower left block, and $E$ be the $(n - j) \times (n - j)$ lower right block. Observe that $E$ has 1 on the diagonal and $-\alpha$ on the super/sub diagonal so it is $MP_{n-j}$, and hence by Theorem 4.7, the determinant is non-zero and therefore invertible with determinant equal
to \(d_{n-j}(\alpha)\). Moreover we observe that the matrix \(D\) is identically zero so

\[
\det(MP_n(i, j)) = \det(B - CE^{-1}D) \det(E) = \det(B) \det(E) = \det(B)d_{n-j}(\alpha).
\]

We now consider the determinant of \(B\). We again use the block matrix determinant formula this time setting \(B'\) to be the \((i-1) \times (i-1)\) upper left block, \(C'\) be the \((i-1) \times (j-i)\) upper right block, \(D'\) be the \((j-i) \times (i-1)\) lower left block, and \(E'\) be the \((j-i) \times (j-i)\) lower right block of the matrix \(B\). We note that \(E'\) is upper triangular with \(-\alpha\) on the diagonal and hence is invertible with determinant equal to \((-\alpha)^{j-i}\). Similar to before, \(D'\) is identically equal to zero hence

\[
\det(B) = \det(B' - C'E'^{-1}D') \det(E') = \det(B') \det(E').
\]

Since \(B'\) has 1 on the diagonal and \(-\alpha\) on the super/sub diagonal, by Theorem 4.7 it has determinant \(d_{i-1}(\alpha)\). Putting everything together gives the desired result of

\[
MP_n(i, j) = (-\alpha)^{j-i}d_{i-1}(\alpha)d_{n-j}(\alpha).
\]

The formula for \(MP_n^{-1}\) follows by plugging the values into Cramer’s rule.

As a result of Theorem 4.7, we prove the following two lemmas using the same terminology.

**Lemma 4.9.** Assuming Conjecture 4.6, \(d_l(\alpha)\) has a multiplicative inverse that can be approximated with the pseudo-inverse where \(s = \lfloor \frac{l}{2} \rfloor\),

\[
d_l^\dagger(\alpha) = \sum_{m=0}^{s} \frac{(l^2 + l - 2m)}{(l + 2m)(l + m + 1)} \left( \frac{l + 2m}{m} \right) \alpha^{2m}.
\]

**Proof.** By definition of the binomial coefficients and \(\alpha\), \(d_l(\alpha)\) is nonzero and real and thus must have a multiplicative inverse. We will multiply \(d_l(\alpha)\) by \(d_l^\dagger(\alpha)\) and show that the
product is approximately 1. Using Equation 4.2 to replace \((-1)^m \binom{l-m}{m}\) with \(\binom{2m-l-1}{m}\), gives

\[
d_l(\alpha) d_{l}^*(\alpha) = \left( \sum_{m=0}^{s} (-1)^m \binom{l-m}{m} \alpha^{2m} \right) \left( \sum_{m=0}^{s} \frac{(l^2 + l - 2m)}{(l + 2m)(l + m + 1)} \binom{l-2m}{m} \alpha^{2m} \right)
\]
\[
= \left( \sum_{m=0}^{s} \binom{2m-l-1}{m} \alpha^{2m} \right) \left( \sum_{m=0}^{s} \frac{(l^2 + l - 2m)}{(l + 2m)(l + m + 1)} \binom{l-2m}{m} \alpha^{2m} \right)
\]
\[
= \sum_{m=0}^{s} \alpha^{2m} \sum_{m=0}^{s} \frac{(2m-k-l-1)}{m-k} \frac{(l^2 + l-2k)}{(l+2k)(l+k+1)} \binom{l-2m}{m} \alpha^{2m}
\]
\[
= \sum_{m=s+1}^{2s} \alpha^{2m} \sum_{m=0}^{s} \frac{(2m-k-l-1)}{m-k} \frac{(l^2 + l-2k)}{(l+2k)(l+k+1)} \binom{l-2m}{m} \alpha^{2m}
\]
\[
= 1 + \sum_{m=1}^{s} (\alpha^{2m} \times 0) + R(j,l,\alpha)
\]
\[
\approx 1,
\]

where the second to last line comes from using Lemma 4.5 with \(j = 0\) to get

\[
\frac{j l + 2j}{(l + 2)(j + m)} \binom{j + 2m - 1}{m} = \frac{0}{m(l + 2)} \binom{2m - 1}{m}
\]

and from Conjecture 4.6.

\[
\square
\]

**Lemma 4.10.** Let \(l \in \mathbb{N}, j \in \mathbb{N}_0, j \leq l, s = \lfloor \frac{l}{2} \rfloor\), and \(\alpha < \frac{1}{2}\). Define \(d_{i,j}^*(\alpha) = \frac{d_{i-j}(\alpha)}{d_i(\alpha)}\). Then assuming Conjecture 4.6,

\[
d_{i,j}^*(\alpha) \approx \sum_{m=0}^{2s} \frac{j(j + 2m - 1)!}{m!(j + m)!} \alpha^{2m}.
\]
Proof. From Lemma 4.9, \( d_t \) has an inverse so \( d_{i,j}^t(\alpha) \) is defined and can be approximated using the pseudoinverse.

Then using the same method as in Theorem 4.9

\[
d_{l-j}(\alpha)d_l^1(\alpha) = \sum_{m=0}^{s} (-1)^k \binom{l-j-m}{m} \alpha^{2m} \times \left( \sum_{m=0}^{\lfloor l/2 \rfloor} \frac{(l^2 + l - 2m)}{(l + 2m)(l + m + 1)} \binom{l + 2m}{m} \alpha^{2m} \right)
\]

\[
= \sum_{m=0}^{s} \binom{j - l - 1 + 2m}{m} \alpha^{2m} \times \left( \sum_{m=0}^{\lfloor l/2 \rfloor} \frac{(l^2 + l - 2m)}{(l + 2m)(l + m + 1)} \binom{l + 2m}{m} \alpha^{2m} \right)
\]

\[
= \sum_{m=0}^{s} \alpha^{2m} \sum_{k=0}^{m} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{(l^2 + l - 2k)}{(l + 2k)(l + k + 1)} \binom{l + 2k}{k} \alpha^{2m}
\]

\[
+ \sum_{m=s+1}^{2s} \alpha^{2m} \sum_{k=m-s}^{s} \binom{j - l - 1 + 2(m - k)}{m - k} \frac{(l^2 + l - 2k)}{(l + 2k)(l + k + 1)} \binom{l + 2k}{k} \alpha^{2m}
\]

\[
= \sum_{m=0}^{s} \left[ \alpha^{2m} \binom{j + 2m - 1}{m} \frac{jl + 2j}{l + 2(j + m)} \right] + R(j, l, \alpha)
\]

\[
= \sum_{m=0}^{s} \left[ \alpha^{2m} \binom{j + 2m - 1}{m} \frac{j}{j + m} \right] + R(j, l, \alpha)
\]

\[
=R(j, l, \alpha) + \sum_{m=0}^{s} \alpha^{2m} \frac{j(j + 2m - 1)!}{(j + m)!m!}.
\]

By Conjecture 4.6, \( R(j, l, \alpha) \) will be very small when \( l \) is large, completing the proof.

\[
\square
\]

Lemma 4.11. As \( n \to \infty \),

\[
\sum_{m=0}^{2s} \frac{j(j + 2m - 1)!}{m!(j + m)!} \alpha^{2m}
\]

converges for fixed \( j \), \( s = \lfloor \frac{n}{2} \rfloor \) and \( \alpha < \frac{1}{2} \).
Proof. Notice that as \( n \to \infty \), \( m \to \infty \) so by the ratio test,

\[
\lim_{n \to \infty} \left| \frac{j(j + 2(m + 1) - 1)!\alpha^{2m+2}}{(m + 1)!((j + (m + 1))!)} \cdot m!(j + m)! \right| \frac{m!(j + m)!}{j(j + 2m - 1)!\alpha^{2m}}
\]

\[
= \lim_{n \to \infty} \left| \frac{(j + 2m + 1)!((j + m + 1))!}{(m + 1)!((j + m + 1))!} \right| \frac{(j + m)!\alpha^2}{(j + 2m - 1)!\alpha^{2m}}
\]

\[
= \lim_{n \to \infty} \left| \frac{(j + 2m + 1)(j + 2m + 1)!}{(m + 1)!((j + m + 1))!} \right| \frac{(2m + j)(2m + j + 1)!}{(m + 1)!((m + j + 1))!}
\]

\[
< \lim_{n \to \infty} \frac{(2m + j)(2m + j + 1)}{4(m + 1)(m + j + 1)} \frac{4m^2 + 4mj + 2m + j^2 + j}{4m^2 + 4mj + 8m + 4j + 4}
\]

\[
= \lim_{n \to \infty} \frac{4m^2 + 4mj + 2m + j^2 + j}{4m^2 + 4mj + 8m + 4j + 4}
\]

\[
= \lim_{m \to \infty} \frac{4m^2 + 4mj + 2m + j^2 + j}{4m^2 + 4mj + 8m + 4j + 4} = 1.
\]

With those results, we can now prove that entries in the Katz matrix converge as the path graph \( G \) increases in nodes.

Theorem 4.12. Assuming Conjecture 4.6 for fixed \( \alpha \), \( j \), and \( i \), \( K^n_{ij}(\alpha) \) converges as \( n \) goes to \( \infty \).

Proof. Since

\[
K^n_{ij}(\alpha) = \alpha^{j-i}d_{i-1}(\alpha)d^*_n(\alpha) - 1_{ij},
\]

\( \alpha^{j-i} \), \( d_{i-1}(\alpha) \) and \( 1_{ij} \) are constant in \( n \), and \( d^*_n(\alpha) \) converges by Lemma 4.11, \( K^n_{ij} \) converges.

The motivation for looking at a path was to compare the ordering of predicted links from the Katz matrix with the effective resistance. On an unweighted path, effective resistance is simply the distance between two nodes. So effective resistance is minimized for nodes that are two apart, and these will be the pairs that are recommended first in a link prediction.
scenario. We want to know what $\alpha$ gives the same Katz ordering of recommending pairs of nodes that are two apart before recommending pair nodes that are separated by three edges. To do this, we look at the $\min K_{j-i=2}^n - \max K_{j-i=3}^n$. When this is positive, the Katz matrix will have the same ordering of pairs that are two and three apart as effective resistance. Computations indicate that the point occurs when $\alpha = \frac{1}{\sqrt{5}}$.

4.3 A Closed Formula for the Katz Matrix on a Fork Graph

A slight modification to a path graph is a fork, shown in Figure 4.3. It consists of one path through $n - 2$ nodes with the last two nodes connected to the end of the path. In this section, we will assume $n > 4$. Due to its similarity to the path graph, we can get a closed formula for the Katz matrix. We note that the adjacency matrix for a fork is of the following form:

$$A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & \cdots & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & \cdots & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}.$$

Lemma 4.13. Given the adjacency matrix $A$ for a fork graph with $n > 4$ nodes and $0 < \alpha < \frac{1}{\rho(A)}$,

$$df_n(\alpha) = \det(I - \alpha A) = (1 - 2\alpha^2) \left[ \left(1 - \frac{\alpha^2}{1 - 2\alpha^2}\right)d_{n-4}(\alpha) - \alpha^2 d_{n-5}(\alpha) \right]$$

where $d_n(\alpha)$ is defined in Theorem 4.7.
Figure 4.3: A fork with 7 nodes that splits on node 5.

**Proof.** Let $G$ be a fork graph with $n > 4$ nodes and adjacency matrix $A$, and let $\alpha < \frac{1}{\rho(A)}$. Define $MF_n = I - \alpha A$ so

$$MF_n = \begin{bmatrix}
1 & -\alpha & 0 & 0 & 0 & 0 & \cdots & 0 \\
-\alpha & 1 & -\alpha & 0 & 0 & 0 & \cdots & 0 \\
0 & -\alpha & 1 & -\alpha & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & -\alpha & 1 & -\alpha & 0 & 0 \\
0 & \cdots & 0 & 0 & -\alpha & 1 & -\alpha & -\alpha \\
0 & \cdots & 0 & 0 & 0 & -\alpha & 1 & 0 \\
0 & \cdots & 0 & 0 & 0 & -\alpha & 0 & 1 \\
\end{bmatrix}$$

We can split $MF_n$ into blocks and use block matrix determinants. If block $E$ is invertible, then

$$\det(MF) = \det \begin{bmatrix} B & C \\ D & E \end{bmatrix} = \det(B - CE^{-1}D) \det(E).$$

Let $B$ be the $(n-3) \times (n-3)$ upper left block, $C$ be the $(n-3) \times 3$ upper right block, $D$ be the $3 \times (n-3)$ lower left block, and $E$ be the lower right $3 \times 3$ block. Since $E$ is a $3 \times 3$ matrix, we can calculate the determinant using the cofactor expansion and the formula for
determinant of a $2 \times 2$ matrix:
\[
\det(E) = 1 \times \det \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - (-\alpha) \det \begin{bmatrix} \alpha & 0 \\ -\alpha & 1 \end{bmatrix} + (-\alpha) \det \begin{bmatrix} -\alpha & 1 \\ -\alpha & 0 \end{bmatrix} = 1 + \alpha(-\alpha) - \alpha \alpha = 1 - 2\alpha^2.
\]
Thus $E$ is invertible, and
\[
E^{-1} = \frac{1}{1 - 2\alpha^2} \begin{bmatrix} 1 & \alpha & \alpha \\ \alpha & 1 - \alpha^2 & \alpha^2 \\ \alpha & \alpha^2 & 1 - \alpha^2 \end{bmatrix}.
\]
Since $C$ and $D$ each only have one nonzero entry, we can calculate $CE^{-1}D$ directly.
\[
CE^{-1}D = \frac{1}{1 - 2\alpha^2} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ -\alpha & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & \alpha & \alpha \\ \alpha & 1 - \alpha^2 & \alpha^2 \\ \alpha & \alpha^2 & 1 - \alpha^2 \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & -\alpha \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}
\]
\[
= \frac{1}{1 - 2\alpha^2} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ -\alpha & -\alpha^2 & -\alpha^2 \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & -\alpha \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}
\]
\[
= \frac{1}{1 - 2\alpha^2} \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \alpha^2 \end{bmatrix}
\]
Let $F = B - CE^{-1}D$. Since $B = MP_{n-3}$, $F$ is an $(n - 3) \times (n - 3)$ matrix:
\[ F = \begin{bmatrix}
1 & -\alpha & 0 & 0 & 0 & \cdots & 0 \\
-\alpha & 1 & -\alpha & 0 & 0 & \cdots & 0 \\
0 & -\alpha & 1 & -\alpha & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & -\alpha & 1 & -\alpha & 0 \\
0 & \cdots & 0 & 0 & -\alpha & 1 & -\alpha \\
0 & \cdots & 0 & 0 & 0 & -\alpha & 1 - \frac{\alpha^2}{1 - 2\alpha^2}
\end{bmatrix}. \]

Let \( F_{m,m} \) be the \((m \times m)\) upper left submatrix of \( F \) and \( f_m \) be its determinant. \( f_m \) can be calculated using the tridiagonal recurrence formula [57]:

\[
\begin{align*}
    f_{-1} &= 0 \\
    f_0 &= 1 \\
    f_m &= f_{m-1} - \alpha^2 f_{m-2} \\
    f_{n-3} &= (1 - \frac{\alpha^2}{1 - 2\alpha^2}) f_{n-4} - \alpha^2 f_{n-5}
\end{align*}
\]

For all \( m < n - 3 \), \( F_{m,m} \) is of the form \( MP_m \) so \( f_{n-3} = (1 - \frac{\alpha^2}{1 - 2\alpha^2}) d_{n-4} - \alpha^2 d_{n-5} \).

Multiplying by \( \det(E) \), we get the desired result:

\[
\det(MF) = \det(F) \det(E) = \left[ (1 - \frac{\alpha^2}{1 - 2\alpha^2}) d_{n-4} - \alpha^2 d_{n-5} \right] (1 - 2\alpha). 
\]

\(\square\)

**Theorem 4.14.** If \( G \) is a fork graph with \( n > 4 \) nodes, then for \( i, j \in \mathbb{N} \), with \( i, j \leq n \), and \( 0 < \alpha < \frac{1}{\rho(A)} = \frac{1}{2} \), the Katz matrix can be defined by
\[
K^n_{i,j}(\alpha) = \begin{cases} 
\frac{d_i(\alpha)}{df_n(\alpha)} - 1 & i = j = n - 1 \\
\frac{\alpha^2 d_{i-2}(\alpha)}{df_n(\alpha)} & i + 1 = j = n \\
\frac{(-\alpha)^{j-i-1} d_{i-1}(\alpha)}{df_n(\alpha)} & i < j = n \\
\frac{(-\alpha)^{j-i-1} d_{i-1}(\alpha) df_{n-j}(\alpha)}{df_n(\alpha)} - 1_{ij}, & \text{otherwise}
\end{cases}
\]

where

\[
df_i(\alpha) = \begin{cases} 
(1 - 2\alpha^2) \left( 1 - \frac{\alpha^2}{1 - 2\alpha^2} \right) d_{i-4}(\alpha) - \alpha^2 d_{i-5}(\alpha) & n - j > 3 \\
1 & \text{otherwise}
\end{cases}
\]

and \(1_{ij} = 1\) when \(i = j\) and 0 otherwise.

Proof. Let \(G\) be a fork with \(n > 4\) nodes and adjacency matrix \(A\) and let \(0 \leq \alpha < \frac{1}{\rho(A)}\). Let \(MF_n = (I - \alpha A)\) and \(MF_n(i, j)\) be \(MF_n\) with the \(i\)th row and \(j\)th column removed. Using Cramer’s rule, we can find \(MF_{n-1}^{-1}\):

\[
MF_{n-1}^{-1} = (-1)^{i+j} \frac{\det(MF_n(i, j))}{\det(MF_n)}.
\]

Since \(MF_n\) is invertible and has determinant \(df_n(\alpha)\) by Lemma 4.13, the denominator in all cases will be \(df_n(\alpha)\). Consider the \((n - 1) \times (n - 1)\) matrix \(MF_n(i, j)\) and without loss of generality, assume \(0 \leq i \leq j \leq n\) since \(MF_n\) is symmetric.

We will again use block matrix determinants to calculate the \(\det(MF_n(i, j))\),

Case 1: \(0 \leq i \leq j < n - 2\).

We split \(MF_n(i, j)\) into blocks by letting \(B\) be the \((j - 1) \times (j - 1)\) upper left matrix, \(C\) the \((j - 1) \times (n - j)\) upper right matrix, \(D\) the \((n - j) \times (j - 1)\) lower left matrix, and \(E\) the \((n - j) \times (n - j)\) lower right matrix. \(E = MF_{n-j}\), so by Theorem 4.13, it is invertible with determinant \(df_{n-j}(\alpha)\). Also, notice that \(D\) is identically zero, so \(\det(M(i, j)) = \det(B - CE^{-1}D)\det(E) = \det(B)\det(E)\). We further split \(B\) into block matrices where \(B'\) is the \((i - 1) \times (i - 1)\) upper left block, \(C'\) is the \((i - 1) \times (j - i)\) upper right block, \(D'\) is the \((j - i) \times (i - 1)\) lower left block, and \(E'\) is the \((j - i) \times (j - i)\) lower right block of the matrix
A. We note that $E'$ is upper triangular with $-\alpha$ on the diagonal, and hence is invertible with determinant equal to $(-\alpha)^{j-i}$. As before, $D'$ is identically zero so $\det(B) = \det(B') \det(E')$. Since $B'$ has 1 on the diagonal and $-\alpha$ on the super/sub diagonal, $B' = MP_{i-1}$. Thus by Equation 4.7, $\det(B') = d_{i-1}(\alpha)$. Putting everything together,

$$\det(MF_n(i, j)) = \det(B') \det(E') \det(E) = d_{i-1}(\alpha)(-\alpha)^{j-i} df_{n-j}(\alpha).$$

Case 2: $n - 2 = j$.

Splitting $M$ as above,

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$ 

If $i < j$, then splitting $B$ again is Case, $B'$, $C'$, and $D'$ are the same as in Case 1 so

$$\det(MF_n(i, j)) = \det(B') \det(E') \det(E) = d_{i-1}(\alpha)(-\alpha)^{j-i} = d_{i-1}(\alpha)(-\alpha)^{j-i} df_{n-j}(\alpha).$$

If $i = j$, then $B = MP_{n-3} = MP_{i-1}$ so we still get that

$$\det(MF_n(i, j)) = \det(B) \det(E) = d_{i-1}(\alpha) = d_{i-1}(\alpha)(-\alpha)^{j-i} df_{n-j}(\alpha).$$

Case 3: $j = n - 1$.

Let $i < j$. In this case, we still split $M$ into block matrices, but of different sizes. Let $B$ be the upper left $(n - 3) \times (n - 3)$ block matrix, $C$ be the upper right $(n - 3) \times 2$ block matrix, $D$ be the identically zero lower left $2 \times (n - 3)$ block matrix and $E$ be the lower right block matrix,

$$E = \begin{bmatrix} -\alpha & 0 \\ -\alpha & 1 \end{bmatrix}.$$ 

Splitting $B$ again as before, $B' = MP_{i-1}$, $D'$ is identically zero, and $E'$ is a $(j-i-1) \times (j-i-1)$...
upper diagonal matrix with $-\alpha$ along the diagonal. Then

$$\det(MF_n(i, j)) = d_{i-1}(\alpha)(-\alpha)^{j-i-1}(-\alpha) = d_{i-1}(\alpha)(-\alpha)^{j-i} = d_{i-1}(\alpha)(-\alpha)^{j-i}f_1(\alpha).$$

If $i = j$, then $MF_n(i, j) = MP_{n-1}$ so

$$\det(MF_n(i, j)) = d_{n-1}(\alpha) = d_i(\alpha).$$

Case 4: $j = n$.

Assume $i < j - 1$. Then following the same pattern as in Case 3, where $E$ is the lower right $2 \times 2$ matrix,

$$E = \begin{bmatrix} -\alpha & 1 \\ -\alpha & 0 \end{bmatrix},$$

$$\det(MF_n(i, j)) = \det(B') \det(E') \det(E) = d_{i-1}(\alpha)(-\alpha)^{j-i-2}(\alpha) = -d_{i-1}(\alpha)(-\alpha)^{j-i-1}.$$

Let $i = j - 1$. Here, $MF_n(i, j) = MF_n(n - 1, n) = MP_{n-1}$ except the $(n - 1)$th entry is 0. We can use the recurrence formula for a tridiagonal matrix to calculate the determinate.

$$\det(MF_n(i, j)) = (-\alpha^2)d_{n-3}(\alpha) = \alpha^2d_{i-2}(\alpha).$$

If $i = j$, $MF_n(i, j) = MF_n(n, n) = MP_{n-1}$

$$\det(MF_n(i, j)) = d_{i-1}(\alpha) = d_{i-1}(\alpha)(-\alpha)^{j-i}f_{n-j}(\alpha).$$

Multiplying each case by $\frac{(-1)^{j-i}}{\det(M)}$ gives the result.

4.4 A Closed Formula for the Katz Matrix on a Cycle Graph

Another simple graph is the cycle, a closed sequence of edges that are connected so from every node, there is a path that goes through every other node exactly one time and ends
on the starting node. See Figure 4.4 for an example. In this section, we will assume that \( n > 4 \). The adjacency matrix for a cycle is

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & \cdots & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & \cdots & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & \cdots & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

Figure 4.4: A cycle graph with 8 nodes.

**Theorem 4.15.** For a cycle \( H \) with \( n > 4 \) vertices with adjacency matrix \( A \) and \( 0 < \alpha < \frac{1}{\rho(A)} \),

\[
det(I - \alpha A) = d_{n-1}(\alpha) - 2\alpha^n - 2\alpha^2 d_{n-2}(\alpha).
\]

**Proof.** Let \( MC_n = (I - \alpha A) \). For this graph, we will use cofactor expansion along the first column to calculate \( det(MC_n) \) and define \( MC_n(i, j) \) to be \( MC_n \) with the \( i \)th row and \( j \)th column removed. Notice that
\[ MC_n = \begin{bmatrix}
1 & -\alpha & 0 & 0 & \cdots & 0 & -\alpha \\
-\alpha & 1 & -\alpha & 0 & \cdots & 0 & 0 \\
0 & -\alpha & 1 & -\alpha & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & -\alpha & 1 & -\alpha & 0 & 0 \\
0 & 0 & \cdots & 0 & -\alpha & 1 & -\alpha & 0 \\
0 & 0 & \cdots & 0 & 0 & -\alpha & 1 & -\alpha \\
-\alpha & 0 & \cdots & 0 & 0 & 0 & -\alpha & 1 
\end{bmatrix} \]

Then

\[
\det(MC_n) = \sum_{i,j=1}^{n} MC_{n_{ij}}(-1)^{i+j}MC_n(i,j) = MC_n(1,1) + \alpha MC_n(2,1) + (-1)^{n+1}(-\alpha)MC_n(n,1).
\]

Observe that \( MC_n(1,1) = MP_{n-1} \),

\[
MC_{n_{21}} = \begin{bmatrix}
-\alpha & & -\alpha \\
-\alpha & 1 & -\alpha \\
0 & -\alpha & 1 & -\alpha \\
\vdots & \vdots & \vdots & \ddots \\
-\alpha & 0 & \cdots & 1 & -\alpha \\
-\alpha & 1 & -\alpha & 0 & \cdots \\
-\alpha & 1 & -\alpha & 0 & \cdots \\
-\alpha & 1 & -\alpha & 0 & \cdots \\
-\alpha & 1 & -\alpha \\
\end{bmatrix}
\]
and

\[ MC_{n1} = \begin{bmatrix} -\alpha & -\alpha \\ 1 & -\alpha \\ -\alpha & 1 & -\alpha \\ \vdots & \vdots & \vdots \\ -\alpha & 1 & -\alpha \\ -\alpha & 1 & -\alpha \end{bmatrix}. \]

We can calculate \( \det(MC_{n21}) \) and \( \det(MC_{n1n}) \) using cofactor expansion along the first row:

\[
\begin{align*}
\det(MC_{n21}) &= -\alpha C_{11} + (-1)^{1+n-1}(-\alpha)C_{1,n-1}, \\
&= -\alpha d_{n-2}(\alpha) + (-1)^n(-\alpha)(-\alpha)^{n-2} \\
&= -\alpha d_{n-2}(\alpha) + (-1)^n(-\alpha)^{n-1} \\
&= -\alpha d_{n-2}(\alpha) - \alpha^{n-1}.
\end{align*}
\]

\[
\begin{align*}
\det(MC_{n1n}) &= (-\alpha)C_{11} + (-1)^{1+n-1}(-\alpha)C_{1,n-1} \\
&= (-\alpha)(-\alpha)^{n-2} + (-1)^{1+n-1}(-\alpha)d_{n-2} \\
&= (-\alpha)^{n-1} - (-1)^n \alpha d_{n-2}(\alpha)
\end{align*}
\]

Plugging the cofactors in, we get

\[
\begin{align*}
\det(MC_n) &= d_{n-1}(\alpha) + \alpha \left( -\alpha d_{n-2}(\alpha) - (\alpha)^{n-1} \right) + (-1)^{n+1}(-\alpha) \left( (-\alpha)^{n-1} - (-1)^n \alpha d_{n-2}(\alpha) \right) \\
&= d_{n-1}(\alpha) - \alpha^2 d_{n-2}(\alpha) - \alpha^n - \alpha^n - \alpha^2 d_{n-2}(\alpha) \\
&= d_{n-1}(\alpha) - 2\alpha^n - 2\alpha^2 d_{n-2}(\alpha).
\end{align*}
\]
To calculate the closed formula for the Katz matrix for a cycle, we will need the following lemma.

**Lemma 4.16.** Let $C$ be a $k \times l$ matrix with zero entries except for the $(l,1)$st entry which is $a$. Let $D$ be an $l \times k$ matrix with zero entries except for the $(l,1)$st entry which is $a$. Let $E$ be an $l \times l$ matrix with entries $e_{ij}$.

Then

$$CED = a^2 \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e_{1,l} & 0 & \cdots & 0 & 0 \end{bmatrix}.$$ 

**Proof.** Let $C$, $D$, and $E$ be as described. Then

$$CED = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ a & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1,l} \\ e_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ e_{l,1} & e_{l,2} & \cdots & e_{l,l} \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ a & 0 & \cdots & 0 \end{bmatrix}$$

$$= a \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ e_{11} & e_{12} & \cdots & e_{1,l} \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ a & 0 & \cdots & 0 \end{bmatrix}$$

$$= a^2 \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ e_{1,l} & 0 & \cdots & 0 \end{bmatrix}.$$ 

□
Theorem 4.17. For a cycle \( H \) with \( n > 4 \) vertices and \( 1 \leq i \leq j \leq n \) and \( 0 < \alpha < \frac{1}{\rho(A)} \), then

\[
K_{i,j}^n(\alpha) = \frac{N}{D} - 1_{ij}
\]

where

\[
N = \alpha^{j-i}d_{n-j+i-1}(\alpha) + \alpha^{n-j+i}d_{j-i+1}(\alpha)
\]

and

\[
D = d_{n-1}(\alpha) - 2\alpha^n - 2\alpha^2d_{n-2}(\alpha)
\]

and \( 1_{ij} \) is the Kronecker delta function, equal to 1 if \( i = j \) and 0 otherwise.

Proof. Let \( MC_n = I - \alpha A \). We first observe that given a sequence of connected nodes, we can rearrange the nodes by starting with a different node. Thus for \( i \leq j \) where \( j - i = r \),

\[
K_{i,j}^n(\alpha) = K_{1,(j-r) \mod n}^n(\alpha).
\]

Thus we only need to calculate \( K_{1,j}^n(\alpha) \).

Using Cramer’s rule, we can find \( MC_n^{-1} \):

\[
MC_{n, i,j}^{-1} = (-1)^{i+j} \frac{\det(MC_n(i, j))}{\det(MC_n)}.
\]

By Theorem 4.15, we know \( \det(MC_n) = D \). Next, we calculate \( \det(MC_n(1, j)) \). For \( j = 1 \), \( MC_n(1, 1) = MP_{n-1} \) so \( \det(MC_n(1, 1)) = d_{n-1} \).

Now consider the case where \( 1 < j \leq n \). Splitting into block matrices where \( B \) is the upper left \((j-1) \times (j-1)\), \( C \) is the \((j-1) \times (n-j)\) upper right, \( D \) is the \((n-j) \times (j-1)\) and \( E \) is the \((n-j) \times (n-j)\) bottom right, we see that \( B \) is upper diagonal with \(-\alpha\) along the diagonal and \( E = MP_{n-j} \). The only nonzero entry of \( D \) is the \((n-j, 1)\)th entry and the only nonzero entries of \( C \) is the \((j-1, 1)\)st entry, both of which are \(-\alpha\).
$C$ and $D$ are the same as in Lemma 4.16 so

$$CE^{-1}D = \alpha^2 \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e_{1,n-1}^{-1} & 0 & \cdots & 0 & 0 \end{bmatrix}.$$  

Since $E = MP_{n-j}$, $e_{1,n-1}^{-1} = K_{1,n-1}^{n-1}$ for a path. Using Theorem 4.8, $e_{1,n-1}^{-1} = \frac{\alpha^{n-j-1}}{d_{n-j}(\alpha)}$.

Now

$$B - CE^{-1}D = \begin{bmatrix} -\alpha & 1 & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & -\alpha & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \vdots & \ddots & \ddots & -\alpha & 1 \\ -\alpha^2 \frac{\alpha^{n-j-1}}{d_{n-j}(\alpha)} & 0 & \cdots & \cdots & 0 & -\alpha \end{bmatrix}$$

so using cofactor expansion again,

$$\det(B - CE^{-1}D) = (-1)^2(-\alpha)^{j-1} - (-1)^{j-1+1}\alpha^2 \frac{\alpha^{n-j-1}}{d_{n-j}(\alpha)} d_{j-2}(\alpha).$$

Thus

$$\det(MC_n(1,j)) = \left[(-\alpha)^{j-1} - (-1)^j \frac{\alpha^{n-j+1}}{d_{n-j}} d_{j-2}(\alpha)\right] d_{n-j}(\alpha)$$

$$= (-\alpha)^{j-1} d_{n-j} - (-1)^j \alpha^{n-j+1} d_{j-2}(\alpha).$$

Generally, for any $1 \leq i \leq j \leq n$, we can replace $j$ with $j-i+1$:

$$\det(MC_n(i,j)) = (-\alpha)^{j-i} d_{n-j+i-1} - (-1)^{j-i+1} \alpha^{n-j+i} d_{j-i-1}(\alpha).$$

Multiplying by $(-1)^{j+i}$ yields the result.
Given that the analytic formula for the Katz matrix for a cycle consists of $\alpha$ and $d_i(\alpha)$ and from computational experiments, we suspect that the terms in the Katz matrix converge, leading to the following conjecture.

**Conjecture 4.18.** $K_{ij}^n(\alpha)$ converges for fixed $i, j, \alpha$ as $n$ goes to infinity.

The effective resistance between two nodes on a cycle graph can be calculated using the formula

$$
(1/|i-j| + 1/(n-|i-j|))^{-1}.
$$

This shows that effective resistance increases as nodes are further apart, with the maximum at the node or nodes that are $\left\lfloor \frac{n}{2} \right\rfloor$ apart. So the minimum effective resistance still occurs at nodes that are two apart. Since we can rearrange nodes in a cycle, all pairs of nodes that are a set distance apart have the same effective resistance, leading to the following theorem.

**Theorem 4.19.** For a cycle graph with $n$ nodes with $i \leq j$,

$$
\min (K_{j-i=2}(\alpha)) - \max (K_{j-i=3}^n(\alpha)) = \frac{N}{D},
$$

where

$$
N = \alpha^2 d_{n-3} + \alpha^{n-2} + \alpha^3 d_{n-4} + \alpha^{n-3} (1 - \alpha^2)
$$

and

$$
D = d_{n-1}(\alpha) - 2\alpha^n - 2\alpha^2 d_{n-2}(\alpha).
$$

**Proof.** Since all nodes that are two apart have the same Katz value and similarly for nodes that are three apart, we can calculate the formula explicitly with $i = 1$ and $j = 3$ or $j = 4$.

Recall that

$$
K_{1,j}^n(\alpha) = \frac{\alpha^{j-1}d_{n-j}(\alpha) + (\alpha)^{n-j+1}d_{j-2}(\alpha)}{D}.
$$
Then,

\[
K^n_{13}(\alpha) - K^n_{14}(\alpha) = \frac{\alpha^2 d_{n-3} + \alpha^{n-2} d_1 - \alpha^3 d_{n-4} + \alpha^{n-3} d_2}{D} = \frac{\alpha^2 d_{n-3} + \alpha^{n-2} - \alpha^3 d_{n-4} - \alpha^{n-3}(1 - \alpha^2)}{D}.
\]

\[\square\]

**Conjecture 4.20.** For a cycle graph, the Katz value decreases as the distance between nodes increases. So for \(1 \leq r \leq \frac{n}{2}\),

\[
K^n_{|j-i|=r}(\alpha) - K^n_{|j-i|=r+1}(\alpha) > 0.
\]

### 4.5 Conclusion

As we have seen for three classes of simple graphs, we have found a closed formula for the Katz matrix. We were then able to use this to show various properties, including convergence of the Katz matrix and how changing \(\alpha\) would give the same ordering as effective resistance. Some future problems related to this work are identifying \(\alpha\) that changes the Katz ordering for all distances, proving convergence of the Katz matrix for forks, expanding this idea to more complicated classes of graphs, and seeing if this extends to graphs with subgraphs isomorphic to these simple graphs.
CHAPTER 5. PREDICTING TEAM TRANSITIONS IN SPORTS DATA

5.1 INTRODUCTION

Social connections exist between and across many different types of groups. This includes social relations between individuals in different schools, families, religious and professional groups, or any other group defined by affiliation. An important feature of these social connections is that they are not always between members of the same group.

This chapter will discuss how these relations affect individual transitions between groups by looking at members of professional sports teams, specifically the MLB (Major League Baseball) and in the NBA (National Basketball Association). The transitions we study are the player transitions from one team to another within MLB and the NBA, respectively, during a player’s career.

These baseball and basketball teams can be thought of as specific types of professional groups, i.e., groups of individuals employed by the same employer, with a similar skill set, and a specific objective. Transitions between such professional groups are not the same as transitions between social communities, which are communities defined purely in terms of social interactions [58]. The dynamics of individual, or more generally, node transition between social communities within social networks is a relatively new field taking cues from mathematics [59] and computer science [60]. We refer the interested reader to a survey of work [61].

Transitions between professional groups have been previously studied by sociologists and economists (see for example [62, 63, 64, 65, 66, 67, 68, 69, 70]). To understand these transitions, features such as the strength of ties between workers [62], the geography of transitions [64], the role social networks play in finding employers and employees [63, 65] are considered. In regards to the social network aspect of such transitions, specific questions that have been considered include whether companies hire workers through social connections
of productive employees [66], whether employees hired through referrals are more likely to stay [68], how unemployed workers find jobs through their social networks [67, 70], differences in salaries between employees found by referrals [63], and motivations for employees to grow their professional social networks [69].

Since social connections often exist between individuals in different professional groups, a natural question is how these connections influence transitions between such groups. Here we consider the specific question of how social data, player performance, and team statistics can be used to improve our ability to predict the way professional athletes transition from one professional group to another, i.e., one team to another specifically in the MLB and NBA.

Other ways of analyzing team transitions in the NBA and MLB that have been considered include analyzing the labor market’s influence on professional baseball and basketball players, which has been studied extensively beginning with the classic work of Rottenberg [71]. The transition between teams in baseball has been studied more recently in light of changes to rules governing transitions [72] and also in terms of player productivity [73]. In professional basketball, hiring decisions have been considered with regards to first-hand experience [74] and also in terms of increased productivity [75]. Tools from network theory have also been used to study interactions in both baseball [76] and basketball [77, 78]. However none of the listed works consider how a player’s social-professional network influences transitions between teams despite the fact that social network analysis has become increasingly popular in sports analytics [79].

From the various professional groups that exist, a major factor in choosing to analyze team dynamics of the MLB and NBA is the availability of data. This includes the player’s social data but also information such as the player’s performance, and other factors that could be used to predict transitions between teams. The size of the data set, measured in terms of the number of individuals, the number of years it spans, and variety of statistics is also important as our analysis relies on machine learning algorithms that require sufficient
amounts of data to both decrease bias and improve accuracy (see Section 5.4 and [31]).

To address the question of what influences transitions between professional teams we consider three factors: individual performance, team fitness, and social data. Of the three, *individual performance* is perhaps the most natural to consider. The idea is that poor performance presumably motivates managers to replace players while high performance makes players more attractive to other teams. To understand this tendency of professional athlete’s to move from one team to another, we also considered team *fitness* together with *individual performance*. Here the idea is that an athlete with high performance is more likely to transition to a team he or she perceives as either being fit, or becoming more fit [80]. A natural assumption is that an athlete with low performance is more likely to get traded to a team with lower fitness, which can help in predicting transitions.

In the context of group dynamics there are many ways to measure *fitness* including how cohesive or stable the group is [81], the strength of individual members, and the ability of the group to perform its designated task. In this study, we considered two proxies which we use to measure the fitness of our groups. The first is *relative team ranking*, which acts as a measure of a team’s ability to achieve success. The second proxy for team fitness is the *financial valuation* of a team, which is based on the notion that a team on firm financial footing is more stable and can likely offer high performers more competitive salaries [82].

The third factor we consider is the *social interactions* individuals have within their social-professional network (see Section 5.3.4). The idea here is that, if the player has social connections to other players from other teams, then this may indicate at least a predisposition to move to that team when compared to other teams.

Our study considers two proxies for the players’ social-professional networks. The first proxy is a snapshot of the Twitter connections that existed between players in 2019. This data set describes which players followed which players up to when this data was collected. The second proxy of a player’s social-professional network is created using the college the player attended, which we refer to as the player’s *college network*. This was collected for
players in the NBA and was done to test whether a shared collegiate experience has an effect
on the transitions players make throughout their career.

As the Twitter data is not retro-actively timestamped it can only be used to predict
player transitions after it was collected. However, it can be used to infer transitions that
occurred prior to 2019. Our first goal is to understand how well this data can be used to predict
player transitions for the 2020-2021 seasons then to use this data to infer transitions
prior to and including 2019. The difference between what we can infer and what we can predict
gives us a sense of the changes in players’ social-professional activity on Twitter
before and after transitions.

Similarly we use the NBA players’ college network to predict future player transitions.
Our goal here with using this college data, similar to using Twitter data, is to understand
how well this data predicts future transitions when it is used with and without performance
and fitness data. The overarching question we hope to answer is how different combination
of these three factors—individual performance, team fitness, and social interaction—improves
or decreases our ability to infer and predict to which team an individual will transition to.

What we find is that the use of Twitter data significantly improves our our ability to
predict transitions for players in the NBA but does little to change our accuracy in predicting
MLB transitions. Similarly, the addition of performance and team data only slightly change
our prediction accuracy in both the NBA and MLB from that of a random guess (see Section
5.5.1). Including the college a player attended, which is our second proxy for social data
in the NBA, similarly increases our prediction accuracy nearly as much as Twitter data.
Overall this suggests that social connections are much more important in the NBA than
MLB in predicting the destination of player transitions.

For inferring past transitions the addition of performance data, team fitness data, and
social data each improve the accuracy of the machine learning algorithms we consider for both
the players in MLB and in the NBA. Performance and team fitness, perhaps surprisingly,
only modestly raise the accuracy of our results. The inclusion of social data from Twitter,
however, dramatically improves the predictive ability of these algorithms when predicting
past transitions in every case we consider. Here predictions are typically better for the NBA
than for MLB. This again suggests that social connections are less important in MLB than
in the NBA. (See Section 5.2 for a summary of these results.)

An interesting feature of the Twitter data is that, over time, an increasing number of
players in both the MLB and NBA begin to follow other players (see Figure 5.1). When we
limit our methods to the latter decade of our study when Twitter use is at its highest, we
can infer transitions much more accurately for both MLB and the NBA than for the first
decade (see Tables 5.12 and 5.11).

We also find that although the Twitter networks for baseball and basketball are fairly
different in size, the two networks are strikingly similar. Specifically, they have very similar
network statistics including mean degree, fraction of nodes in the largest strongly connected
component, mean distance between connected node pairs, clustering coefficient, reciprocity,
and the degree assortativity (see Table 5.13). Therefore, it seems unlikely that the particular
structure of these networks can explain why Twitter data leads to higher prediction and
inference accuracy for basketball when compared to baseball.

The chapter is organized as follows. In Section 5.2 we give a brief summary of our
results regarding prediction and inference accuracy in both MLB and the NBA. In Section
5.3 we describe our methodology including which social and nonsocial data we collected
and some of the features of this data. This includes performance, fitness, social, and other
data we used to train the machine learning algorithms we selected. In Section 5.4 we give a
brief description of these algorithms. In Section 5.5 we describe how different combinations
of social and nonsocial data effect the accuracy of these algorithms. In Section 5.6 we
analyze the basic statistics of the baseball and basketball Twitter networks. In Section 5.7
we discuss a few limitations of our data and methods of analysis. We conclude in Section
5.8 with some open questions that specifically relate to how this type of analysis could be
extended to study group transitions in other settings, i.e., other professional groups and
more general social networks. This work was done with Emily Evans, Joseph Leung and Benjamin Webb and is currently in submission at PLOS ONE, with an older version on arXiv at https://arxiv.org/abs/2009.00550.

5.2 Summary of Results

Here we give a brief summary of the results found in Section 5.5 regarding the accuracy of the machine learning algorithms we consider. The different types of data we use to determine player transition between teams are broadly speaking player performance, team fitness, and social data, which are described in detail in Section 5.3.

When predicting future transitions in both the NBA and MLB we find that the addition of player and team data does little to raise our prediction accuracy over the probability $1/29 \approx 3.45\%$ of a correct random guess. In fact, using all nonsocial data improves our accuracy by at most 1% over this probability. In contrast, using social data dramatically improves our accuracy in predicting transitions in the NBA. Using Twitter data alone allows for an accuracy of up to 20% while using college data gives us an accuracy of up to 17.4% with similar F1 scores. Using social data to predict transition in MLB, however, has little effect, only slightly improving scores beyond the probability of a correct guess. In fact, the inclusion of social data alone only gives an accuracy of 4.6% which is not as good as only taking into account the career length of the player which gives an accuracy of 6.7%. Here the inclusion of social data typically decreases the accuracy of our machine learning algorithms for baseball leading us to conclude that social data does not provide any valuable information as far as transitions are concerned. (See Tables 5.7 and 5.8.)

We find when inferring past transition that the addition of each of player performance, team fitness, and social data each improve the predictive ability of the algorithms we consider. However, as mentioned performance data by itself does little to raise our inference and prediction accuracy. Specifically, including performance data raises the accuracy of these algorithms by at most 1% for both the MLB and the NBA over the probability of a correct
guess. Similarly, using team fitness data improves accuracy by at most 0.85% for the MLB and 1.35% for the NBA. Using all nonsocial data together including performance, team fitness, player position, team, and career length improves accuracy by at most 1.055% for the MLB and 5.25% for the NBA (see Tables 5.10 and 5.9).

When using social data to infer past transitions the situation improves significantly. When using data derived from Twitter connections, with no other information, the prediction accuracy of the algorithms can be as high as 21.2% for the MLB and 27.4% for the NBA, an increase of 17.75% and 23.95% over random guessing, respectively. Using college data for the NBA similarly increases the accuracy of prediction to as much as 8.8%. The F1 scores follow the same pattern. It is worth noting that our maximum accuracy is found in the MLB using only the player’s team together with Twitter data while in the NBA our maximum accuracy is found using only the team’s fitness combined with Twitter data (see Tables 5.10 and 5.9).

As mentioned in the introduction, over time an increasing number of players in both the MLB and NBA follow other players (see Figure 5.1). When we limit our predictions to the last decade of our study when Twitter use is at its highest, we can predict with up to 19.4% of the time where a player will transition to in the MLB and up to 30.2% of the time in the NBA (see Tables 5.12 and 5.11).

5.3 Methodology

5.3.1 Data Collection. Performance data was scraped from www.basketball-reference.com/leagues and www.baseball-reference.com/leagues using Python and Beautiful Soup both of which are packages used to extract data from htmls. Since we looked at historical data and used appropriate crawl delays, we met the scraping terms defined in the robots.txt file for both sites. The Twitter data was collected using the Twitter API. First, we scraped the Twitter usernames for each player listed on www.baseball-reference.com/friv/baseball-player-twitter-accounts.shtml and www.basketball-reference.com/
friv/twitter.html. Then using tweepy, a python package for connecting to the Twitter API, we were able to collect the Twitter IDs of the other MLB/NBA players that each player followed. We chose to look at those “followed” instead of those “following” because it significantly sped up the data collection process. By using the Twitter API and tweepy, we were able to follow all necessary protocols, including rate limits and only accessing publicly available information. The data, as well as the code to run the predictions, is publicly available at github.com/rdorff/BasketballBaseball and on Dryad, a data publishing site [83].

As all data used in the development of our datasets came from publicly available websites and included only factual data about people, IRB approval was not required. In addition we anonymized the data.

5.3.2 Baseball performance dataset. Major League Baseball consists of 30 teams evenly split between the American and the National league. Each full team roster consists of 40 players. A baseball season consists of 162 regular season games with some player in certain positions playing most games, and some players in positions like pitcher playing in a fraction of these games. In our analysis we consider 3 high-level positions that players can be in: pitcher, catcher, and fielder, where the position of fielder represents all other positions. We note that positions are more fine grained, but typically players who play in the infield and outfield have some flexibility in the actual position that they play. We singled out the catcher position because, usually, one of the catchers serves as team captain. It is worth noting that the exact composition of a team’s roster varies with some teams having more of one position than another. The baseball performance data for the 2002 – 2021 seasons we use were obtained from https://www.baseball-reference.com. Although the website contains a wide variety of statistics such as number of games played, points scored, and total hits for our analysis we focused primarily on a few advanced statistics and a few engineered statistics instead of generic totals. The data collected for a player includes the main position played, the team played on, and the player’s age for a given season. The advanced data we collected for each player includes: the fielding percentage (FLD%), offensive winning
OWn% is the percentage of games that a team would win if the batting was done by 9 copies of the player, assuming average offense and defense. BtWins estimates a player’s total contribution to his team’s wins with his bats. BtRuns is an estimate of a player’s running contribution to a team’s wins. FLD is the number of putouts and assists divided by the sum of putouts, assists, and fielding errors. This data provides an overall picture of a player’s performance during the season. While other metrics are often used in evaluating player performance, we selected metrics that were representative of both pitching and catching positions and were available on www.baseball-reference.com.

We then created the following engineered data for each player and each season:

- Position – created by merging actual players positions into the three positions we identified: pitcher, catcher, and fielder.
- Career length – number of prior seasons played until the year under consideration (i.e., rookies have a career length of zero).
- Leave variable – specifies if a player is to leave their current team after the season under consideration.
- Target variable – specifies which team a player plays for the next year, or if they do not return to play that next year.

The leave variable is critical in identifying which players transition at the end of the season to another team allowing us to focus on predicting the transitions of only those players. The target variable provides us the ground truth for measuring the accuracy of our results.

To illustrate our collected and engineered features we display three seasons of data for a random baseball player in Table 5.1. We note that at the end of 2017, this player switched teams, (to the New York Yankees), hence the engineered field of target was set to NYY.
Table 5.1: Three years of collected and feature engineered data for a random baseball player. We observe that at the conclusion of the 2017 season, this player transitioned from the Miami Marlins (MIA) to the New York Yankees (NYY). Thus in 2017 his target value is set to NYY. In this table, and in Table 5.3 we use the abbreviation CL for career length (an engineered variable).

We show the specific distribution of players, players leaving their team, players retiring, and players transitioning for each year in Table 5.2. We note that each year approximately 50% of players leave their team in some manner.

Table 5.2: The number of Major League baseball players per year comprising the 2002-2020 baseball seasons. We observe that each year 48.7% of the players leave their current team on average. Of those that transition about one-half, 49.0%, of the players transition to a new team and the other half of the players end their professional careers.

5.3.3 Basketball performance dataset. Similar to baseball, the National Basketball Association consists of 30 teams evenly split between two conferences. In the NBA, each
team’s roster consists of only 17 players, with only eight players required to be active at any one time. Basketball has five positions: point guard, shooting guard, small forward, power forward, and center; however most basketball players are capable of playing in more than one of the positions. Each team plays 82 games in a standard season.

The basketball performance data for the 2001–2021 seasons was collected from www.basketball-reference.com. Similar to baseball we choose to use advanced data statistics, focusing on three advanced stats. PER, *Player Efficiency Rating*, measures how much a player produced in one minute of play. *Win Shares* or WS is an estimate of how many wins were contributed by a player. BPM, *Box Plus/Minus*, is an estimate of the number of points per 100 possessions that a player contributed. To illustrate both the collected and engineered statistics we consider a few seasons of a representative basketball player’s career in Table 5.3, and note that he switched teams in 2018.

<table>
<thead>
<tr>
<th>Season</th>
<th>Team</th>
<th>PER</th>
<th>WS</th>
<th>BPM</th>
<th>CL</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017</td>
<td>NOP</td>
<td>23.2</td>
<td>1.6</td>
<td>5.5</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>2018</td>
<td>NOP</td>
<td>22.6</td>
<td>4.7</td>
<td>4.7</td>
<td>7</td>
<td>GSW</td>
</tr>
<tr>
<td>2019</td>
<td>GSW</td>
<td>21.4</td>
<td>2.4</td>
<td>3</td>
<td>8</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5.3: Three years of collected and engineered data for our representative basketball player. At the conclusion of the 2018 season, this player transitioned from the New Orleans Pelicans (NOP) to the Golden State Warriors (GSW) so the target variable was set to be GSW.

Similar to baseball, we also created engineered features for individuals each season. Since basketball has only 5 positions, we did not modify this feature, and only engineered values for career length, leave and target. Ultimately, there were 3688 basketball players who switched teams between 2001 and 2020. The distribution of the leaving players is shown in Table 5.4. The average percent of players leaving their team each year is 67%, and approximately 32% of those that leave retire.

5.3.4 Social network datasets. As it is extremely difficult to impossible to create a ground truth social-professional network for players, we created an approximation of this net-
Figure 5.1: A comparison of the percentage of players who had an active Twitter account before and including July 2020 and who played between 2002 – 2018 for the MLB (blue) and between 2001 – 2019 for the NBA (orange). The datasets we use only give the number of players that have Twitter and who played in a given year, not the specific players that used Twitter during that year. This explains why there is Twitter data as early as 2001, even though Twitter began in 2006.
<table>
<thead>
<tr>
<th>Season</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Players</td>
<td>441</td>
<td>440</td>
<td>428</td>
<td>442</td>
<td>464</td>
<td>458</td>
<td>458</td>
</tr>
<tr>
<td>Total Leaving</td>
<td>251</td>
<td>238</td>
<td>284</td>
<td>289</td>
<td>282</td>
<td>245</td>
<td>253</td>
</tr>
<tr>
<td>Retiring</td>
<td>84</td>
<td>43</td>
<td>85</td>
<td>78</td>
<td>62</td>
<td>98</td>
<td>89</td>
</tr>
<tr>
<td>Switched Teams</td>
<td>167</td>
<td>153</td>
<td>206</td>
<td>227</td>
<td>184</td>
<td>156</td>
<td>166</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Season</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Players</td>
<td>450</td>
<td>444</td>
<td>442</td>
<td>452</td>
<td>478</td>
<td>468</td>
</tr>
<tr>
<td>Total Leaving</td>
<td>258</td>
<td>259</td>
<td>281</td>
<td>257</td>
<td>295</td>
<td>273</td>
</tr>
<tr>
<td>Retiring</td>
<td>85</td>
<td>79</td>
<td>71</td>
<td>73</td>
<td>100</td>
<td>83</td>
</tr>
<tr>
<td>Switched Teams</td>
<td>173</td>
<td>180</td>
<td>210</td>
<td>184</td>
<td>195</td>
<td>190</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of Players</td>
<td>481</td>
<td>492</td>
<td>476</td>
<td>486</td>
<td>540</td>
<td>530</td>
<td>529</td>
</tr>
<tr>
<td>Total Leaving</td>
<td>267</td>
<td>279</td>
<td>259</td>
<td>270</td>
<td>310</td>
<td>334</td>
<td>296</td>
</tr>
<tr>
<td>Retiring</td>
<td>86</td>
<td>96</td>
<td>90</td>
<td>94</td>
<td>128</td>
<td>130</td>
<td>94</td>
</tr>
<tr>
<td>Switched Teams</td>
<td>181</td>
<td>183</td>
<td>169</td>
<td>176</td>
<td>182</td>
<td>204</td>
<td>202</td>
</tr>
</tbody>
</table>

Table 5.4: The number of National Basketball Association (NBA) players per year comprising the 2001-2020 seasons. We note that there are approximately half the number of players in this dataset compared to the baseball dataset. Also, a higher percentage of players transition annually in the NBA (on average about 58%). A high amount of players switch teams each year, about 67%, while only 32% retire.

work utilizing Twitter data. Twitter is a social networking site that allows users to exchange short “Tweets” with followers. Twitter was chosen because player Twitter handles were available from from both [www.basketball-reference.com](http://www.basketball-reference.com) and [www.baseball-reference.com](http://www.baseball-reference.com), and because Twitter provides an easy API that can be used to obtain both the followers and those followed by a user. A downside of using Twitter is that the “followers” information is not time stamped. Hence our network created with the Twitter data is a snap shot of the relationships that existed before and up to July 2020 when we scraped the data with no way to pinpoint when a player started to follow another.

With the Twitter data we created a directed social network of players where player A has a connection directed to Player B if Player A followed Player B, which we refer to as our baseball Twitter network and basketball Twitter network, respectively. Of the 4207 unique baseball players that switched to a different team from 2002 – 2018, we were able to obtain
Figure 5.2: Left: A histogram of the degree (centrality) for the baseball Twitter network and the basketball Twitter network shown in orange and grey, respectively. Right: A histogram of the number of colleges that had one of more basketball players join the NBA from 2001–2019.

Twitter handles for 702 of them. For basketball players that switched between 2001 – 2019, we were able to collect Twitter handles for 784 of the 1847 players, a significantly larger percentage indicating how active NBA players are on Twitter compared with MLB player (see Figure 5.1). The resulting Twitter network is a social network with 53690 directed edges for baseball and 43827 directed edges for basketball. Most players in both datasets have a relatively small number of connections or degree (centrality) to others, which is the number of followers together with number of players followed for a specific player. A few players do have a large number of connections though (over 100). The distribution of connections for both baseball and basketball players having at least one Twitter connection is shown in Figure 5.2 (left). (A more thorough analysis of these networks is given in Section 5.6.)

For the NBA we also investigate whether the college a player attended can serve as a proxy for social connections and whether this data helps predict where player’s transfer during their professional career. To test this idea, we pulled the college data for each basketball player from www.basketball-reference.com and created a categorical feature for colleges. If a player in this set did not go to college, they were included and their college category was N/A.

In total there were 259 different colleges attended by future NBA players in the data set. Figure 5.4 (right) shows the number of colleges in the data set that had a given number of
Figure 5.3: Left: The affinity network of a representative MLB player from the 2017 season. Right: The affinity network of a representative NBA player from the 2018 season. In both, edge weights indicate the player’s affinity score where darker edge colors indicate a higher score.

Players attend. The N/A category is excluded. For example, from the data there were 5 schools that each had between 15 and 17 players attend: Syracuse: 16, Michigan State: 15, Georgetown: 17, LSU: 15, and Georgia Tech: 16. 940 players did not attend a college.

Using our Twitter networks and the team each player played on for a given season we create an affinity network for each player as follows: For a given transitioning player we add a weighted edge connecting the player to each of the teams in the MLB/NBA. The weight of an edge is the number of other players from that team this player followed during that season, which we call the affinity score (see Figure 5.3). We emphasize that the social network between players is fixed across seasons but the social affinity score changes between seasons since players change teams. This weight gives a score of the affinity that a player has for the team for a given season. Since we do not allow a player who has been identified as transitioning to remain on their current team we set the affinity score for the current team to zero. Finally the way we handle mid-year transitions (i.e., midyear trades) is different between the two sports. In basketball we consider only the team the player was on at the beginning of the season. For baseball, due to the way information is presented at baseball-reference.com we omit players who transitioned during mid-season from the
calculation of the affinity score for a given year.

5.3.5 Team stratification engineered data. Using the idea that successful players move to successful teams, at least on average, we created a measure of team fitness. We collected data on each team’s dollar valuation for each year in question through Forbes.com. We also retrieved team rankings for each year from www.basketball-reference.com and www.baseball-reference.com The result is a number of new features in each of the players’ data (see Table 5.5 which extends Table 5.3).

<table>
<thead>
<tr>
<th>Season</th>
<th>Team</th>
<th>Rank</th>
<th>Value</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017</td>
<td>NOP</td>
<td>20</td>
<td>750</td>
<td>N/A</td>
</tr>
<tr>
<td>2018</td>
<td>NOP</td>
<td>8</td>
<td>1000</td>
<td>GSW</td>
</tr>
<tr>
<td>2019</td>
<td>GSW</td>
<td>2</td>
<td>3500</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5.5: Three years of collected team data for a representative basketball player.

5.4 Analysis Techniques

In this section, we describe the techniques used to make our group transition predictions. We chose to utilize machine learning methods, rather than more classical statistical techniques, to see if different metrics, rather than those traditionally used, could provide better predictive power. In a nod to more traditional techniques we include logistic regression for comparison. We did not include neural networks or deep learning algorithms because our data set was not rich enough support the data requirements of these algorithms and typical data augmentation techniques are not easily applicable to the problem at hand.

Since our question was a classification question, the majority of the techniques we use are ensemble methods: (a) Random Forests (Forest), (b) Extremely Randomized Trees (Extra-Trees), (c) Adaptive Boosting (AdaBoost), and (d) Extreme Gradient Boosting (XGBoost). As mentioned, we also use (e) Logistic Regression and (f) $k$-Nearest Neighbors (KNN).

Tools and Methods: The tests were run using Python, Pandas, and Scikit-learn. After collecting the data, we use Pandas, a Python database package, to create the final data sets.
Scikit-learn is a widely used Python package for machine learning. All of the algorithms except XGBoost were implemented using Scikit-learn algorithms. XGBoost was implemented using xgboost, a package for running XGBoost in several popular languages.

SkLearn’s GridSearch was performed to identify appropriate hyperparameters. Since the MLB and NBA data sets were different, the search was performed separately on both data sets. The values for the hyperparameters used can be found in the Table A.17 in the Appendix. For more information on the hyperparameters, see the sklearn documentation for each algorithm.

Since it was possible that the algorithms would predict the player’s current team, instead of calculating the predicted result directly, we used the predict_proba method to identify the top two most likely targets. If the top target was the player’s current team, then we predicted they would move to the team with the second highest probability. After the predictions were generated, we calculated the F1 score and accuracy using Scikit-learn’s accuracy_score and f1_score. For the latter, we used the macro average which counts the total number of false positives, true positives, and false negatives over each team. The accuracy shown here is the mean of the accuracy over these 100 runs.

5.5 RESULTS

Before presenting the results, we recall the overarching question “Does a player’s social-professional network influence which team the player transitions to?” As described in the previous section we apply a variety of machine learning techniques with and without social network information as a feature to answer this question. We note that in both sports the number of teams is 30. However, once we have identified a given player as transitioning to a new team we prohibit the player from transitioning to their current team. Hence each transitioning player has 29 possible teams to transition to, and the naïve probability of transitioning to a given team is approximately 3.45%.

We note that each experiment was performed 100 times and the presented accuracy
and F1 scores are the mean of these 100 experiments. Complete statistical data tables including confidence intervals and confusion matrices can be found at github.com/rdorff/BasketballBaseball.

As the Twitter data we collected does not contain dates players started following other players this limits the transitions we can predict to the time after it was collected. Hence, we can only predict transitions that happened in 2020-2021 using Twitter data (see Section 5.5.1). Later we use this data to infer past transitions that happened during the years 2001-2019 (see Section 5.5.2). The difference between our prediction and inference accuracy gives us a measure of how player’s social activity shifted after they transitioned from one team to another.

<table>
<thead>
<tr>
<th>Season</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLB</td>
<td>NA</td>
<td>23</td>
<td>31</td>
<td>30</td>
<td>48</td>
<td>43</td>
<td>39</td>
<td>62</td>
<td>61</td>
<td>69</td>
</tr>
<tr>
<td>NBA</td>
<td>41</td>
<td>31</td>
<td>48</td>
<td>60</td>
<td>50</td>
<td>43</td>
<td>58</td>
<td>73</td>
<td>82</td>
<td>123</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MLB</td>
<td>94</td>
<td>119</td>
<td>123</td>
<td>151</td>
<td>148</td>
<td>138</td>
<td>157</td>
<td>169</td>
<td>103</td>
<td>111</td>
</tr>
<tr>
<td>NBA</td>
<td>121</td>
<td>127</td>
<td>141</td>
<td>150</td>
<td>145</td>
<td>128</td>
<td>150</td>
<td>148</td>
<td>180</td>
<td>153</td>
</tr>
</tbody>
</table>

Table 5.6: The number of MLB and NBA players in the final dataset, which contains those players who switched to a different team and had a Twitter account.

5.5.1 Predictive Results. We collected player performance and team fitness data for years 2020-2021 using www.basketball-reference.com and www.baseball-reference.com. We did not gather new Twitter data, so that all Twitter connections in our data set were made before any team transitions occurred.

As mentioned in our summary section, our ability to predict transitions in the NBA using Twitter data is significantly different from our ability to do the same in MLB. When predicting transitions in the NBA we find that including Twitter data allows for a prediction accuracy of up to 20.3%. Using college data also has a significant effect on the accuracy of our predictions giving us an accuracy of up to 17.4%. Combining both of these social features slightly increases this probability to a maximum of 20.6%.
In contrast, using player performance and team fitness data has little effect on prediction accuracy. In fact, using this information without social data typically causes the prediction accuracy to drop below the probability of 3.45% of randomly choosing the correct team. This suggests that social data alone is useful in predicting where players will transition to in the NBA (see Table 5.7 as well as the more complete set of data found in Tables A.10, A.11, and A.12 in the Appendix.)

In MLB the results are essentially the same if we consider nonsocial data. Using player performance and team fitness to predict MLB transitions during the years 2020-2021 results in probabilities that are similar to those seen in the NBA predictions. The slight difference is that, whereas the NBA probabilities typically drop beneath 3.45% when including nonsocial data, MLB probabilities climb a few percent above this number on average. The major difference, between the NBA and MLB is that prediction accuracy is very low in MLB compared to the NBA when using social data. Using Twitter data in the NBA allows us to achieve accuracy up to 20.3%. In MLB the maximum accuracy we achieve using our algorithms and social data, by itself, is 4.6% only a percent higher than a random guess. The highest predictive accuracy we achieve in MLB is 6.5% when we use only the player’s position. In fact, adding social data often decreases our ability to predict player transitions suggesting that, at least in predicting future transitions, social-professional connections have little to do with a player’s transition from team to team in MLB. (See Table Table 5.8 as well as the more complete set of data found in Tables A.4, A.5 and A.6 in the Appendix.)

5.5.2 Inferring Prior Transitions. Since our Twitter data does not contain individual time-stamps indicating when one player starts to follow another, it is not possible to use this data to predict transitions that happened before this data was collected. However, it is possible to use the current state of the Twitter data we can collect to infer which transitions have already happened. That is, we can test to see if Twitter data contains enough information to reconstruct which transitions have already taken place.
<table>
<thead>
<tr>
<th>Features</th>
<th>Social</th>
<th>Accuracy</th>
<th>Accuracy</th>
<th>F1</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Twitter</td>
<td>Forest</td>
<td>Trees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position</td>
<td>No</td>
<td>0.015</td>
<td>0.014</td>
<td>0.005</td>
<td>0.006</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.196</td>
<td>0.193</td>
<td>0.161</td>
<td>0.156</td>
</tr>
<tr>
<td>Team</td>
<td>No</td>
<td>0.031</td>
<td>0.028</td>
<td>0.022</td>
<td>0.023</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.206</td>
<td>0.104</td>
<td>0.171</td>
<td>0.162</td>
</tr>
<tr>
<td>Career Length</td>
<td>No</td>
<td>0.063</td>
<td>0.063</td>
<td>0.035</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.199</td>
<td>0.200</td>
<td>0.164</td>
<td>0.161</td>
</tr>
<tr>
<td>Performance</td>
<td>No</td>
<td>0.017</td>
<td>0.013</td>
<td>0.011</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.186</td>
<td>0.190</td>
<td>0.154</td>
<td>0.156</td>
</tr>
<tr>
<td>Rank</td>
<td>No</td>
<td>0.035</td>
<td>0.028</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.195</td>
<td>0.210</td>
<td>0.158</td>
<td>0.160</td>
</tr>
<tr>
<td>Value</td>
<td>No</td>
<td>0.034</td>
<td>0.032</td>
<td>0.023</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.210</td>
<td>0.207</td>
<td>0.17</td>
<td>0.169</td>
</tr>
<tr>
<td>Twitter Only</td>
<td>Yes</td>
<td>0.200</td>
<td>0.203</td>
<td>0.155</td>
<td>0.163</td>
</tr>
<tr>
<td>College Only</td>
<td>No</td>
<td>0.174</td>
<td>0.172</td>
<td>0.126</td>
<td>0.124</td>
</tr>
<tr>
<td>All Social</td>
<td>Yes</td>
<td>0.200</td>
<td>0.206</td>
<td>1.63</td>
<td>1.63</td>
</tr>
<tr>
<td>All Features</td>
<td>No</td>
<td>0.045</td>
<td>0.044</td>
<td>0.032</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.187</td>
<td>0.196</td>
<td>0.157</td>
<td>0.163</td>
</tr>
</tbody>
</table>

Table 5.7: Basketball Prediction Results: The predictions accuracy and F1 score is shown for team transition in the NBA during the 2020 season for players who had Twitter accounts before 2020. Each row indicates the feature(s) used. We note that a “yes” in the social column implies Twitter data was used. The “All Social” row includes both Twitter and College data.
<table>
<thead>
<tr>
<th>Features</th>
<th>Social Only</th>
<th>Accuracy</th>
<th>Accuracy</th>
<th>F1</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Forest</td>
<td>XGB</td>
<td>Forest</td>
<td>XGB</td>
</tr>
<tr>
<td>Position</td>
<td>No</td>
<td>0.055</td>
<td>0.065</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.046</td>
<td>0.041</td>
<td>0.151</td>
<td>0.176</td>
</tr>
<tr>
<td>Team ID</td>
<td>No</td>
<td>0.052</td>
<td>0.056</td>
<td>0.021</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.054</td>
<td>0.038</td>
<td>0.160</td>
<td>0.204</td>
</tr>
<tr>
<td>Career Length</td>
<td>No</td>
<td>0.056</td>
<td>0.067</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.046</td>
<td>0.035</td>
<td>0.151</td>
<td>0.178</td>
</tr>
<tr>
<td>Performance</td>
<td>No</td>
<td>0.040</td>
<td>0.040</td>
<td>0.029</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.047</td>
<td>0.037</td>
<td>0.146</td>
<td>0.170</td>
</tr>
<tr>
<td>Rank</td>
<td>No</td>
<td>0.059</td>
<td>0.062</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.050</td>
<td>0.041</td>
<td>0.157</td>
<td>0.178</td>
</tr>
<tr>
<td>Value</td>
<td>No</td>
<td>0.056</td>
<td>0.049</td>
<td>0.025</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.046</td>
<td>0.040</td>
<td>0.151</td>
<td>0.173</td>
</tr>
<tr>
<td>Social Only</td>
<td>Yes</td>
<td>0.046</td>
<td>0.039</td>
<td>0.155</td>
<td>0.176</td>
</tr>
<tr>
<td>All Features</td>
<td>No</td>
<td>0.047</td>
<td>0.041</td>
<td>0.038</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.051</td>
<td>0.039</td>
<td>0.146</td>
<td>0.187</td>
</tr>
</tbody>
</table>

Table 5.8: Baseball Prediction Results: The prediction accuracy and F1 score is shown for team transition in MLB during 2020 season for players who had Twitter accounts. Each row indicates which feature(s) were used.

**Basketball Results.** A summary of the results inferring prior transitions in the NBA can be found in Table 5.9, and a more complete summary can be found in Tables A.7, A.8 and A.9 in the Appendix. In Table 5.9 we see that adding social data improves performance remarkably. Similar to our previous results regarding future predictions nonsocial features had very little impact on accuracy. Performance data alone is about the same as randomly guessing, while using only social data results in a much higher accuracy. In fact, adding social data improves accuracy across all features, and using only social data is worse than using social data with any other feature. Using Twitter data with each nonsocial feature results in a 28-29% accuracy in all cases. As this is higher than our future prediction accuracy this suggests that once players move teams their online activity shifts to indicate the new team they are on. Presumably the players begin to follow other players on their new team.

We also investigated how the use of college data, which we consider to be a form of social data, effects the predictiveness of our algorithms. For the years 2001-2019, college data is somewhat advantageous over Twitter data as it allows us to make future predictions.
Table 5.9: Basketball Inference and Prediction Results: (Top) The inference accuracy and F1 score is shown for team transition in the NBA during the time period 2001-2019 for players who had Twitter accounts. Each row indicates the feature(s) used. We note that a “yes” in the social column means Twitter data was used. (Bottom) The prediction accuracy and F1 score using the players’ college data to predict team transitions during 2001-2019 is shown.

<table>
<thead>
<tr>
<th>Features</th>
<th>Social</th>
<th>Twitter</th>
<th>Accuracy</th>
<th>F1</th>
<th>Accuracy</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Forest</td>
<td></td>
<td></td>
<td>Trees</td>
<td></td>
</tr>
<tr>
<td>Position</td>
<td>No</td>
<td>0.040</td>
<td>0.039</td>
<td>0.008</td>
<td>0.008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.294</td>
<td>0.291</td>
<td>0.302</td>
<td>0.302</td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>No</td>
<td>0.042</td>
<td>0.040</td>
<td>0.028</td>
<td>0.028</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.290</td>
<td>0.275</td>
<td>0.292</td>
<td>0.273</td>
<td></td>
</tr>
<tr>
<td>Career Length</td>
<td>No</td>
<td>0.047</td>
<td>0.050</td>
<td>0.028</td>
<td>0.031</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.295</td>
<td>0.296</td>
<td>0.300</td>
<td>0.301</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>No</td>
<td>0.037</td>
<td>0.037</td>
<td>0.034</td>
<td>0.033</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.287</td>
<td>0.294</td>
<td>0.287</td>
<td>0.298</td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>No</td>
<td>0.031</td>
<td>0.033</td>
<td>0.022</td>
<td>0.024</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.281</td>
<td>0.288</td>
<td>0.286</td>
<td>0.293</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>No</td>
<td>0.051</td>
<td>0.050</td>
<td>0.046</td>
<td>0.045</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.292</td>
<td>0.294</td>
<td>0.296</td>
<td>0.298</td>
<td></td>
</tr>
<tr>
<td>Twitter Only</td>
<td>Yes</td>
<td>0.293</td>
<td>0.292</td>
<td>0.298</td>
<td>0.297</td>
<td></td>
</tr>
<tr>
<td>All Features</td>
<td>No</td>
<td>0.087</td>
<td>0.085</td>
<td>0.084</td>
<td>0.083</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.273</td>
<td>0.282</td>
<td>0.276</td>
<td>0.282</td>
<td></td>
</tr>
<tr>
<td>College Only</td>
<td>No</td>
<td>0.084</td>
<td>0.086</td>
<td>0.074</td>
<td>0.076</td>
<td></td>
</tr>
</tbody>
</table>
regarding transitions rather than inferring them. We find that using college data increased prediction accuracy but not as much as inferring these transitions by using social data.

There are potentially two reasons for this. First, inferring prior transitions may be easier than predicting future transition in general. Second the decrease in accuracy may be due to the fact that even the most frequently attended schools have at most about a dozen active players each year (see Table A.16). To give some indication of the prevalence of a fellow alumni on the target team (i.e., the team being transitioned to), for every transitioning player we counted the number of alumni on the target team. As seen in the left panel of Figure 5.4, in less than half of the transition cases, a fellow alumni is on the team. In the right panel of Figure 5.4 we considered the slightly different transition problem of whether the prevalence of a fellow alumni influenced what team a rookie player begins on. As can be seen in that histogram, it is likely that alumni connections do not influence initial placement.

Accuracy and F1 score increased using Random Forests, XGB, KNN, and Extra Trees but fluctuated slightly using ADA and Logistic Regression. A complete listing of these scores are included in the appendix as Tables A.13) and A.14.

**Baseball Results.** We summarize the results of our machine learning experiments for inferring prior transitions in the MLB in Table 5.10. In the table each row indicates which features are used. For instance, in the first row only knowledge of the player’s position is used to predict where the player transitions to. In the second row both the player’s position
Table 5.10: Baseball Inference Results: The inference accuracy and F1 score is shown for team transition in MLB during the time period 2002-2019 for players who had Twitter accounts. Each row indicates which feature(s) were used. We note that a “yes” in the social column means Twitter data was used.

<table>
<thead>
<tr>
<th>Features</th>
<th>Social</th>
<th>Forest Accuracy</th>
<th>XGB Accuracy</th>
<th>F1 Forest</th>
<th>F1 XGB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>0.044</td>
<td>0.044</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.171</td>
<td>0.187</td>
<td>0.151</td>
<td>0.176</td>
</tr>
<tr>
<td>Position</td>
<td>Yes</td>
<td>0.179</td>
<td>0.218</td>
<td>0.160</td>
<td>0.204</td>
</tr>
<tr>
<td>Team ID</td>
<td>No</td>
<td>0.032</td>
<td>0.031</td>
<td>0.021</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.179</td>
<td>0.218</td>
<td>0.160</td>
<td>0.204</td>
</tr>
<tr>
<td>Career Length</td>
<td>No</td>
<td>0.051</td>
<td>0.050</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.172</td>
<td>0.189</td>
<td>0.151</td>
<td>0.178</td>
</tr>
<tr>
<td>Performance</td>
<td>No</td>
<td>0.032</td>
<td>0.036</td>
<td>0.029</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.172</td>
<td>0.189</td>
<td>0.146</td>
<td>0.170</td>
</tr>
<tr>
<td>Rank</td>
<td>No</td>
<td>0.039</td>
<td>0.039</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.177</td>
<td>0.190</td>
<td>0.157</td>
<td>0.178</td>
</tr>
<tr>
<td>Value</td>
<td>No</td>
<td>0.041</td>
<td>0.043</td>
<td>0.025</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.170</td>
<td>0.184</td>
<td>0.151</td>
<td>0.173</td>
</tr>
<tr>
<td>Social Only</td>
<td>Yes</td>
<td>0.174</td>
<td>0.187</td>
<td>0.155</td>
<td>0.176</td>
</tr>
<tr>
<td>All Features</td>
<td>No</td>
<td>0.041</td>
<td>0.040</td>
<td>0.038</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.169</td>
<td>0.204</td>
<td>0.146</td>
<td>0.187</td>
</tr>
</tbody>
</table>

and the player’s social network, i.e. affinity scores, are used. A more complete summary of the data can be found in Tables A.1, A.2 and A.3 located in the Appendix. Here, we observe that including social data always has a positive effect, bringing the algorithms’ maximum accuracy up to 17%. This is statistically significant (see the confidence intervals in Table A.3). Moreover, each of the nonsocial features yield approximately the same accuracy level, and the combination of all these features does not significantly improve any algorithm’s accuracy. This suggests that these individual features are either in some sense linearly dependent, i.e. they are imparting the same information about a particular player, or that the features work against each other in some way. The F1 scores follow the same pattern as the accuracy, with the highly accurate models having the highest F1 scores.

5.5.3 A Temporal Comparison. Using Twitter data to determine transitions in the NBA and MLB has at least two drawbacks. The first, already mentioned, is that this data
Table 5.11: Basketball Inference and Prediction Accuracy: (Top) The inference accuracy using Random Forest for team transition in NBA during the time periods 2002-2010, 2011-2019, and 2002-2019 is shown. (Bottom) The prediction accuracy using the players’ college data to predict team transitions during 2001-2019 is shown.

is not time-stamped. The second is that Twitter was not founded until 2006, and although players from the early years of our study have joined Twitter a much lower percentage of these players have accounts and thus our proxy social-professional network is less complete for those years (see Figure 5.1).

With this in mind we considered one additional test of the efficacy of using Twitter data by comparing the accuracy of our machine learning algorithms for the earlier years (2001-2010) and the later years (2011-2019). The results are shown in Tables 5.12 and 5.11 for MLB and the NBA, respectively. In every case considered in these tables, if social data is used the algorithm’s accuracy is significantly higher in the later time period when Twitter usage is higher than in the earlier time period (see Figure 5.1). For baseball, using Twitter data alone increases accuracy from around 10% to 20% as the average Twitter usage climbs from 16.4% to 45.3% during 2002-2009 and 2010-2018, respectively. For basketball, accuracy increases from about 21% to 31% as the average Twitter usage climbs from 28.7% to 64.3%
Table 5.12: Baseball Inference Accuracy: The inference accuracy using XGBoost for team transition in MLB during the time periods 2002-2010, 2011-2019, and 2002-2019 is shown.

during 2001–2009 and 2010–2019, respectively. This suggests that the more complete our information is on the social interactions of players the better we can infer and possibly predict their transitions.

5.6 Network Analysis of the Twitter MLB and NBA Data Sets

In this section we investigate the properties of both the MLB Twitter and NBA Twitter networks described in Section 5.3. We first consider the basic statistical properties of these networks and then compare their degree, eigenvector, closeness, and betweenness centralities.

Table 5.13: Basic statistics for the MLB and NBA Twitter networks using Mathematica. Properties measured are: total number of nodes $n$, number of directed edges $m$; mean degree $c$; fraction of nodes in the largest strongly connected component $S$; mean distance between connected node pairs $\ell$; clustering coefficient $C$; reciprocity $r$, and the degree assortativity $a$.  

<table>
<thead>
<tr>
<th>Network</th>
<th>$n$</th>
<th>$m$</th>
<th>$c$</th>
<th>$S$</th>
<th>$\ell$</th>
<th>$C$</th>
<th>$r$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLB</td>
<td>1364</td>
<td>76977</td>
<td>56.43</td>
<td>0.931</td>
<td>2.07</td>
<td>0.189</td>
<td>0.605</td>
<td>-0.043</td>
</tr>
<tr>
<td>NBA</td>
<td>1003</td>
<td>58750</td>
<td>58.57</td>
<td>0.971</td>
<td>2.15</td>
<td>0.190</td>
<td>0.614</td>
<td>-0.023</td>
</tr>
</tbody>
</table>
The basic network statistics we consider are the network’s total number of nodes $n$, number of directed edges $m$, mean degree $c$, fraction of nodes in the largest strongly connected component $S$, mean distance between connected node pairs $\ell$, clustering coefficient $C$, reciprocity $r$, and the degree assortativity $a$, defined in Section 2.6. For a more detailed description of these network quantities see [84].

In Table 5.13 these statistics are shown for both networks. Although the number of nodes and edges in these networks are, relatively speaking, quite different each of the other statistics in the table are very similar. In fact, it is striking how similar some of these statistics are. This suggests that these two networks have very similar structures which in turn suggests that the reason we have better predictions for the NBA versus the MLB is not due to specific structural features of these networks.

To give more evidence to the notion that the baseball Twitter and basketball Twitter networks have a similar structure, we note that the distribution of the networks’ degrees (Figure 5.2), in-degrees, and out-degrees (Figure 5.5) have very similar shapes and that the same holds for the networks’ eigenvector, closeness, and betweenness centralities (Figure 5.6). Here an individual’s in-degree is the number of Twitter followers they have while out-degree is the number of player’s they follow. An individual’s eigenvector centrality is high if they are followed by players that collectively have a high centrality. To have high closeness centrality a player’s mean distance to all other players in the network should be small. To have high betweenness centrality the player should be on many of the shortest paths between other pairs of players.

5.7 LIMITATIONS

We have considered using social networks to predict and infer player transitions in both professional basketball and baseball. Naturally, limitations in the data available, differences between the two sports, and our desire to have a similar model for both sports put constraints on our analysis. In this section we detail a number of these constraints and their
Figure 5.5: A histogram of the in-degree and out-degree centrality for the baseball and basketball-Twitter networks are shown left and right, respectively. Baseball is shown in orange and basketball is shown in blue in each histogram.

Figure 5.6: Histograms of the eigenvector centrality (EC), closeness centrality (CC), and betweenness centrality (BC) for the baseball and basketball-Twitter networks shown left, center, and right, respectively. Baseball is shown in orange and basketball is shown in blue in each histogram. Overlap between the two is grey.
Perhaps the largest constraint comes from the use of Twitter data as a proxy for our two social networks. Although Twitter allows us to create an approximation of the players’ social-professional network, this data is not time stamped. Consequently, it is not possible to determine whether a social connection existed before or after a transition was made between teams.

It is possible to avoid this issue by gathering data at regular intervals or even at the end of a “typical” year. This, now time-stamped, data could then be used to predict transitions in the following year without the ambiguity of knowing whether the social connection preceded the transition or not. One drawback to this strategy is that we may need to wait for a typical year. The 2020-2021 transitions are potentially quite distinct from any of the previous years due to the Covid-19 pandemic’s effect on the MLB and NBA. However, our results suggest that using time-stamped social data together with the methods introduced in this chapter could further extend our understanding of the influence of social interactions on group transitions in general social-professional networks.

Aside from the temporal limitations of our data, one limitation of our analysis is that we do not differentiate between free agent movement and trades between teams. The primary reason for this is the difference in contracts between the two sports, both in terms of obligated contract length and the way that free agency works. In baseball, players are contractually obligated to their teams for longer periods of time, and players remain contractually obligated to that team even if their contract to play is not renewed. In basketball, players can be considered as either unrestricted free agents or restricted free agents. In order to take into account the differences between the two leagues we would need to treat them separately, track trade combinations, i.e., these two players are traded for those three players, etc. and track when players resign with a team due to their restricted status.

Another limitation is that, although the financial health of the teams is considered, we did not consider the salaries of players who transitioned between teams. In baseball, it is possible
for a financially healthy team to acquire a very good player by offering a high salary despite having other highly paid players, but this is much less likely in basketball where salary caps are strictly enforced. In our analysis we ignore this difference. In a more detailed analysis it might be possible to track the salary cap space, along with player’s current contract numbers, or projected worth to determine if it is even possible for a player to join a given team. For this to work well, we would have to track the order in which transitions occur as teams release players to make salary cap room for a star player.

5.8 Conclusion

In this chapter we consider the question “Do social connections influence professional group transition?” in the context of both Major League Baseball and the National Basketball Association. Specifically, we analyze to what extent social connections can help predict how players change teams. We find that the addition of social data significantly improve the accuracy of our results. In particular we compare which of the following types of data player performance, team fitness, and social data are more predictive in the context of machine learning. We find that the addition of player performance and team fitness data can both slightly improve and slightly decrease the performance of our algorithms but overall has little effect on prediction and inference accuracy. In contrast, the use of social data significantly improves our ability to predict future transitions in the NBA bringing our accuracy up to 20%. In MLB the results are quite different as including social data does little to improve accuracy and in many cases actually degrades our accuracy.

For inferring past transitions the use of social data does improve our predictions for both the NBA and MLB. In fact, our highest accuracies were obtained in this manner. This suggests that once players shift teams their online activities shift in a way that indicates this transition. The difference between prediction and inference accuracy gives us a sense of how large this shift is.

The fact that performance and team data did little to change our scores is, to us, a
bit surprising. There may be several reasons for this lack of improvement. In separate experiments, we discovered that performative data does influence the likelihood of a player not returning to play the following year. That is, performance data seems to be better suited to answer “if” a player will leave a team rather than “where” the player will go. This is important in the sense that the number of players leaving the MLB and NBA is nearly equal to the number of players transitioning most years.

We also note that the social networks under consideration are strikingly similar, hence the differences in prediction accuracy between baseball and basketball are likely not due to network structure. We conjecture that the differences in accuracy with the inclusion of social data between baseball and basketball may, in fact, be partially due to the percentage of players for which we have social data. As further evidence we compare the accuracy of our machine learning algorithms on the early years of the data versus the later years of the data. Both baseball and basketball show an increase in the percentage of players with social information and also an increase in the accuracy of the algorithm in the later years. Again, we temper these results with a reminder that these social networks were created from un stamped Twitter data and consequently, it was not possible to determine whether a social connection existed before or after a transition was made between teams.

To counter this limitation, we also consider college attendance as a proxy for social-professional network in the NBA. We find that this can also increase the prediction accuracy of our chosen algorithms. We were able to obtain college information for a larger percentage of our basketball players, and although the accuracy of the results did not improve as much as when we used Twitter data it did improve more than using any other non-social data.

As mentioned, empirical data from early experiments show that performance data is a good indicator of retirement. Future work includes quantifying these results, and also investigating if a strong social network helps to delay retirement. An interesting question to investigate is whether the inclusion of external social networks, for example between college and professional level coaches, would impact the results. Finally, we hope to extend
our results to other types of professional groups including groups that make up academic networks and industry networks to see if the impact of social networks is the same.
Chapter 6. User Role Identification in Software Vulnerability Discussion Over Social Networks

6.1 Introduction

A recent study [85] tested commercial off-the-shelf software applications that contain open-source components. Of those products, they found that 100% contained software vulnerabilities. More concerning, 85% of the products contained critical vulnerabilities. Similarly, the 2021 Open Source Security and Risk Analysis Report (OSSRA) [86] found that the average number of open-source components per application has increased 259% since 2016. Approximately 85% of codebases have at least one vulnerability, while the average codebase has 158 vulnerabilities. Finally, the report indicated an increase of 11% of high-risk vulnerabilities in the codebases. These security reports convey the susceptibility of many of our daily online activities to cyber attacks. Another cybersecurity statistics report [87] found that malware infections have increased by over 100 million per year for the past seven years, and ransomware attacks increased by 350% in 2018.

Social networks have been identified as a key source of information relevant to national security with regards to understanding cyber attacks and vulnerabilities [88]. Another recent study [89] evaluated social network discussions of Common Vulnerabilities and Exposures (CVEs) and concluded that social networks can provide useful insight about the spread of information, as well as monitoring vulnerabilities before they are officially announced.

Expanding upon the current literature, we apply role identification methods to social network discussions of vulnerabilities. The study of roles has been applied to different networks [14], but to our best knowledge, it has not been applied to social cyber discussions. Roles, the function a user plays in a network, can be used to describe networked entities and evolution of dynamic networks. Role-based insights can help analyze the extent of software
vulnerabilities before they are announced and develop proactive mitigation strategies. We use non-negative matrix factorization (NMF) methods as described in [21] and [25] to identify user roles.

This research was done at Pacific Northwest National Laboratory with Daniel Fortin, Samrat Chatterjee, Dennis Thomas, and Lisa Newburn and was presented at the 2021 IEEE International Conference on Intelligence and Security Informatics (ISI) [90].

6.2 Data

6.2.1 Common Software Vulnerability and Exposures Dataset. The National Vulnerability Database [91] produces a list of over 100,000 software vulnerabilities with information about each vulnerability. Included in the information is a unique Cyber Vulnerability Exploit (CVE) identifier, the published date, a description of the vulnerability, an exploitability score, and a base score. For this research, we used the version 3 exploitability scores for vulnerabilities—a number between 0 and 10—where greater value indicates ease of exploit.

6.2.2 Social Network Data. To create our networks, we used the the publicly available Twitter dataset published in [89]. This dataset contains a collection of anonymous Tweets posted between 2012 and 2018 that reference software vulnerabilities. Due to lower frequency of posts in earlier years, we focused on Tweets between March 2016 and June 2018. Each data entry included the type of action, one of retweet, quote or reply, and also a unique ID, time of post, a unique user ID for the poster, and the CVE identifier that applies to the Tweet. Additionally, the unique ID of the parent Tweet and the root Tweet were identified. The parent ID is a unique identifier of the Tweet that is acted upon while the root ID references the original Tweet. In one case, a Tweet that is retweeted, replied to, or quoted is both the parent and the root Tweet. In another case, a Tweet that is retweeted and then that retweet is retweeted is only the root Tweet and the first retweet is only a parent Tweet.
6.2.3 Network Creation. We created a dynamic network from the Twitter data by splitting the data by month resulting in 28 time periods. There was a directed edge from a user (node) to another if the first user acted upon the second user’s Tweets. Edge weights were calculated as the frequency of any action between two users. Several node attributes were included in the networks. The exploitability score of a user is the average exploitability score of all of the users’ actions. For example, if a user retweeted about a 3.9 CVE and a 1.8 CVE, as well as quoted a 2.8 CVE, their exploitability score would be 2.83. We also calculated the number of root Tweets, the number of retweets, the number of quotes and replies, and the number of all downstream actions for each user in each time period.

6.3 Methodology

The main approach we used is to apply non-negative matrix factorization (NMF) to matrices created with features from the networks, that can then be used to describe roles. These features include network topology-based metrics and CVE exploitability scores. For each time period, we considered the largest connected component and calculated the following topological metrics [6]: in-degree, out-degree, betweenness centrality, closeness centrality, eigenvector centrality, voter-rank, hub and authority. Using these metrics, along with the CVE-based node attributes described earlier, we created a feature matrix, an $n \times m$ matrix $M$ where $n$ is the number of features, and $m$ is the number of nodes in the connected component. The $ij^{th}$ entry of the matrix is the value of the $i^{th}$ feature of node $j$. So each row represents a feature and each column a node in the network. Each row is normalized to be between 0 and 1.

Since the feature matrix is non-negative, we can factor it into two non-negative matrices $B$ and $C$ using NMF such that $M \approx BC$. In most cases, a rank $r$ is required where $B$ is of size $m \times r$ and $C$ is of size $r \times n$. To identify the best $r$, we ran NMF on each possible rank and calculated the root mean squared error of the reconstruction matrix $BC$. We chose the rank where the error dropped, creating an elbow as can be seen in Figure 6.1, which displays
the root mean squared error for several time periods.

![Graph](image.png)

Figure 6.1: The root mean squared error for several time periods. We can visually locate the arms. For snapshot 27 (brown), the arm is at 4, while for snapshot 0 (blue), the arm is at 5.

This process was repeated ten times for each time period and then averaged to identify the ideal arm for each period, as discussed in [8]. We then chose a rank, 6, for all time periods by taking the ceiling of the average of all time periods. Using rank 6, we performed NMF on all time periods, calculating the basis and coefficient matrices for each. This was repeated ten times. We note that NMF does not guarantee an ordering of the vectors in $B$ and $C$. To account for this, for each time period, we reordered the vectors in each basis matrix based off of the vectors in the initial run using the Jonker-Volgenant algorithm [18], which solves the linear assignment problem described in Section 2.8.1. Once the basis matrices had a consistent vector ordering, the ten matrices were averaged to get a final basis for each time period.

Since the set of final basis matrices did not have the same distribution, the next step involved identifying a consistent and reliable basis across all time periods (as in [8]). We defined two basis matrices to be reliable if their cosine similarity was over 0.75. Starting
with the first time period, we matched the basis vectors of the next five basis matrices using
the Jonker-Volgenant algorithm to the first time period and determined which matrices were
reliable with the first basis matrix. We proceeded in this manner, incrementing by one time
period until we found a basis matrix that was reliable with all five subsequent basis matrices.

This basis matrix was termed the first reference basis. We then compared every other
basis matrix to the first reference basis, again using Jonker-Volgenant and the cosine sim-
ilarity to determine if the pair was reliable. All basis matrices that were reliable with the
first reference basis, along with the first reference basis, were averaged to create our final
basis matrix.

6.4 Results

The consistent basis identified six roles, based on the features. The method above returns
a histogram of features for each role (see Figure 6.2). The larger a feature is in a role’s
histogram, the more prevalent that feature is in the role. We named each role based on the
feature distribution and subject matter expertise. For example, exploitability is very high
in Roles 1 and 5. However, the exploitability, in-degree and closeness are all higher in Role
1, while in-degree and closeness are almost 0 in Role 5. This means that users in Role 1 are
posting original content as well as reposting about the exploits with the highest exploitability
score. Users in Role 5 are mainly reposting about highly exploitable vulnerabilities. Since
out-degree is associated more with spreading, we called Role 5 Exploit Spreaders, people
who spread vulnerable exploits, and Role 1 is comprised of people who post about critical
exploits, hence the name.

Role 0 includes spreaders, people who are influential in spreading information. Bots/Au-
thorities are in role 2. This role comprises bots, automated Twitter accounts that post every
time a new vulnerability is named, and people who function similarly. They post about
every vulnerability, which is why the exploitability is low, and they have a high number of
root posts because they only post original content. The debaters, people who quote and
Figure 6.2: The six Twitter roles described by the features used.
reply to tweets and make comments, are in role 3. Role 4 is the hubs, people who point to authority figures inside the network. These people repost from authorities frequently.

We also investigated how the roles change in the dynamic network. Figure 6.3 shows the percentage of each role in each time period. We observe that the Exploit Spreader role is most common, followed by the Hubs.

![Figure 6.3: Percentage of each role over time.](image)

Probabilities of transition between roles is presented in Figure 6.4. A lighter color indicates a high probability of transitioning. The Exploit Spreaders is the most populous role, and we observe that it has the highest transition probabilities, as shown in the rightmost column.
Figure 6.4: Probability of transitioning from a role (row) to a role (column).
6.5 CONCLUSION

While merely identifying roles may have limited uses in security, there are several promising areas of further research that could expand our role identification approach to identify proactive vulnerability mitigation strategies. One area is to look further at the dynamics of roles as the networks change. An aspect of this is to expand the method into tensors, performing NMF on all time periods at once. Another area of is to investigate why the roles generated from this process are always the same, even though NMF does not guarantee it. It would also be interesting to understand why NMF creates roles with skewed distributions and identify how this impacts the roles it produces. Answering such questions would help us better understand NMF and the roles it creates.
Chapter 7. Predicting Roles in Dynamic Social Networks

In Chapter 6, we looked at identifying consistent, meaningful roles in a dynamic network. As mentioned in Section 6.5, merely identifying the roles is not as useful as being able to make predictions about the roles. Some recent work has focused on predicting how the distributions of roles will change [25]. Other work has focused on predicting the actual roles [24]. However, little work has been done in predicting how roles will change over time and when nodes will change roles. This chapter will look at applying several time series algorithms to try to predict what role a node will have in a future time period.

7.1 Data

The data used in all of the algorithms is political data from the US Congress. It contains the representatives and senators for each US Congress (except the 109th), their state, political affiliation, and how they voted on each bill. A network for each chamber, the House of Representatives and the Senate, was created in the following manner, based on [25] for all 117 Congresses, each of which lasts two years. There is an edge between two Congress members if they both voted on at least one bill together during that Congress. Since most Congress members vote on a majority of the bills, these networks are almost complete. The edges are weighted with the similarity of votes. So if two members both voted on twenty bills, and they voted the same way on fifteen of them, the edge between them would have a weighted edge of .75. Since some Congresspeople only stay for a term or two, I created continuity by replacing the members by their seat number. So when a Senator is replaced, the new Senator is identified by the same seat number. That way, we get much longer sequences.

Using the non-negative matrix factorization (NMF) method in Section 6.3, I identified persistent roles for both the House of Representatives and the Senate using a rank of 4. The
roles, shown in Figure 7.1 and Figure 7.2, are the same for both chambers. The typical congressperson votes with their party and their state, rarely votes with the other party, and has lots of connections. The state loyalist votes with other congresspeople from their own state, regardless of parties. Collaborators vote against their party and have high centrality, meaning they are highly connected to congresspeople both in and out of their own party. Finally, party loyalists vote mostly with their own party.

Figure 7.1: The 4 roles identified for the House of Representatives.

With the basis matrices, the coefficient matrices were calculated for each Congress using linear programming. The coefficient matrices relate the roles to the Congresspeople: each column is a probability vector of the Congressperson’s roles. Combining the roles in each time period, sequences of roles were created for each person and each seat number, which were then applied to time series algorithms for predictive purposes. We also calculated the distributions of each role for each Congress, as can be seen in Figures 7.3 and 7.4.

Figure 7.2: The 4 roles identified for the Senate.
Figure 7.3: The distributions of the 4 roles identified for the House of Representatives.

Figure 7.4: The distributions of the 4 roles identified for the Senate.
7.2 ARIMA

The first approach was to predict the distribution of roles using ARIMA. This was done in python using Sklearn and statsmodel. We created a multivariate time series where each individual time series was the number of each role over the total number of Congresspeople for each time period. In order to apply ARIMA, each pair of series must be independent, or stationary. The Johanson cointegration test was applied to the four time series to check for cointegration, which tests if the difference between the means are constant. It does this by performing eigenvalue decomposition on the matrix of time series and sequentially testing if the rank is equal to 0, 1, ...n − 1. A null hypothesis of 0 rank indicates no cointegration while a a rank of i means that there is probably a cointegration relationship between i time series, and that a linear combination of i series is needed to form a stationary series.

<table>
<thead>
<tr>
<th>r ≤</th>
<th>Test Stat</th>
<th>C(95%)</th>
<th>Significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50.86</td>
<td>47.8545</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>31.18</td>
<td>29.7961</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>16.09</td>
<td>15.4943</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>5.81</td>
<td>3.8415</td>
<td>True</td>
</tr>
</tbody>
</table>

Table 7.1: Results of Johanson cointegration test on House of Representative Roles. The first column indicates the rank hypothesis. The Test Stat is the result of the test. The next column gives the critical value for a 95% confidence interval, and the last columns displays whether to accept (False) or reject (True) the null hypothesis r in the first column. In this case, the null hypothesis is rejected in all cases, meaning the rank of the matrix is probably 4.

The results of the Johanson cointegration test on the House of Representative role time series are shown in Table 7.1. It shows that the four time series are linearly related, and that to use ARIMA, we have to find the coefficients of the linear combination. This is done with the eigenvector associated with the largest eigenvalue calculated during the Johanson cointegration test. For the House data, the associated eigenvector is [19.67630667, 15.09162741, 10.28237144, 5.80904885]. Now that the data is stationary, ARIMA can be applied. The results are shown in Figure 7.5, with a forecast of fifteen time periods. Unfortunately, the predictions appear to take a near constant value close to the mean of
each distribution. Trying different parameters and manipulating the data did not yield any better results.

Figure 7.5: Prediction of the distribution of the four roles in the House of Representatives. The top graph represents the Typical roles, and the second represents State Loyalist. The third graph represents the Collaborator. The bottom graph represents Party Loyalist.

7.3 LSTM

Another approach was to create sequences of roles based on a single seat in a Congress. For example, the roles of the first senator from Alabama were combined, regardless if the senator changed. This resulted in 18284 sequences for the House of Representatives of with at least three entries. Because of population changes and territories becoming states, the number of seats for each state changed, so later states are included in fewer time periods overall.
Neural networks have become successful at sequential learning in applications like music generation and speech prediction, so I hypothesized that a long short term memory neural network (LSTM) might be more successful at predicting roles. Using pytorch and the seat role sequences, I created an LSTM model with 3 linear layers, 2 dropout layers for regularization, and 3 batch normalization layers. The model had 340239 parameters. I used cross entropy loss and Adam for the optimizer, with a learning rate of .001 and trained for 20 epochs. However, the training loss did not reduce significantly.

I tried different variations, such as changing the sequence length, changing the number of parameters by modifying the linear layers, changing the optimizer to RMSprop, and changing the learning rate. Since training loss did not decrease, hyperparameter searching was difficult, and did not generate a usable model. After investigation, the model usually predicted only one role, the one with the greatest distribution. Since the data was so imbalanced, as seen in Figures 7.3 and 7.4, and with a small number of possible outcomes, I theorized that LSTMs might not be effective at predicting roles with the data given.

7.4 Predicting Roles Using LSTM-Node-to-vec

After trying an LSTM on roles identified by NMF, I switched approaches and tried applying a graph neural network, GNN, to identify the roles. There is some precedent for this approach, as described by in Grover and Leskovec in [92]. They created an algorithm called Node2vec that applies a skip-gram to networks. The goal of a skip-gram is to learn word associations while the goal of node2vec is to learn node representations. The main difficulty in applying a skip-gram to a network is that the structure of sentences is much different than that of a network. Grover and Leskovec adapted the model by applying second order random walk sampling to create the needed sequences.

Node2vec has become a popular approach for classifying nodes and link prediction with the algorithm being included in standard graph neural network programming packages. However, it does not address dynamic networks. A modification, LSTM-node2vec, [93] applies
node2vec to dynamic networks. While the results in the paper were more accurate than other current techniques (DeepWalk, node2vec, dyngraph2vec), the algorithm was not well documented and no code was provided. My approach to the problem of using unsupervised learning to identify roles was to implement a variation of LSTM-node2vec and then apply K-means clustering to group the nodes and logistic regression to classify the nodes based off of the previously identified roles.

7.4.1 Method. The first step was creating the random walks. In node2vec, a second order random walk is used. However, I created a temporal random walk where at each time stamp, a neighbor of \( n \) is randomly sampled according to the following distribution:

\[
P(c_t = x) = \begin{cases} 
\frac{w(v,x)}{Z} & \text{if } (v,x) \in E \\
0 & \text{else}
\end{cases}
\]

Here \( c_t \) is the \( t^{th} \) node in the random walk, \( w(v,x) \) is the edge weight between nodes \( v \) and \( x \) and \( Z \) the sum of the weights of edges connecting to \( z \),

\[
Z = \sum_x w(v,x)
\]

For example, using the three graphs in Figure 7.6, some random walks for node 4 may be (6, 6, 5), (1, 2, 1), (5, 3, 8).

![Figure 7.6: Three time periods of a dynamic network.](image)

Using alias sampling [94], which turns a discrete sampling into uniform distribution over
binary outcomes, the transition probabilities for each snapshot can be computed before creating the random walks, speeding up the computation time. The random walks are created by iterating through each time period. For each node in the $t$th graph $G_t$, there are $k$ random walks created from previous snapshots $G_{t-L}, G_{t-L+1}, \ldots, G_{t-1}$, sampled in the above manner. Each random walk has length between $[2, L]$, where $L$ is the maximum random walk length.

After the random walks are created, I passed them through a basic LSTM autoencoder. I used one LSTM for the encoder and one for the decoder, followed by a linear layer. Since the sequences are not the same length, they were padded. The autoencoder learns the sequences, so the output is the same as the input. However, in this case, we ignored the output and used the input weights of the encoder and decoder to initialize the weights of a skip-gram. The skip-gram then learns the dynamics of the network as well as the static information. The skip-gram is simple, consisting of matrix multiplication and the negative log likelihood loss. The weights from the skip-gram are the node representations, which we can cluster using K-means for each snapshot. The output is a role for each node. The weights from the skip-gram are used to initialize the LSTM for the next snapshot, creating a feedback loop so that past states of the network influence future states.

7.4.2 Results. To analyze the results, I ran K-means clustering with $K = 5$ on the node representations from LSTM-node2vec. I ran logistic regression for multiclass classification using the roles identified through non-negative matrix factorization to try to get a sense of how good these roles were. With 5 classes, the percent of roles that matched the NMF roles for each Congress was between $13 - 27\%$, so not very good. However, this is an arbitrary metric used for comparison, and it is possible that the clusters discovered are meaningful. Unfortunately, to our knowledge there is no method to qualify the role of the clusters, which is why NMF has been popular in role detection. For the full results, see Table 7.2. Figure 7.7 is a picture of the fifth House of Representative network clustered using KMeans.
<table>
<thead>
<tr>
<th>Congress</th>
<th>Score</th>
<th>Congress</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23.52</td>
<td>11</td>
<td>20.51</td>
</tr>
<tr>
<td>2</td>
<td>33.33</td>
<td>12</td>
<td>24.32</td>
</tr>
<tr>
<td>3</td>
<td>22.222</td>
<td>13</td>
<td>22.44</td>
</tr>
<tr>
<td>4</td>
<td>13.79</td>
<td>14</td>
<td>18.36</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>15</td>
<td>14.28</td>
</tr>
<tr>
<td>6</td>
<td>13.79</td>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>14.28</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>23.68</td>
<td>18</td>
<td>26.78</td>
</tr>
<tr>
<td>9</td>
<td>16.21</td>
<td>19</td>
<td>23.63</td>
</tr>
<tr>
<td>10</td>
<td>15.8</td>
<td>20</td>
<td>21.81</td>
</tr>
</tbody>
</table>

Table 7.2: Accuracy of predicting roles using KMeans clustering on LSTM-node2vec node representations compared with roles identified in non-negative matrix factorization.

Figure 7.7: The 5th House of Representatives network
7.5 CONCLUSION

There were a lot of difficulties trying to predict roles and role distributions. One of reasons was that the members in the data set changed so much. States were added until Hawaii joined the union in 1959, and population changes affected the number of seats for each state in the House of Representatives. The datasets were small, with less than 100 seats in the Senate and at most 435 in the House of Representatives. Using NMF to identify the roles resulted in imbalanced data, with one role comprising around 70% of all nodes. Machine learning algorithms struggle with imbalanced data, as ARIMA and the LSTM showed, and augmenting it to be realistic would have been difficult.

There were several other issues with the GNN approach. Firstly, it is computationally expensive, even with less than 2000 sequences. This may be because the padded sequences increased the data set significantly. The lack of resources prevented me from running on all of the graphs, limiting the usefulness of the model. Overall, this is a difficult problem to solve and may need new techniques or better datasets to get meaningful results.
Network theory has evolved in the past few decades to study large, complex datasets that were previously not possible without computers, which has led to interesting problems, especially concerning dynamic networks. In this work, we have discussed three such problems: predicting links via studying properties of Katz distance on simple networks, using social bipartite networks to predict group transitions in professional sports, and identifying roles in cyber social networks and attempting to predict roles in political social networks. There are several ways we can build off of this research.

There are more interesting problems in comparing Katz and effective resistance on simple graphs. Proving the convergence of the Katz matrix on the fork and cycle could lead to future insights on the Katz distance. One could look at more complicated graphs: different structures, directed graphs, and weighted graphs. Or one could compare Katz distance to other similarity measures.

If it were possible to get the data, it would be nice to see if the same trend occurs with social data versus performance data in a professional setting. However, this problem is really about predicting groups in bipartite networks, so this could be extended to any bipartite graph, like citation or coauthor networks. Potentially, the groups could even be hidden, like roles, making this a role detection problem. Similarly, one could look at community detection on bipartite graphs using machine learning techniques, especially dimension-reducing algorithms.

Looking at roles in dynamic networks is especially interesting. There is not a lot of work using neural networks to predict roles and this seems like an area that could have lots of success. One of the main issues with it is that the roles are not described, so there would need to be some method of doing this. Another aspect would be to consider why NMF creates unbalanced roles and if there are ways to compensate for this. NMF does not guarantee that the same roles exist when the number of roles are increased. However, this appears to be
the case in practice, so it would be nice to know why.
The appendix includes informative data that extends the data presented in the main body of the work. It includes a complete summary information for all of the machine learning algorithms utilized, and all of the combinations of features. It also includes the tables of the most socially active players in both baseball and basketball for all of the centralities we consider.

<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>4.2%</td>
<td>4.2%</td>
<td>4.4%</td>
<td>4.4%</td>
<td>3.9%</td>
<td>4.5%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>17.1%</td>
<td>15.4%</td>
<td>17.1%</td>
<td>18.7%</td>
<td>7.7%</td>
<td>13.1%</td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>3.8%</td>
<td>3.0%</td>
<td>3.2%</td>
<td>3.1%</td>
<td>3.7%</td>
<td>3.3%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>17.4%</td>
<td>16.7%</td>
<td>17.9%</td>
<td>21.8%</td>
<td>5.5%</td>
<td>13.7%</td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>5.1%</td>
<td>4.7%</td>
<td>5.1%</td>
<td>5.0%</td>
<td>4.2%</td>
<td>5.3%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>16.7%</td>
<td>16.7%</td>
<td>17.2%</td>
<td>18.9%</td>
<td>9.0%</td>
<td>14.2%</td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>4.0%</td>
<td>3.7%</td>
<td>3.2%</td>
<td>3.6%</td>
<td>3.3%</td>
<td>3.2%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>16.7%</td>
<td>15.1%</td>
<td>16.6%</td>
<td>18.1%</td>
<td>8.2%</td>
<td>13.4%</td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>4.4%</td>
<td>4.1%</td>
<td>3.9%</td>
<td>3.9%</td>
<td>4.0%</td>
<td>3.7%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>17.2%</td>
<td>16.0%</td>
<td>17.7%</td>
<td>19.0%</td>
<td>9.0%</td>
<td>14.1%</td>
</tr>
<tr>
<td>Value Only</td>
<td>N</td>
<td>3.8%</td>
<td>3.5%</td>
<td>4.1%</td>
<td>4.3%</td>
<td>4.0%</td>
<td>4.0%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>17.1%</td>
<td>16.1%</td>
<td>17.0%</td>
<td>18.4%</td>
<td>8.2%</td>
<td>14.0%</td>
</tr>
<tr>
<td>Social Only</td>
<td>Y</td>
<td>17.4%</td>
<td>16.0%</td>
<td>17.4%</td>
<td>18.7%</td>
<td>9.0%</td>
<td>14.0%</td>
</tr>
<tr>
<td>All data</td>
<td>N</td>
<td>4.7%</td>
<td>3.9%</td>
<td>4.1%</td>
<td>4.0%</td>
<td>3.8%</td>
<td>4.0%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>15.8%</td>
<td>15.8%</td>
<td>16.9%</td>
<td>20.4%</td>
<td>4.7%</td>
<td>13.0%</td>
</tr>
</tbody>
</table>

Table A.1: Summary of Algorithm Accuracy for Baseball data from 2001-2019. Using non-social features, the accuracy was typically slightly better than guessing, 3.44%. Using social data, accuracy increased in every case and was 15-21% using Logistic Regression, Random Forest, and XGBoost.
<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>N</td>
<td>0.01</td>
<td>0.009</td>
<td>0.01</td>
<td>0.01</td>
<td>0.008</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.157</td>
<td>0.146</td>
<td>0.151</td>
<td>0.176</td>
<td>0.070</td>
<td>0.117</td>
</tr>
<tr>
<td>Team</td>
<td>N</td>
<td>0.008</td>
<td>0.020</td>
<td>0.021</td>
<td>0.022</td>
<td>0.022</td>
<td>0.020</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.160</td>
<td>0.158</td>
<td>0.160</td>
<td>0.204</td>
<td>0.049</td>
<td>0.121</td>
</tr>
<tr>
<td>Career</td>
<td>N</td>
<td>0.016</td>
<td>0.008</td>
<td>0.025</td>
<td>0.024</td>
<td>0.021</td>
<td>0.026</td>
</tr>
<tr>
<td>Length Only</td>
<td>Y</td>
<td>0.150</td>
<td>0.155</td>
<td>0.151</td>
<td>0.178</td>
<td>0.080</td>
<td>0.125</td>
</tr>
<tr>
<td>Performance</td>
<td>N</td>
<td>0.012</td>
<td>0.017</td>
<td>0.029</td>
<td>0.030</td>
<td>0.030</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.152</td>
<td>0.141</td>
<td>0.146</td>
<td>0.170</td>
<td>0.073</td>
<td>0.118</td>
</tr>
<tr>
<td>Rank</td>
<td>N</td>
<td>0.012</td>
<td>0.007</td>
<td>0.025</td>
<td>0.024</td>
<td>0.025</td>
<td>0.023</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.159</td>
<td>0.151</td>
<td>0.157</td>
<td>0.178</td>
<td>0.082</td>
<td>0.125</td>
</tr>
<tr>
<td>Value</td>
<td>N</td>
<td>0.010</td>
<td>0.005</td>
<td>0.025</td>
<td>0.026</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.158</td>
<td>0.152</td>
<td>0.151</td>
<td>0.173</td>
<td>0.073</td>
<td>0.124</td>
</tr>
<tr>
<td>Social Only</td>
<td>Y</td>
<td>0.158</td>
<td>0.152</td>
<td>0.155</td>
<td>0.176</td>
<td>0.081</td>
<td>0.124</td>
</tr>
<tr>
<td>All data</td>
<td>N</td>
<td>0.018</td>
<td>0.033</td>
<td>0.038</td>
<td>0.033</td>
<td>0.036</td>
<td>0.037</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.14</td>
<td>0.149</td>
<td>0.146</td>
<td>0.187</td>
<td>0.042</td>
<td>0.113</td>
</tr>
</tbody>
</table>

Table A.2: Summary of Algorithm F1 for Baseball data for 2002-2019. We observe that the highest F1 score was obtained using the same model that achieved the highest accuracy.

<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>N</td>
<td>(.03, .055)</td>
<td>(.023, .057)</td>
<td>(.028, .06)</td>
<td>(.029, .059)</td>
<td>(.021, .056)</td>
<td>(.029, .06)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.142, .201)</td>
<td>(.122, .182)</td>
<td>(.147, .198)</td>
<td>(.158, .221)</td>
<td>(.052, .10)</td>
<td>(.101, .158)</td>
</tr>
<tr>
<td>Team</td>
<td>N</td>
<td>(.024, .051)</td>
<td>(.019, .042)</td>
<td>(.021, .045)</td>
<td>(.021, .043)</td>
<td>(.023, .052)</td>
<td>(.021, .046)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.137, .216)</td>
<td>(.137, .203)</td>
<td>(.152, .206)</td>
<td>(.187, .245)</td>
<td>(.035, .071)</td>
<td>(.112, .167)</td>
</tr>
<tr>
<td>Career</td>
<td>N</td>
<td>(.033, .07)</td>
<td>(.029, .064)</td>
<td>(.037, .065)</td>
<td>(.031, .066)</td>
<td>(.027, .062)</td>
<td>(.035, .068)</td>
</tr>
<tr>
<td>Length Only</td>
<td>Y</td>
<td>(.136, .206)</td>
<td>(.146, .195)</td>
<td>(.145, .198)</td>
<td>(.161, .218)</td>
<td>(.066, .112)</td>
<td>(.114, .172)</td>
</tr>
<tr>
<td>Performance</td>
<td>N</td>
<td>(.021, .054)</td>
<td>(.023, .054)</td>
<td>(.019, .043)</td>
<td>(.023, .052)</td>
<td>(.022, .05)</td>
<td>(.018, .046)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.139, .203)</td>
<td>(.125, .176)</td>
<td>(.142, .193)</td>
<td>(.147, .21)</td>
<td>(.062, .104)</td>
<td>(.108, .158)</td>
</tr>
<tr>
<td>Rank</td>
<td>N</td>
<td>(.027, .06)</td>
<td>(.024, .06)</td>
<td>(.024, .056)</td>
<td>(.027, .052)</td>
<td>(.026, .06)</td>
<td>(.024, .05)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.139, .211)</td>
<td>(.133, .189)</td>
<td>(.149, .203)</td>
<td>(.165, .214)</td>
<td>(.069, .114)</td>
<td>(.116, .168)</td>
</tr>
<tr>
<td>Value</td>
<td>N</td>
<td>(.025, .054)</td>
<td>(.021, .048)</td>
<td>(.026, .054)</td>
<td>(.024, .059)</td>
<td>(.026, .056)</td>
<td>(.028, .055)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.131, .212)</td>
<td>(.133, .184)</td>
<td>(.14, .199)</td>
<td>(.16, .21)</td>
<td>(.062, .102)</td>
<td>(.118, .165)</td>
</tr>
<tr>
<td>Social Only</td>
<td>Y</td>
<td>(.133, .217)</td>
<td>(.131, .186)</td>
<td>(.153, .203)</td>
<td>(.151, .217)</td>
<td>(.07, .113)</td>
<td>(.116, .172)</td>
</tr>
<tr>
<td>All data</td>
<td>N</td>
<td>(.031, .06)</td>
<td>(.024, .058)</td>
<td>(.026, .056)</td>
<td>(.023, .057)</td>
<td>(.025, .052)</td>
<td>(.026, .058)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.126, .191)</td>
<td>(.129, .19)</td>
<td>(.143, .195)</td>
<td>(.177, .236)</td>
<td>(.033, .064)</td>
<td>(.109, .161)</td>
</tr>
</tbody>
</table>

Table A.3: Summary of confidence intervals for Baseball data for 2002-2019 over 100 runs.
<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>N</td>
<td>5.1%</td>
<td>4.7%</td>
<td>5.5%</td>
<td>6.5%</td>
<td>3.9%</td>
<td>6.2%</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>4.4%</td>
<td>5.0%</td>
<td>4.6%</td>
<td>4.1%</td>
<td>3.8%</td>
<td>4.4%</td>
</tr>
<tr>
<td>Team</td>
<td>N</td>
<td>5.2%</td>
<td>5.3%</td>
<td>5.2%</td>
<td>5.6%</td>
<td>5.1%</td>
<td>5.7%</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>5.2%</td>
<td>5.2%</td>
<td>5.4%</td>
<td>3.8%</td>
<td>4.9%</td>
<td>4.7%</td>
</tr>
<tr>
<td>Career</td>
<td>N</td>
<td>6.4%</td>
<td>3.5%</td>
<td>5.6%</td>
<td>6.7%</td>
<td>5.2%</td>
<td>5.1%</td>
</tr>
<tr>
<td>Length Only</td>
<td>Y</td>
<td>4.6%</td>
<td>4.6%</td>
<td>4.6%</td>
<td>3.5%</td>
<td>3.6%</td>
<td>4.3%</td>
</tr>
<tr>
<td>Performance</td>
<td>N</td>
<td>3.5%</td>
<td>2.9%</td>
<td>4.0%</td>
<td>4.0%</td>
<td>5.0%</td>
<td>4.0%</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>4.4%</td>
<td>4.7%</td>
<td>4.7%</td>
<td>3.7%</td>
<td>3.1%</td>
<td>4.4%</td>
</tr>
<tr>
<td>Rank</td>
<td>N</td>
<td>5.0%</td>
<td>3.1%</td>
<td>5.9%</td>
<td>6.2%</td>
<td>5.1%</td>
<td>5.9%</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>4.5%</td>
<td>3.9%</td>
<td>5.0%</td>
<td>4.1%</td>
<td>4.0%</td>
<td>4.7%</td>
</tr>
<tr>
<td>Value</td>
<td>N</td>
<td>4.5%</td>
<td>2.8%</td>
<td>5.6%</td>
<td>4.9%</td>
<td>5.2%</td>
<td>5.3%</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>5.2%</td>
<td>4.7%</td>
<td>4.6%</td>
<td>4.0%</td>
<td>3.2%</td>
<td>4.9%</td>
</tr>
<tr>
<td>Social Only</td>
<td>Y</td>
<td>4.9%</td>
<td>5.0%</td>
<td>4.6%</td>
<td>3.9%</td>
<td>3.6%</td>
<td>5.1%</td>
</tr>
<tr>
<td>All data</td>
<td>N</td>
<td>5.5%</td>
<td>4.2%</td>
<td>4.7%</td>
<td>4.1%</td>
<td>4.4%</td>
<td>5.4%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>5.2%</td>
<td>4.0%</td>
<td>5.1%</td>
<td>3.9%</td>
<td>5.4%</td>
<td>4.4%</td>
</tr>
</tbody>
</table>

Table A.4: Summary of Algorithm Accuracy for Baseball data from 2020.

<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>N</td>
<td>0.015</td>
<td>0.015</td>
<td>0.017</td>
<td>0.019</td>
<td>0.012</td>
<td>0.019</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>0.014</td>
<td>0.029</td>
<td>0.02</td>
<td>0.023</td>
<td>0.021</td>
<td>0.021</td>
</tr>
<tr>
<td>Team</td>
<td>N</td>
<td>0.009</td>
<td>0.032</td>
<td>0.031</td>
<td>0.007</td>
<td>0.029</td>
<td>0.032</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>0.014</td>
<td>0.031</td>
<td>0.025</td>
<td>0.019</td>
<td>0.024</td>
<td>0.026</td>
</tr>
<tr>
<td>Career</td>
<td>N</td>
<td>0.017</td>
<td>0.006</td>
<td>0.024</td>
<td>0.030</td>
<td>0.023</td>
<td>0.022</td>
</tr>
<tr>
<td>Length Only</td>
<td>Y</td>
<td>0.014</td>
<td>0.026</td>
<td>0.020</td>
<td>0.017</td>
<td>0.023</td>
<td>0.021</td>
</tr>
<tr>
<td>Performance</td>
<td>N</td>
<td>0.014</td>
<td>0.013</td>
<td>0.026</td>
<td>0.024</td>
<td>0.026</td>
<td>0.027</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>0.013</td>
<td>0.028</td>
<td>0.022</td>
<td>0.020</td>
<td>0.018</td>
<td>0.022</td>
</tr>
<tr>
<td>Rank</td>
<td>N</td>
<td>0.016</td>
<td>0.009</td>
<td>0.037</td>
<td>0.035</td>
<td>0.028</td>
<td>0.035</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>0.013</td>
<td>0.024</td>
<td>0.021</td>
<td>0.023</td>
<td>0.024</td>
<td>0.023</td>
</tr>
<tr>
<td>Value</td>
<td>N</td>
<td>0.009</td>
<td>0.006</td>
<td>0.033</td>
<td>0.030</td>
<td>0.029</td>
<td>0.030</td>
</tr>
<tr>
<td>Only</td>
<td>Y</td>
<td>0.018</td>
<td>0.028</td>
<td>0.02</td>
<td>0.022</td>
<td>0.019</td>
<td>0.126</td>
</tr>
<tr>
<td>Social Only</td>
<td>Y</td>
<td>0.016</td>
<td>0.028</td>
<td>0.22</td>
<td>0.021</td>
<td>0.021</td>
<td>0.026</td>
</tr>
<tr>
<td>All data</td>
<td>N</td>
<td>0.019</td>
<td>0.024</td>
<td>0.029</td>
<td>0.025</td>
<td>0.026</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.015</td>
<td>0.024</td>
<td>0.026</td>
<td>0.023</td>
<td>0.030</td>
<td>0.023</td>
</tr>
</tbody>
</table>

Table A.5: Summary of Algorithm F1 for Baseball data for 2020.
<table>
<thead>
<tr>
<th>Features</th>
<th>Social Data</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
<td>(.0,.0118)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
<td>(.0,.104)</td>
<td>(.0,.088)</td>
<td>(.0,.088)</td>
<td>(.0,.133)</td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.133)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.133)</td>
<td>(.0,.118)</td>
<td>(.0,.133)</td>
<td>(.0,.088)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>(.0,.147)</td>
<td>(.0,.118)</td>
<td>(.0,.147)</td>
<td>(.0,.147)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.104)</td>
<td>(.0,.088)</td>
<td>(.0,.104)</td>
<td>(.0,.104)</td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>(.0,.104)</td>
<td>(.0,.088)</td>
<td>(.0,.088)</td>
<td>(.0,.104)</td>
<td>(.0,.118)</td>
<td>(.0,.104)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.133)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.104)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.133)</td>
<td>(.0,.133)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.133)</td>
<td>(.0,.088)</td>
<td>(.0,.133)</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td>Value Only</td>
<td>N</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.088)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td>Social Only</td>
<td>N</td>
<td>(.0,.118)</td>
<td>(.0,.147)</td>
<td>(.0,.118)</td>
<td>(.0,.104)</td>
<td>(.0,.104)</td>
<td>(.0,.118)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.118)</td>
<td>(.0,.088)</td>
<td>(.0,.133)</td>
<td>(.0,.118)</td>
</tr>
</tbody>
</table>

Table A.6: Summary of confidence intervals for Baseball data for 2020 over 100 runs.
**Table A.7: Summary of algorithm accuracy for basketball data for 2001-2019.**

<table>
<thead>
<tr>
<th>Data Used</th>
<th>Twitter</th>
<th>College</th>
<th>ADA</th>
<th>Logistic</th>
<th>Random</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra</th>
<th>Regression</th>
<th>Forest</th>
<th>Trees</th>
<th>Regression</th>
<th>Forest</th>
<th>Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positions</strong></td>
<td>N</td>
<td>N</td>
<td>4.1%</td>
<td>3.4%</td>
<td>4.0%</td>
<td>4.1%</td>
<td>3.4%</td>
<td>3.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.2%</td>
<td>3.6%</td>
<td>10.9%</td>
<td>8.1%</td>
<td>10.7%</td>
<td>10.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.3%</td>
<td>19.6%</td>
<td>29.4%</td>
<td>23.7%</td>
<td>23.0%</td>
<td>29.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>4.4%</td>
<td>19.4%</td>
<td>29.9%</td>
<td>23.7%</td>
<td>23.4%</td>
<td>29.8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Team</strong></td>
<td>N</td>
<td>N</td>
<td>3.9%</td>
<td>4.0%</td>
<td>4.2%</td>
<td>3.9%</td>
<td>3.6%</td>
<td>4.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>3.8%</td>
<td>3.9%</td>
<td>9.0%</td>
<td>6.0%</td>
<td>8.5%</td>
<td>8.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.9%</td>
<td>18.1%</td>
<td>29%</td>
<td>21.9%</td>
<td>10.2%</td>
<td>27.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.9%</td>
<td>17.8%</td>
<td>29.6%</td>
<td>22.2%</td>
<td>10.8%</td>
<td>28.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Career Length Only</strong></td>
<td>N</td>
<td>N</td>
<td>4.5%</td>
<td>4.4%</td>
<td>4.7%</td>
<td>4.7%</td>
<td>4.0%</td>
<td>5.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.5%</td>
<td>4.3%</td>
<td>13.9%</td>
<td>8.8%</td>
<td>13.8%</td>
<td>14.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.3%</td>
<td>20.3%</td>
<td>29.5%</td>
<td>23.8%</td>
<td>24.3%</td>
<td>29.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.0%</td>
<td>20.2%</td>
<td>29.9%</td>
<td>23.9%</td>
<td>24.7%</td>
<td>29.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Performance Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.6%</td>
<td>3.4%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>3.9%</td>
<td>3.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.1%</td>
<td>3.5%</td>
<td>4.9%</td>
<td>4.9%</td>
<td>4.1%</td>
<td>6.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.0%</td>
<td>19.9%</td>
<td>28.7%</td>
<td>20.7%</td>
<td>21.9%</td>
<td>29.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.0%</td>
<td>19.9%</td>
<td>28.7%</td>
<td>20.7%</td>
<td>21.9%</td>
<td>29.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rank Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.9%</td>
<td>3.0%</td>
<td>3.1%</td>
<td>3.3%</td>
<td>3.5%</td>
<td>3.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.0%</td>
<td>3.3%</td>
<td>4.3%</td>
<td>5.7%</td>
<td>4.2%</td>
<td>4.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.9%</td>
<td>19.8%</td>
<td>28.1%</td>
<td>21.9%</td>
<td>21.9%</td>
<td>28.8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>10.0%</td>
<td>19.6%</td>
<td>28.1%</td>
<td>22.1%</td>
<td>22.7%</td>
<td>28.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Valuation Only</strong></td>
<td>N</td>
<td>N</td>
<td>4.1%</td>
<td>3.5%</td>
<td>5.1%</td>
<td>4.2%</td>
<td>4.6%</td>
<td>5.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>3.9%</td>
<td>3.4%</td>
<td>8.5%</td>
<td>6.3%</td>
<td>7.8%</td>
<td>9.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.3%</td>
<td>19.9%</td>
<td>29.2%</td>
<td>22.3%</td>
<td>23.5%</td>
<td>29.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.2%</td>
<td>19.9%</td>
<td>29.4%</td>
<td>22.4%</td>
<td>24.0%</td>
<td>29.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Twitter College All Social</strong></td>
<td>Y</td>
<td>N</td>
<td>9.2%</td>
<td>20.2%</td>
<td>29.3%</td>
<td>23.5%</td>
<td>23.5%</td>
<td>29.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.3%</td>
<td>2.9%</td>
<td>8.4%</td>
<td>7.4%</td>
<td>7.2%</td>
<td>8.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.5%</td>
<td>19.5%</td>
<td>29.6%</td>
<td>23.5%</td>
<td>24.2%</td>
<td>29.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>All Data</strong></td>
<td>N</td>
<td>N</td>
<td>4.0%</td>
<td>4.3%</td>
<td>8.7%</td>
<td>6.2%</td>
<td>7.0%</td>
<td>8.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.0%</td>
<td>4.2%</td>
<td>8.7%</td>
<td>6.2%</td>
<td>7.1%</td>
<td>8.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>9.0%</td>
<td>17.9%</td>
<td>27.3%</td>
<td>19.6%</td>
<td>11.1%</td>
<td>28.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>9.2%</td>
<td>17.5%</td>
<td>27.6%</td>
<td>20.3%</td>
<td>11.3%</td>
<td>28.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Used</td>
<td>Twitter</td>
<td>College</td>
<td>ADA</td>
<td>Logistic</td>
<td>Random</td>
<td>XGBoost</td>
<td>KNN</td>
<td>Extra Trees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---------</td>
<td>-----</td>
<td>----------</td>
<td>--------</td>
<td>---------</td>
<td>-----</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Regression</td>
<td>Forest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>N</td>
<td>0.008</td>
<td>0.011</td>
<td>0.008</td>
<td>0.008</td>
<td>0.001</td>
<td>0.008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.012</td>
<td>0.017</td>
<td>0.103</td>
<td>0.072</td>
<td>0.103</td>
<td>0.104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.081</td>
<td>0.189</td>
<td>0.302</td>
<td>0.241</td>
<td>0.232</td>
<td>0.302</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.082</td>
<td>0.189</td>
<td>0.302</td>
<td>0.241</td>
<td>0.232</td>
<td>0.302</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>N</td>
<td>0.008</td>
<td>0.029</td>
<td>0.028</td>
<td>0.027</td>
<td>0.023</td>
<td>0.027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.009</td>
<td>0.030</td>
<td>0.087</td>
<td>0.056</td>
<td>0.083</td>
<td>0.084</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.084</td>
<td>0.173</td>
<td>0.292</td>
<td>0.221</td>
<td>0.1</td>
<td>0.273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.085</td>
<td>0.169</td>
<td>0.296</td>
<td>0.224</td>
<td>0.105</td>
<td>0.284</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>N</td>
<td>0.012</td>
<td>0.008</td>
<td>0.028</td>
<td>0.027</td>
<td>0.025</td>
<td>0.031</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.016</td>
<td>0.015</td>
<td>0.134</td>
<td>0.082</td>
<td>0.134</td>
<td>0.138</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.079</td>
<td>0.198</td>
<td>0.300</td>
<td>0.245</td>
<td>0.241</td>
<td>0.301</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.077</td>
<td>0.195</td>
<td>0.305</td>
<td>0.243</td>
<td>0.247</td>
<td>0.305</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>N</td>
<td>0.009</td>
<td>0.018</td>
<td>0.034</td>
<td>0.033</td>
<td>0.036</td>
<td>0.033</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.011</td>
<td>0.020</td>
<td>0.134</td>
<td>0.082</td>
<td>0.134</td>
<td>0.138</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.081</td>
<td>0.190</td>
<td>0.287</td>
<td>0.209</td>
<td>0.212</td>
<td>0.298</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.077</td>
<td>0.192</td>
<td>0.292</td>
<td>0.21</td>
<td>0.218</td>
<td>0.299</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>N</td>
<td>0.012</td>
<td>0.005</td>
<td>0.022</td>
<td>0.024</td>
<td>0.024</td>
<td>0.024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.015</td>
<td>0.011</td>
<td>0.042</td>
<td>0.054</td>
<td>0.041</td>
<td>0.044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.088</td>
<td>0.195</td>
<td>0.286</td>
<td>0.224</td>
<td>0.218</td>
<td>0.293</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.088</td>
<td>0.192</td>
<td>0.285</td>
<td>0.226</td>
<td>0.226</td>
<td>0.294</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>N</td>
<td>0.010</td>
<td>0.006</td>
<td>0.046</td>
<td>0.035</td>
<td>0.040</td>
<td>0.045</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.012</td>
<td>0.011</td>
<td>0.082</td>
<td>0.058</td>
<td>0.075</td>
<td>0.091</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.079</td>
<td>0.193</td>
<td>0.296</td>
<td>0.228</td>
<td>0.232</td>
<td>0.298</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.078</td>
<td>0.193</td>
<td>0.298</td>
<td>0.229</td>
<td>0.239</td>
<td>0.305</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Twitter College All Social</td>
<td>N</td>
<td>N</td>
<td>0.082</td>
<td>0.194</td>
<td>0.298</td>
<td>0.242</td>
<td>0.233</td>
<td>0.297</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.010</td>
<td>0.005</td>
<td>0.074</td>
<td>0.062</td>
<td>0.066</td>
<td>0.076</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.083</td>
<td>0.192</td>
<td>0.299</td>
<td>0.239</td>
<td>0.241</td>
<td>0.301</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All Data</td>
<td>N</td>
<td>N</td>
<td>0.017</td>
<td>0.038</td>
<td>0.084</td>
<td>0.060</td>
<td>0.068</td>
<td>0.083</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.016</td>
<td>0.038</td>
<td>0.083</td>
<td>0.059</td>
<td>0.069</td>
<td>0.082</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.075</td>
<td>0.172</td>
<td>0.276</td>
<td>0.199</td>
<td>0.109</td>
<td>0.282</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.077</td>
<td>0.168</td>
<td>0.279</td>
<td>0.205</td>
<td>0.111</td>
<td>0.285</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.8: Summary of algorithm F1 score for basketball data from 2001-2019. The model with the greatest accuracy also had the highest F1 score.
<table>
<thead>
<tr>
<th>Data Used</th>
<th>Twitter</th>
<th>College</th>
<th>ADA</th>
<th>Logistic Regression</th>
<th>Random Forest</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>N</td>
<td>(.028, .058)</td>
<td>(.021, .051)</td>
<td>(.027, .054)</td>
<td>(.026, .053)</td>
<td>(.019, .048)</td>
<td>(.026, .053)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.026, .061)</td>
<td>(.023, .049)</td>
<td>(.088, .129)</td>
<td>(.061, .099)</td>
<td>(.089, .131)</td>
<td>(.088, .129)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.068, .124)</td>
<td>(.169, .224)</td>
<td>(.265, .332)</td>
<td>(.215, .261)</td>
<td>(.204, .258)</td>
<td>(.256, .315)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.007, .118)</td>
<td>(.168, .221)</td>
<td>(.269, .330)</td>
<td>(.21, .266)</td>
<td>(.203, .266)</td>
<td>(.267, .33)</td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>N</td>
<td>(.026, .053)</td>
<td>(.028, .053)</td>
<td>(.03, .057)</td>
<td>(.028, .052)</td>
<td>(.025, .048)</td>
<td>(.027, .052)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.026, .052)</td>
<td>(.026, .054)</td>
<td>(.069, .109)</td>
<td>(.044, .075)</td>
<td>(.068, .107)</td>
<td>(.068, .106)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.067, .125)</td>
<td>(.155, .216)</td>
<td>(.262, .321)</td>
<td>(.189, .242)</td>
<td>(.081, .126)</td>
<td>(.242, .302)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.065, .132)</td>
<td>(.153, .20)</td>
<td>(.269, .324)</td>
<td>(.187, .253)</td>
<td>(.09, .129)</td>
<td>(.251, .314)</td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>N</td>
<td>(.03, .059)</td>
<td>(.027, .057)</td>
<td>(.033, .059)</td>
<td>(.033, .061)</td>
<td>(.025, .058)</td>
<td>(.037, .061)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.031, .059)</td>
<td>(.028, .058)</td>
<td>(.115, .168)</td>
<td>(.066, .109)</td>
<td>(.113, .16)</td>
<td>(.119, .167)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.071, .121)</td>
<td>(.178, .229)</td>
<td>(.262, .328)</td>
<td>(.205, .269)</td>
<td>(.215, .266)</td>
<td>(.265, .336)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.061, .118)</td>
<td>(.175, .232)</td>
<td>(.268, .329)</td>
<td>(.21, .271)</td>
<td>(.218, .277)</td>
<td>(27, .33)</td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>N</td>
<td>(.023, .051)</td>
<td>(.021, .047)</td>
<td>(.025, .053)</td>
<td>(.025, .049)</td>
<td>(.029, .052)</td>
<td>(.023, .051)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.026, .057)</td>
<td>(.023, .049)</td>
<td>(.031, .067)</td>
<td>(.039, .061)</td>
<td>(.027, .053)</td>
<td>(.041, .084)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.064, .128)</td>
<td>(.164, .226)</td>
<td>(.248, .313)</td>
<td>(.177, .233)</td>
<td>(.182, .24)</td>
<td>(.26, .332)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.064, .117)</td>
<td>(.162, .225)</td>
<td>(.258, .32)</td>
<td>(.179, .234)</td>
<td>(.189, .247)</td>
<td>(.263, .332)</td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>N</td>
<td>(.026, .054)</td>
<td>(.015, .044)</td>
<td>(.02, .041)</td>
<td>(.023, .0444)</td>
<td>(.024, .052)</td>
<td>(.02, .047)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.027, .055)</td>
<td>(.02, .047)</td>
<td>(.029, .054)</td>
<td>(.04, .074)</td>
<td>(.028, .061)</td>
<td>(.032, .061)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.074, .125)</td>
<td>(.165, .224)</td>
<td>(.252, .31)</td>
<td>(.196, .246)</td>
<td>(.192, .245)</td>
<td>(.259, .312)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.068, .133)</td>
<td>(.172, .22)</td>
<td>(.245, .313)</td>
<td>(.184, .251)</td>
<td>(.197, .256)</td>
<td>(.261, .318)</td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>N</td>
<td>(.028, .057)</td>
<td>(.023, .048)</td>
<td>(.038, .063)</td>
<td>(.028, .054)</td>
<td>(.033, .06)</td>
<td>(.037, .064)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.027, .054)</td>
<td>(.024, .046)</td>
<td>(.069, .102)</td>
<td>(.041, .083)</td>
<td>(.06, .096)</td>
<td>(.074, .111)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.066, .115)</td>
<td>(.173, .227)</td>
<td>(.26, .315)</td>
<td>(.196, .249)</td>
<td>(.201, .264)</td>
<td>(.263, .325)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.067, .122)</td>
<td>(.176, .226)</td>
<td>(.267, .33)</td>
<td>(.196, .254)</td>
<td>(.215, .269)</td>
<td>(.272, .33)</td>
</tr>
<tr>
<td>Twitter College All Social</td>
<td>N</td>
<td>N</td>
<td>(.071, .118)</td>
<td>(.171, .229)</td>
<td>(.253, .323)</td>
<td>(.209, .261)</td>
<td>(.264, .266)</td>
<td>(.265, .323)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.025, .057)</td>
<td>(.017, .041)</td>
<td>(.067, .104)</td>
<td>(.055, .096)</td>
<td>(.053, .089)</td>
<td>(.07, .104)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.072, .126)</td>
<td>(.167, .226)</td>
<td>(.226, .326)</td>
<td>(.203, .263)</td>
<td>(.213, .272)</td>
<td>(.267, .329)</td>
</tr>
<tr>
<td>All Data</td>
<td>N</td>
<td>N</td>
<td>(.025, .056)</td>
<td>(.03, .058)</td>
<td>(.066, .109)</td>
<td>(.042, .079)</td>
<td>(.052, .085)</td>
<td>(.068, .109)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>(.025, .057)</td>
<td>(.028, .06)</td>
<td>(.065, .105)</td>
<td>(.046, .077)</td>
<td>(.05, .089)</td>
<td>(.066, .106)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>(.062, .116)</td>
<td>(.153, .205)</td>
<td>(.243, .304)</td>
<td>(.172, .225)</td>
<td>(.093, .131)</td>
<td>(.256, .307)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>(.06, .12)</td>
<td>(.148, .202)</td>
<td>(.246, .31)</td>
<td>(.178, .232)</td>
<td>(.089, .133)</td>
<td>(.248, .321)</td>
</tr>
</tbody>
</table>

Table A.9: Summary of 95% confidence intervals for the basketball data from 2001-2019. Each algorithm was run 100 times.
<table>
<thead>
<tr>
<th>Data Used</th>
<th>Twitter</th>
<th>College</th>
<th>ADA</th>
<th>Logistic</th>
<th>Random</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Positions Only</strong></td>
<td>N</td>
<td>N</td>
<td>2.5%</td>
<td>1.9%</td>
<td>1.4%</td>
<td>2.1%</td>
<td>2.7%</td>
<td>1.7%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.7%</td>
<td>2.8%</td>
<td>14.8%</td>
<td>14.4%</td>
<td>16.2%</td>
<td>14.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>5.6%</td>
<td>16.3%</td>
<td>19.9%</td>
<td>18.8%</td>
<td>16.5%</td>
<td>19.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>5.0%</td>
<td>18.2%</td>
<td>20.5%</td>
<td>19.8%</td>
<td>17.2%</td>
<td>20.1%</td>
</tr>
<tr>
<td><strong>Team Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.5%</td>
<td>3.6%</td>
<td>3.9%</td>
<td>2.8%</td>
<td>3.1%</td>
<td>3.0%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.4%</td>
<td>3.6%</td>
<td>4.2%</td>
<td>15.2%</td>
<td>4.0%</td>
<td>4.8%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>4.6%</td>
<td>9.9%</td>
<td>20.8%</td>
<td>19.5%</td>
<td>9.5%</td>
<td>20.7%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>4.5%</td>
<td>10.0%</td>
<td>20.7%</td>
<td>20.4%</td>
<td>9.1%</td>
<td>20.2%</td>
</tr>
<tr>
<td><strong>Career Length Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.7%</td>
<td>3.3%</td>
<td>7.1%</td>
<td>4.4%</td>
<td>5.9%</td>
<td>6.0%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.6%</td>
<td>4.6%</td>
<td>21.0%</td>
<td>18.4%</td>
<td>20.5%</td>
<td>20.5%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>4.7%</td>
<td>17.1%</td>
<td>19.6%</td>
<td>20.5%</td>
<td>20.3%</td>
<td>20.4%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>5.2%</td>
<td>16.7%</td>
<td>21.2%</td>
<td>20.0%</td>
<td>20.5%</td>
<td>20.6%</td>
</tr>
<tr>
<td><strong>Performance Only</strong></td>
<td>N</td>
<td>N</td>
<td>2.4%</td>
<td>2.3%</td>
<td>2.0%</td>
<td>3.0%</td>
<td>0.9%</td>
<td>1.1%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>3.7%</td>
<td>3.9%</td>
<td>6.4%</td>
<td>6.8%</td>
<td>3.4%</td>
<td>10.5%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>4.8%</td>
<td>15.7%</td>
<td>19.1%</td>
<td>15.7%</td>
<td>19.3%</td>
<td>20.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>4.5%</td>
<td>17.2%</td>
<td>19.0%</td>
<td>18.2%</td>
<td>20.5%</td>
<td>20.9%</td>
</tr>
<tr>
<td><strong>Rank Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.0%</td>
<td>1.8%</td>
<td>3.6%</td>
<td>2.8%</td>
<td>3.0%</td>
<td>3.0%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.6%</td>
<td>2.9%</td>
<td>4.8%</td>
<td>9.1%</td>
<td>3.7%</td>
<td>6.3%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>5.0%</td>
<td>16.3%</td>
<td>21.4%</td>
<td>18.4%</td>
<td>20.7%</td>
<td>20.7%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>4.8%</td>
<td>17.0%</td>
<td>20.8%</td>
<td>18.6%</td>
<td>20.2%</td>
<td>20.5%</td>
</tr>
<tr>
<td><strong>Valuation Only</strong></td>
<td>N</td>
<td>N</td>
<td>3.4%</td>
<td>2.2%</td>
<td>3.3%</td>
<td>3.8%</td>
<td>3.0%</td>
<td>2.9%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>4.2%</td>
<td>4.5%</td>
<td>7.5%</td>
<td>11.9%</td>
<td>8.3%</td>
<td>8.8%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>5.0%</td>
<td>17.1%</td>
<td>20.6%</td>
<td>18.9%</td>
<td>20.8%</td>
<td>20.5%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>5.2%</td>
<td>17.7%</td>
<td>19.9%</td>
<td>19.0%</td>
<td>20.4%</td>
<td>20.5%</td>
</tr>
<tr>
<td><strong>Twitter College All Social</strong></td>
<td>Y</td>
<td>N</td>
<td>4.7%</td>
<td>16.6%</td>
<td>21.5%</td>
<td>19.9%</td>
<td>21.1%</td>
<td>20.4%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>5.8%</td>
<td>4.1%</td>
<td>18.3%</td>
<td>16.2%</td>
<td>18.3%</td>
<td>17.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>5.2%</td>
<td>17.5%</td>
<td>20.4%</td>
<td>19.8%</td>
<td>21.3%</td>
<td>21.2%</td>
</tr>
<tr>
<td><strong>All Data</strong></td>
<td>N</td>
<td>N</td>
<td>4.0%</td>
<td>3.1%</td>
<td>4.3%</td>
<td>6.9%</td>
<td>2.2%</td>
<td>5.3%</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>3.9%</td>
<td>2.6%</td>
<td>4.3%</td>
<td>6.3%</td>
<td>2.5%</td>
<td>4.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>4.2%</td>
<td>10.3%</td>
<td>19.0%</td>
<td>12.9%</td>
<td>7.6%</td>
<td>18.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>4.3%</td>
<td>10.5%</td>
<td>19.0%</td>
<td>14.1%</td>
<td>9.0%</td>
<td>19.3%</td>
</tr>
</tbody>
</table>

Table A.10: Summary of algorithm accuracy for 2020 basketball data.
<table>
<thead>
<tr>
<th>Data Used</th>
<th>Twitter</th>
<th>College</th>
<th>ADA</th>
<th>Logistic</th>
<th>Random</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra</th>
<th>Regression</th>
<th>Forest</th>
<th>Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>N</td>
<td>0.006</td>
<td>0.007</td>
<td>0.005</td>
<td>0.006</td>
<td>0.008</td>
<td>0.007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.022</td>
<td>0.017</td>
<td>0.109</td>
<td>0.106</td>
<td>0.118</td>
<td>0.104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.024</td>
<td>0.126</td>
<td>0.164</td>
<td>0.148</td>
<td>0.133</td>
<td>0.161</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.021</td>
<td>0.146</td>
<td>0.17</td>
<td>0.155</td>
<td>0.139</td>
<td>0.164</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>N</td>
<td>0.006</td>
<td>0.026</td>
<td>0.029</td>
<td>0.003</td>
<td>0.027</td>
<td>0.025</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.012</td>
<td>0.028</td>
<td>0.036</td>
<td>0.105</td>
<td>0.033</td>
<td>0.044</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.016</td>
<td>0.083</td>
<td>0.169</td>
<td>0.153</td>
<td>0.07</td>
<td>0.174</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.017</td>
<td>0.084</td>
<td>0.173</td>
<td>0.159</td>
<td>0.07</td>
<td>0.163</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>N</td>
<td>0.014</td>
<td>0.006</td>
<td>0.041</td>
<td>0.021</td>
<td>0.031</td>
<td>0.034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.022</td>
<td>0.02</td>
<td>0.164</td>
<td>0.141</td>
<td>0.159</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.02</td>
<td>0.133</td>
<td>0.157</td>
<td>0.161</td>
<td>0.163</td>
<td>0.168</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.023</td>
<td>0.129</td>
<td>0.171</td>
<td>0.161</td>
<td>0.16</td>
<td>0.163</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>N</td>
<td>0.009</td>
<td>0.01</td>
<td>0.013</td>
<td>0.021</td>
<td>0.006</td>
<td>0.008</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.016</td>
<td>0.023</td>
<td>0.046</td>
<td>0.05</td>
<td>0.025</td>
<td>0.075</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.021</td>
<td>0.122</td>
<td>0.155</td>
<td>0.122</td>
<td>0.155</td>
<td>0.168</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.017</td>
<td>0.135</td>
<td>0.157</td>
<td>0.145</td>
<td>0.161</td>
<td>0.17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>N</td>
<td>0.009</td>
<td>0.003</td>
<td>0.027</td>
<td>0.021</td>
<td>0.026</td>
<td>0.026</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.018</td>
<td>0.012</td>
<td>0.035</td>
<td>0.069</td>
<td>0.027</td>
<td>0.046</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.018</td>
<td>0.128</td>
<td>0.174</td>
<td>0.147</td>
<td>0.168</td>
<td>0.165</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.019</td>
<td>0.128</td>
<td>0.168</td>
<td>0.149</td>
<td>0.164</td>
<td>0.163</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>N</td>
<td>0.01</td>
<td>0.004</td>
<td>0.026</td>
<td>0.025</td>
<td>0.027</td>
<td>0.023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.019</td>
<td>0.019</td>
<td>0.054</td>
<td>0.085</td>
<td>0.066</td>
<td>0.062</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.02</td>
<td>0.132</td>
<td>0.166</td>
<td>0.153</td>
<td>0.168</td>
<td>0.162</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.021</td>
<td>0.139</td>
<td>0.164</td>
<td>0.151</td>
<td>0.157</td>
<td>0.164</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Twitter</td>
<td>Y</td>
<td>N</td>
<td>0.019</td>
<td>0.128</td>
<td>0.175</td>
<td>0.154</td>
<td>0.174</td>
<td>0.162</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>College</td>
<td>N</td>
<td>Y</td>
<td>0.021</td>
<td>0.007</td>
<td>0.131</td>
<td>0.114</td>
<td>0.128</td>
<td>0.122</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All Social</td>
<td>Y</td>
<td>Y</td>
<td>0.021</td>
<td>0.134</td>
<td>0.169</td>
<td>0.157</td>
<td>0.164</td>
<td>0.169</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All Data</td>
<td>N</td>
<td>N</td>
<td>0.013</td>
<td>0.026</td>
<td>0.032</td>
<td>0.052</td>
<td>0.015</td>
<td>0.046</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Y</td>
<td>0.013</td>
<td>0.021</td>
<td>0.033</td>
<td>0.048</td>
<td>0.017</td>
<td>0.037</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>N</td>
<td>0.013</td>
<td>0.084</td>
<td>0.163</td>
<td>0.102</td>
<td>0.058</td>
<td>0.156</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>0.015</td>
<td>0.087</td>
<td>0.159</td>
<td>0.112</td>
<td>0.067</td>
<td>0.162</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.11: Summary of algorithm F1 score for basketball data from 2020.
<table>
<thead>
<tr>
<th>Data Used</th>
<th>Twitter</th>
<th>College</th>
<th>ADA</th>
<th>Logistic</th>
<th>Random</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra</th>
<th>Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Regression</td>
<td>Forest</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Positions</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.065)</td>
<td>(.0043)</td>
<td>(.00465)</td>
<td>(.0055)</td>
<td>(.0065)</td>
<td>(.0065)</td>
</tr>
<tr>
<td><strong>Only</strong></td>
<td>N (.12)</td>
<td>(.0065)</td>
<td>(.054, 239)</td>
<td>(.043, 229)</td>
<td>(.087, 261)</td>
<td>(.087, 294)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.13)</td>
<td>(.076, 239)</td>
<td>(.097, 294)</td>
<td>(.087, 272)</td>
<td>(.087, 261)</td>
<td>(.087, 294)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.13)</td>
<td>(.097, 261)</td>
<td>(.087, 283)</td>
<td>(.109, 283)</td>
<td>(.054, 283)</td>
<td>(.119, 304)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Team</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.077)</td>
<td>(.0109)</td>
<td>(.0.077)</td>
<td>(.0.087)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
</tr>
<tr>
<td><strong>Only</strong></td>
<td>N (.109)</td>
<td>(.0087)</td>
<td>(.097, 316)</td>
<td>(.097, 326)</td>
<td>(.109, 294)</td>
<td>(.109, 283)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.13)</td>
<td>(.022, 174)</td>
<td>(.109, 304)</td>
<td>(.097, 304)</td>
<td>(.032, 152)</td>
<td>(.13, 304)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.043, 174)</td>
<td>(.109, 304)</td>
<td>(.109, 316)</td>
<td>(.01, 164)</td>
<td>(.109, 294)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Career</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.065)</td>
<td>(.0055)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
</tr>
<tr>
<td><strong>Length Only</strong></td>
<td>N (.109)</td>
<td>(.0087)</td>
<td>(.022, 109)</td>
<td>(.022, 109)</td>
<td>(.109, 272)</td>
<td>(.109, 272)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.13)</td>
<td>(.054, 272)</td>
<td>(.109, 283)</td>
<td>(.065, 261)</td>
<td>(.109, 272)</td>
<td>(.097, 326)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.12)</td>
<td>(.087, 261)</td>
<td>(.109, 294)</td>
<td>(.109, 304)</td>
<td>(.13, 283)</td>
<td>(.119, 316)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.077)</td>
<td>(.0065)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
<td>(.0.065)</td>
</tr>
<tr>
<td><strong>Only</strong></td>
<td>N (.109)</td>
<td>(.0087)</td>
<td>(.022, 109)</td>
<td>(.01, 142)</td>
<td>(.0.087)</td>
<td>(.043, 174)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.076, 239)</td>
<td>(.109, 304)</td>
<td>(.087, 272)</td>
<td>(.109, 304)</td>
<td>(.109, 305)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.087, 261)</td>
<td>(.109, 304)</td>
<td>(.097, 294)</td>
<td>(.13, 294)</td>
<td>(.119, 316)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rank</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.098)</td>
<td>(.0065)</td>
<td>(.0.087)</td>
<td>(.0.087)</td>
<td>(.0.065)</td>
<td>(.0.087)</td>
</tr>
<tr>
<td><strong>Only</strong></td>
<td>N (.109)</td>
<td>(.0098)</td>
<td>(.022, 152)</td>
<td>(.043, 196)</td>
<td>(.043, 152)</td>
<td>(.043, 142)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.12)</td>
<td>(.097, 272)</td>
<td>(.119, 326)</td>
<td>(.076, 283)</td>
<td>(.13, 304)</td>
<td>(.087, 294)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.109, 283)</td>
<td>(.109, 304)</td>
<td>(.076, 304)</td>
<td>(.109, 305)</td>
<td>(.119, 316)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Valuation</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.098)</td>
<td>(.0065)</td>
<td>(.0.087)</td>
<td>(.0.087)</td>
<td>(.0.065)</td>
<td>(.0.087)</td>
</tr>
<tr>
<td><strong>Only</strong></td>
<td>N (.109)</td>
<td>(.0098)</td>
<td>(.022, 152)</td>
<td>(.043, 196)</td>
<td>(.043, 152)</td>
<td>(.043, 142)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.12)</td>
<td>(.097, 272)</td>
<td>(.119, 326)</td>
<td>(.076, 283)</td>
<td>(.13, 304)</td>
<td>(.087, 294)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.109, 283)</td>
<td>(.109, 304)</td>
<td>(.076, 304)</td>
<td>(.109, 305)</td>
<td>(.119, 316)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Twitter</strong></td>
<td></td>
<td></td>
<td></td>
<td>Y (.109)</td>
<td>(.087, 261)</td>
<td>(.13, 304)</td>
<td>(.097, 283)</td>
<td>(.119, 304)</td>
<td>(.13, 304)</td>
</tr>
<tr>
<td><strong>College</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.142)</td>
<td>(.0087)</td>
<td>(.097, 294)</td>
<td>(.065, 261)</td>
<td>(.109, 283)</td>
<td>(.097, 272)</td>
</tr>
<tr>
<td><strong>All Social</strong></td>
<td></td>
<td></td>
<td></td>
<td>Y (.13)</td>
<td>(.065, 272)</td>
<td>(.13, 316)</td>
<td>(.109, 326)</td>
<td>(.13, 283)</td>
<td>(.097, 272)</td>
</tr>
<tr>
<td><strong>All</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.109)</td>
<td>(.0087)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td></td>
<td></td>
<td></td>
<td>N (.109)</td>
<td>(.0087)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
<td>(.0.109)</td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.043, 174)</td>
<td>(.097, 283)</td>
<td>(.054, 217)</td>
<td>(.022, 142)</td>
<td>(.109, 283)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y (.109)</td>
<td>(.054, 174)</td>
<td>(.109, 294)</td>
<td>(.065, 229)</td>
<td>(.043, 164)</td>
<td>(.109, 272)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.12: Summary of 95% confidence intervals for the basketball data from 2020. Each algorithm was run 100 times.
Table A.13: Summary of algorithm accuracy for college basketball data for 2001-2019. As with baseball, the inclusion of social data greatly increases the accuracy, sometimes by over 25%.

<table>
<thead>
<tr>
<th>Data Used</th>
<th>College</th>
<th>ADA %</th>
<th>Logistic Regression %</th>
<th>Random Forest %</th>
<th>XGBoost %</th>
<th>KNN %</th>
<th>Extra Trees %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>6.1%</td>
<td>11.2%</td>
<td>16.6%</td>
<td>13.4%</td>
<td>13.3%</td>
<td>16.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.0%</td>
<td>11.2%</td>
<td>20.4%</td>
<td>14.0%</td>
<td>17.8%</td>
<td>20.8%</td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>6.6%</td>
<td>10.9%</td>
<td>15.8%</td>
<td>12.5%</td>
<td>7.1%</td>
<td>15.4%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.7%</td>
<td>11.1%</td>
<td>17.7%</td>
<td>13.1%</td>
<td>9.7%</td>
<td>17.3%</td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>6.0%</td>
<td>11.9%</td>
<td>16.9%</td>
<td>14.3%</td>
<td>13.6%</td>
<td>16.9%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.0%</td>
<td>12.3%</td>
<td>22.6%</td>
<td>14.7%</td>
<td>20.7%</td>
<td>22.7%</td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>5.9%</td>
<td>11.2%</td>
<td>15.8%</td>
<td>11.7%</td>
<td>12.9%</td>
<td>16.4%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.0%</td>
<td>11.3%</td>
<td>16.8%</td>
<td>12.0%</td>
<td>13.5%</td>
<td>17.8%</td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>6.6%</td>
<td>11.0%</td>
<td>15.7%</td>
<td>12.3%</td>
<td>12.8%</td>
<td>16.5%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.4%</td>
<td>11.2%</td>
<td>16.5%</td>
<td>13.1%</td>
<td>13.7%</td>
<td>16.8%</td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>5.9%</td>
<td>11.7%</td>
<td>16.5%</td>
<td>12.6%</td>
<td>14.1%</td>
<td>16.9%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.0%</td>
<td>11.5%</td>
<td>18.7%</td>
<td>13.4%</td>
<td>16.5%</td>
<td>19.9%</td>
</tr>
<tr>
<td>College Data</td>
<td>N</td>
<td>4.3%</td>
<td>4.2%</td>
<td>8.4%</td>
<td>6.8%</td>
<td>7.2%</td>
<td>8.6%</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>6.4%</td>
<td>11.1%</td>
<td>16.6%</td>
<td>12.3%</td>
<td>8.7%</td>
<td>17.2%</td>
</tr>
<tr>
<td>Data Used</td>
<td>College</td>
<td>ADA</td>
<td>Logistic</td>
<td>Random</td>
<td>XGBoost</td>
<td>KNN</td>
<td>Extra</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>-------</td>
<td>----------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>0.036</td>
<td>0.118</td>
<td>0.191</td>
<td>0.154</td>
<td>0.147</td>
<td>0.189</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.035</td>
<td>0.117</td>
<td>0.208</td>
<td>0.152</td>
<td>0.175</td>
<td>0.208</td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>0.041</td>
<td>0.108</td>
<td>0.17</td>
<td>0.132</td>
<td>0.071</td>
<td>0.164</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.043</td>
<td>0.11</td>
<td>0.179</td>
<td>0.136</td>
<td>0.097</td>
<td>0.173</td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>0.033</td>
<td>0.128</td>
<td>0.188</td>
<td>0.158</td>
<td>0.144</td>
<td>0.186</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.033</td>
<td>0.127</td>
<td>0.227</td>
<td>0.157</td>
<td>0.206</td>
<td>0.227</td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>0.033</td>
<td>0.127</td>
<td>0.227</td>
<td>0.157</td>
<td>0.206</td>
<td>0.227</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.036</td>
<td>0.12</td>
<td>0.179</td>
<td>0.128</td>
<td>0.136</td>
<td>0.181</td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>0.047</td>
<td>0.123</td>
<td>0.176</td>
<td>0.138</td>
<td>0.134</td>
<td>0.177</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.043</td>
<td>0.121</td>
<td>0.169</td>
<td>0.141</td>
<td>0.137</td>
<td>0.171</td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>0.035</td>
<td>0.127</td>
<td>0.176</td>
<td>0.141</td>
<td>0.144</td>
<td>0.177</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.035</td>
<td>0.124</td>
<td>0.191</td>
<td>0.145</td>
<td>0.163</td>
<td>0.2</td>
</tr>
<tr>
<td>College</td>
<td>N</td>
<td>0.008</td>
<td>0.007</td>
<td>0.074</td>
<td>0.057</td>
<td>0.067</td>
<td>0.077</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>0.04</td>
<td>0.111</td>
<td>0.168</td>
<td>0.127</td>
<td>0.086</td>
<td>0.173</td>
</tr>
</tbody>
</table>

Table A.14: Summary of algorithm F1 score for college basketball data from 2001-2019. The model with the greatest accuracy also had the highest F1 score.

<table>
<thead>
<tr>
<th>Data Used</th>
<th>College</th>
<th>ADA</th>
<th>Logistic</th>
<th>Random</th>
<th>XGBoost</th>
<th>KNN</th>
<th>Extra</th>
<th>Regression</th>
<th>Forest</th>
<th>Trees</th>
<th>Forest</th>
<th>Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions Only</td>
<td>N</td>
<td>(.046, .075)</td>
<td>(.094,.127)</td>
<td>(.15,.19)</td>
<td>(.114,.154)</td>
<td>(.117,.156)</td>
<td>(.149,.186)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.045, .076)</td>
<td>(.094,.123)</td>
<td>(.183,.222)</td>
<td>(.124,.16)</td>
<td>(.159,.2)</td>
<td>(.189,.227)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Only</td>
<td>N</td>
<td>(.049, .083)</td>
<td>(.094,.123)</td>
<td>(.139,.174)</td>
<td>(.107,.142)</td>
<td>(.061,.084)</td>
<td>(.14,.174)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.051, .084)</td>
<td>(.096,.127)</td>
<td>(.157,.199)</td>
<td>(.116,.147)</td>
<td>(.08,.112)</td>
<td>(.158,.191)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Career Length Only</td>
<td>N</td>
<td>(.046, .076)</td>
<td>(.104,.132)</td>
<td>(.147,.188)</td>
<td>(.128,.164)</td>
<td>(.116,.155)</td>
<td>(.149,.187)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.044,.075)</td>
<td>(.11,.139)</td>
<td>(.207,.245)</td>
<td>(.131,.161)</td>
<td>(.186,.227)</td>
<td>(.205,.248)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Only</td>
<td>N</td>
<td>(.046, .076)</td>
<td>(.095,.129)</td>
<td>(.138,.179)</td>
<td>(.101,.133)</td>
<td>(.112,.147)</td>
<td>(.149,.181)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.047,.074)</td>
<td>(.099,.13)</td>
<td>(.15,.186)</td>
<td>(.106,.135)</td>
<td>(.115,.152)</td>
<td>(.156,.199)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank Only</td>
<td>N</td>
<td>(.05, .08)</td>
<td>(.095,.126)</td>
<td>(.139,.177)</td>
<td>(.107,.143)</td>
<td>(.11,.142)</td>
<td>(.142,.183)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.049,.079)</td>
<td>(.096,.127)</td>
<td>(.144,.183)</td>
<td>(.11,.151)</td>
<td>(.117,.155)</td>
<td>(.149,.186)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valuation Only</td>
<td>N</td>
<td>(.041, .074)</td>
<td>(.104,.137)</td>
<td>(.149,.181)</td>
<td>(.11,.143)</td>
<td>(.124,.159)</td>
<td>(.148,.187)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.042,.075)</td>
<td>(.098,.13)</td>
<td>(.167,.211)</td>
<td>(.118,.151)</td>
<td>(.148,.183)</td>
<td>(.174,.218)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>College</td>
<td>N</td>
<td>(.032, .053)</td>
<td>(.027,.056)</td>
<td>(.069,.097)</td>
<td>(.053,.082)</td>
<td>(.054,.085)</td>
<td>(.073,.102)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>(.051,.081)</td>
<td>(.095,.129)</td>
<td>(.142,.185)</td>
<td>(.108,.139)</td>
<td>(.074,.101)</td>
<td>(.151,.192)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.15: Summary of 95% confidence intervals for the college basketball data from 2001-2019. Each algorithm was run 100 times.
<table>
<thead>
<tr>
<th>Top Attended Colleges</th>
<th>Num of Total Alumni</th>
<th>Max Alumni in a Year</th>
<th>Max Alumni on a Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kentucky</td>
<td>119</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>UNC</td>
<td>115</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>UCLA</td>
<td>98</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>UConn</td>
<td>96</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Duke</td>
<td>92</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

Table A.16: Statistics about the number of alumni from a college playing for the top five schools with the most alumni.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Hyperparameter</th>
<th>MLB Value</th>
<th>NBA Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic</td>
<td>fit_intercept</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Logistic</td>
<td>max_iter</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Regression</td>
<td>penalty</td>
<td>none</td>
<td>l2</td>
</tr>
<tr>
<td>Regression</td>
<td>solver</td>
<td>newton-cg</td>
<td>liblinear</td>
</tr>
<tr>
<td>Ada</td>
<td>learning_rate</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ada</td>
<td>algorithm</td>
<td>SAMME</td>
<td>SAMME</td>
</tr>
<tr>
<td>Ada</td>
<td>n_estimators</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Random</td>
<td>criterion</td>
<td>gini</td>
<td>gini</td>
</tr>
<tr>
<td>Random</td>
<td>min_samples_leaf</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Random</td>
<td>min_samples_split</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Random</td>
<td>n_estimators</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Random</td>
<td>max_features</td>
<td>auto</td>
<td>auto</td>
</tr>
<tr>
<td>Random</td>
<td>max_leaf_nodes</td>
<td>null</td>
<td>None</td>
</tr>
<tr>
<td>Random</td>
<td>max_depth</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>KNN</td>
<td>algorithm</td>
<td>auto</td>
<td>auto</td>
</tr>
<tr>
<td>KNN</td>
<td>n_neighbors</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>KNN</td>
<td>leaf_size</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>KNN</td>
<td>weight</td>
<td>distance</td>
<td>distance</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>criterion</td>
<td>gini</td>
<td>gini</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>max_depth</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>max_leaf_nodes</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>max_features</td>
<td>auto</td>
<td>auto</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>min_samples_leaf</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>min_samples_split</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>n_estimators</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>XGB</td>
<td>max_depth</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>XGB</td>
<td>learning_rate</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>XGB</td>
<td>min_child_weight</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XGB</td>
<td>reg_alpha</td>
<td>.2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table A.17: The hyperparameters used for MLB and NBA models.
Bibliography


[71] Simon Rottenberg. The baseball players’ labor market. 64(3):242–258.


[82] Katie Willyerd. What high performers want at work.


[85] Uncovering the presence of vulnerable open-source components in commercial software.

[86] Open source security and risk analysis report.


[91] National vulnerability database.

