Can We Out-scout The Scouts? A Rigorous Exploration and Modeling of NBA Draft Analyses

September 14, 2023

Matthew Spitulnik (msspitul@syr.edu)

Beau Spratt (bspratt@syr.edu)

Sintia Stabel (sstabel@syr.edu)

Matthew Smith (msmit146@syu.edu)

Final Project Paper

Syracuse University

ISTS 736 – Text Mining

INTRODUCTION

The NBA Draft: an annual event that captures the imagination of basketball fans around the globe, serving as a pivotal juncture in both the league's and young players' narratives. This event, brimming with anticipation, symbolizes hope for many teams looking to rebuild or reinforce their rosters. It is the stage where college standouts, international prodigies, and young hopefuls converge, aspiring to make their mark in the illustrious world of professional basketball. As teams vie for the next superstar or hidden gem, they are faced with a myriad of decisions, each carrying the weight of potential success or regret.

To navigate this complex decision-making process, teams have historically relied on traditional scouting methods, involving exhaustive player observations, interviews, and individual workouts. Scouts would traverse the country, and often the globe, seeking to uncover talent and gather firsthand insights into a player's skill set, work ethic, and mentality. Yet, in today's data-driven era, these age-old techniques are being augmented, and in some cases replaced, by sophisticated analytical methods.

Embarking on this comprehensive exploration, the focal point of the analysis will be the extraction of intricate scouting data from a myriad of online sources. These data points capture both the strengths and potential pitfalls associated with each player, as meticulously detailed in scouting reports. Such reports are not just mere summaries; they are the bedrock upon which teams, especially those in the NBA, base their decisions during pivotal events like the NBA Draft. On the momentous night of the draft, all 30 teams are in a strategic battle, leveraging their most refined player scouting intelligence to outmaneuver their counterparts.

But could the advancements in data science and the power of automation challenge the traditional norms of scouting? Could these modern techniques emulate, or even surpass, the insights gleaned by professional scouts who dedicate entire years to player assessment, and do

so in mere seconds? This investigative paper dives deep into the world of predictive modeling, employing text mining techniques on pre-draft text data. The overarching goal is to explore an automated, data-driven approach to scout and predict the future trajectory of players.

Forecasting the potential success or downfall of a player in professional sports is no trivial feat. It's a labyrinthine challenge that necessitates the careful formulation of precise labels to capture the essence of player performance. These labels, pivotal for any predictive modeling exercise, serve as the linchpin for translating raw athletic achievements into meaningful training data. Particularly in basketball, a sport rife with unpredictability where a player's career trajectory can be influenced by a multitude of factors, there exists no universally accepted metric to brand a player as a "bust". Nevertheless, subtle hints embedded within a player's statistics, especially in relation to their draft position, can sometimes provide foresight into their future performance.

Venturing into this domain, our study meticulously evaluates four distinct methodologies for label creation. Each of these methodologies provides a fresh lens through which player performance can be scrutinized, offering insights into the potential hallmarks of success or indicators of impending failure. By juxtaposing these diverse methodologies, our intent is to illuminate the multifaceted nature of predictive modeling within sports analytics. We underscore the paramount importance of label formulation and its profound influence on the precision and reliability of the predictive outcomes.

ANALYSIS

Overall Data.

NBA_API.

The “nba_api” is an open-source API that is used to collect data directly from NBA.com. It allows access to a large array of information, past and present, including individual player stats, organizational histories, and draft recaps. To begin collecting the data that would be needed for this analysis, the “nba_api” was used to collect every NBA player that had been drafted, going all the way back to 2000. If this information were to be viewed on the NBA.com website, it would appear as is shown in Figure 1:

NBA.com Draft Information

PLAYER	TEAM	AFFILIATION	YEAR	OVERALL PICK
Paolo Banchero	Orlando Magic	Duke	2022	1
Chet Holmgren	Oklahoma City Thunder	Gonzaga	2022	2
Jabari Smith Jr.	Houston Rockets	Auburn	2022	3
Keegan Murray	Sacramento Kings	Iowa	2022	4
Jaden Ivey	Detroit Pistons	Purdue	2022	5
Bennedict Mathurin	Indiana Pacers	Arizona	2022	6
Shaedon Sharpe	Portland Trail Blazers	Kentucky	2022	7
Dyson Daniels	New Orleans Pelicans	Ignite (G League)	2022	8
Jeremy Sochan	San Antonio Spurs	Baylor	2022	9

Figure 1

When the nba_api is used to collect the draft data within python, it can be directly imported into an organized, dataframe format. Figures 2 and 3 show excerpts from the imported draft dataframe:

Draft Dataframe Imported Using NBA_API

Index	PERSON_ID	PLAYER_NAME	SEASON	ROUND_NUMBER	ROUND_PICK	OVERALL_PICK	DRAFT_TYPE
0	1641705	Victor Wembanyama	2023	1	1	1	Draft
1	1641706	Brandon Miller	2023	1	2	2	Draft
2	1630703	Scoot Henderson	2023	1	3	3	Draft
3	1641708	Amen Thompson	2023	1	4	4	Draft
4	1641709	Ausar Thompson	2023	1	5	5	Draft
5	1641710	Anthony Black	2023	1	6	6	Draft
6	1641731	Bilal Coulibaly	2023	1	7	7	Draft
7	1641716	Jarace Walker	2023	1	8	8	Draft
8	1641707	Taylor Hendricks	2023	1	9	9	Draft
9	1641717	Cason Wallace	2023	1	10	10	Draft

Figure 2

Draft Dataframe Imported Using NBA_API (continued)

TEAM_ID	TEAM_CITY	TEAM_NAME	TEAM_ABBREVIATION	ORGANIZATION	ORGANIZATION_TYPE	R_PROFI
1610612759	San Antonio	Spurs	SAS	Metropolitans 92 (France)	Other Team/Club	1
1610612766	Charlotte	Hornets	CHA	Alabama	College/University	1
1610612757	Portland	Trail Blazers	POR	Ignite (G League)	Other Team/Club	1
1610612745	Houston	Rockets	HOU	Overtime Elite	Other Team/Club	1
1610612765	Detroit	Pistons	DET	Overtime Elite	Other Team/Club	1
1610612753	Orlando	Magic	ORL	Arkansas	College/University	1
1610612754	Indiana	Pacers	IND	Metropolitans 92 (France)	Other Team/Club	1
1610612764	Washington	Wizards	WAS	Houston	College/University	1
1610612762	Utah	Jazz	UTA	Central Florida	College/University	1
1610612742	Dallas	Mavericks	DAL	Kentucky	College/University	1

Figure 3

This dataframe contained key information like each player's name, the year they were drafted, and at what point in the draft they were selected. Since scouting reports for players could not be located online before the year 2000, anyone drafted before then was dropped from the dataframe.

NBADRAFT.NET

Although a plethora of information could be imported using the NBA API, it did not have easy access to the pre-draft player scouting reports that would be required to perform this analysis. A website needed to be found that contained a large collection of these scouting reports in a standardized format across all of its pages so that a script could be written that would be able to easily collect as much of this information as possible. NBADRAFT.NET contained pre-draft scouting reports going all the way back to the year 2000, stored in a format that was mostly standard from player page to player page. Figure 4 displays one of the scouting report pages from NBADRAFT.NET:

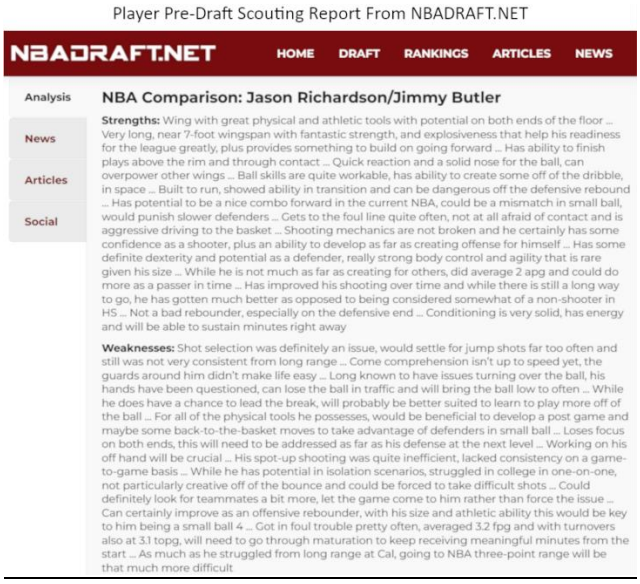


Figure 4

Each page containing a scouting report on NBADRAFT.net had the same base URL:" https://www.nbadraft.net/players/". For each player, the link to get to their report contained the base URL followed by the players first name, followed by a dash (-), followed by their last name. For example, the link to the page for the NBA player Jaylen Brown is: https://www.nbadraft.net/players/jaylen-brown/. The original data collected using the nba_api contained a players full name in one column, instead of separated into two columns. The link for each player also did not contain any suffix's that they may go by, such as "jr." or "III", but the nba_api information did contain them. Since the players first name and last name needed to be manipulated so that they were separated by a dash, and the suffix information had to be removed, the column containing each player's name from the imported nba_api dataframe was manually split it up into its individual elements. Some players had extended names, so two suffix columns were required. The updated columns can be seen in Figure 5:

New Player Name Columns

First Name	Last Name	Suffix1	Suffix2
Victor	Wembanyama	None	None
Brandon	Miller	None	None
Scoot	Henderson	None	None
Amen	Thompson	None	None
Ausar	Thompson	None	None
Anthony	Black	None	None
Bilal	Coulibaly	None	None
Jarace	Walker	None	None
Taylor	Hendricks	None	None
Cason	Wallace	None	None
Jett	Howard	None	None
Dereck	Lively	II	None

Figure 5

A script was now created that used the new name columns to go through each player in the draft dataframe, form an NBADRAFT.NET link that should redirect to that players scouting report page, download the contents of the page, then use BeautifulSoup (an HTML parser) to clean the data and extract the desired information. The script also included a built-in timer that paused its running for 45-60 seconds after each player's info was collected to ensure no terms-of-service were being violated.

The main data being targeted by this script was the write up of each player's "Strengths", and each players "Weaknesses", but some scouting reports included additional information that was also extracted when available in case it could be used later: an overall numeric score for each player that took into account various basketball related skills, a comparison to a previous or current NBA player, additional player notes in the write up section, a predicted outlook for the player in the write up section, and an overall summary of the player in the write up section. As this data was collected, it was all added directly back into the draft dataframe, which can be seen in Figure 6:

Scouting Report Data Added To Draft Dataframe

Strengths	Weaknesses	Outlook	Notes	OverTxt
A generational talent … The sky is the limit and…	He has bulked up considerably, but he …	Victor Wembanyama i…	nan	nan
An extremely talented 6'9 SF/PF with a lanky fra…	Though he has solid athleticism, he la…	Incoming Alabama fr…	nan	Miller was a consens…
Henderson is a 6'2 190 guard with outstanding le…	3-point shot needs improvement (27 3FG…	nan	nan	Scoot Henderson has …
Thompson is a 6'7 210 perimeter player who is ph…	Jump shot is a big concern currently (…	nan	Measured 6'…	Amen Thompson is a p…
Thompson is a 6'7 215 wing with the physical too…	Can be too reliant on his athleticism;…	nan	Measured 6'…	Ausar Thompson who, …
A physically gifted PG/SG who stands a legitimat…	Weaknesses: Some question his toughne…	Incoming Arkansas f…	Native of D…	Anthony Black was a …
Ultra athletic wing, with great size … Excellent…	He needs to bulk up considerably to pl…	nan	nan	Coulibaly is an athl…
6'8 versatile forward/small ball five-man with a…	While versatile, he doesn't have great…	Incoming Houston fr…	Measured 6'…	nan
Hendricks is a slender 6'9 215 PF/C prospect who…	Will need to add more muscle to his fr…	nan	Measured 6'…	Hendricks is a Fr. w…
Wallace may very well be the most physical perim…	Productivity (11.7 ppg) and efficienc…	Incoming Kentucky f…	Measured 6'…	Cason Wallace projec…

Figure 6

For various reasons, the information for some players could not be collected by the script. This included some players that would be important inclusions to the analysis, like Zion Williamson—who was a first overall pick and considered a generational talent but hasn't lived up to the potential—and Giannis Antetokounmpo—who wasn't drafted until the middle of the first round but is now arguably the best player in the NBA. Other players had Strength's information imported but not Weaknesses, and vice versa. These cases were all individually examined and if possible, their information was manually updated in the data frame. Ultimately, out of the 1425 players that had been drafted since the year 2000, the Strengths were collected for 1047 players, and the Weaknesses were collected for 1042 players.

Basketball-Reference.com.

T perform some of the analysis that would determine if a player was a bust, additional statistics needed to be collected for each player. The nba_api could provide some of this information, but it did not include all the advanced stats that exist for NBA players. Instead, the website "basketball-reference.com" was used. Basketball-reference.com (and it's parent site,

"sports-reference.com"), is an encyclopedia-like collection of vast amounts of sports data from all major North American professional sports leagues, as well as come college sports. Figure 7 displays a tiny fraction of some of the advanced stats that basketball-reference.com can provide:

Advanced Stats From Basketball-Reference.com

Advanced **Bold** indicates league leader Share & Export ▾ Glossary

Regular Season | Playoffs

Season	Age	Tm	Lg	Pos	G	MP	PER	TS%	3PAr	FTr	ORB%	DRB%	TRB%	AST%	STL%	BLK%	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM	DBPM	BPM	VORP
2016-17	20	BOS	NBA	SF	78	1341	10.3	.539	.319	.293	3.8	14.4	9.1	7.2	1.3	1.1	12.5	18.1	0.2	1.3	1.5	.053	-3.0	-0.8	-3.7	-0.6
2017-18	21	BOS	NBA	SG	70	2152	13.6	.562	.381	.290	3.4	14.0	8.8	8.5	1.6	1.0	12.0	21.4	1.2	3.2	4.5	.100	-0.8	0.3	-0.5	0.8
2018-19	22	BOS	NBA	SG	74	1913	13.5	.547	.348	.255	3.6	13.9	8.7	7.7	1.7	1.5	10.1	22.1	0.7	2.4	3.0	.076	-1.6	-0.4	-2.1	0.0
2019-20	23	BOS	NBA	SG	57	1934	16.9	.583	.381	.274	3.5	16.5	10.1	9.7	1.6	1.0	11.2	24.7	2.3	2.6	4.9	.123	1.0	-0.2	0.8	1.4
2020-21 ★	24	BOS	NBA	SG	58	1999	19.9	.586	.370	.225	3.9	15.5	9.7	16.5	1.8	1.5	11.5	29.7	2.8	2.0	4.8	.115	3.2	-0.7	2.5	2.3
2021-22	25	BOS	NBA	SF	66	2220	18.9	.574	.381	.261	2.6	16.7	9.8	17.9	1.6	0.7	11.6	30.5	2.3	3.4	5.8	.124	1.9	0.1	2.1	2.2
2022-23 ★	26	BOS	NBA	SF	67	2405	19.1	.581	.352	.249	3.6	17.0	10.4	16.5	1.5	0.9	11.4	31.4	1.6	3.4	5.0	.100	1.5	-0.2	1.3	2.0
Career			**NBA**		**470**	**13964**	**16.4**	**.572**	**.365**	**.259**	**3.5**	**15.5**	**9.6**	**12.5**	**1.6**	**1.1**	**11.4**	**26.0**	**11.1**	**18.4**	**29.5**	**.101**	**0.5**	**-0.2**	**0.3**	**8.1**

Figure 7

The three stats columns that are highlighted in the image, "PER", "WS/48", and "VORP", are 3 key statistics that can be used to evaluate how much a good a player is and how much they contribute. PER, or "Player Efficiency Rating", measures a players per-minute production. WS/48, which is "Win Shares Per 48 Minutes", estimates the number of wins a player contributes to their team for every 48 minutes played. VORP, or "Value Over Replacement Player", calculates how much more a player can contribute to their team over a hypothetical replacement player that would be brought in instead.

Like DRAFTNET.com, basketball-reference.com has a pattern that is used for each player's page. The basketball-reference.com link hierarchy is a little more in-depth, however. Every single person (not just players—coaches, referees, team owners, etc.) that has information on basketball-reference.com is assigned a unique reference-ID, which is comprised of the first 5 letters of their last name, plus the first 2 letters of their first name, plus 01 if they are the first person to have that letter combination, 02 if they are the second person, etc. For example, the letter portion of Jaylen Brown's reference ID would be "brownja", but he is not the first person on the site to have that letter combination, so his full reference ID is "brownja02".

The URL pattern to get to a player's page is www.basketball-reference.com/players/, followed by the first letter of their last name, followed by /<their reference-ID>.html. Following these rules, the link to get to Jaylen Brown's page would be https://www.basketball-reference.com/players/b/brownja02.html. Because the wrinkle created by the player's reference-ID was not guaranteed to be predictable, a few steps had to be taken to locate their stats instead. First, instead of trying to go to an individual player's page, the first letter from their last name was used to create a link to the page that lists all players with last names starting with the same letter. In Jaylen Brown's case, this meant going to the page that listed all NBA players, current and past, with last names that started with "B", or "/www.basketball-reference.com/players/b/#players". Part of this page can be seen in Figure 8:

"B" Last Name Players, Basketball-Reference.com

Player	From	To	Pos	Ht	Wt	Birth Date
Gerald Brown	1999	1999	G	6-4	210	July 28, 1975
Greg Brown III	2022	2023	F	6-9	205	September 1, 2001
Harold Brown	1947	1947	G	6-0	155	October 2, 1923
Jabari Brown	2015	2015	G	6-4	215	December 18, 1992
Jaylen Brown	2017	2023	F-G	6-6	223	October 24, 1996
John Brown	1974	1980	F	6-7	220	December 14, 1951
Kedrick Brown	2002	2005	G	6-7	222	March 18, 1981
Kendall Brown	2023	2023	G-F	6-8	205	May 11, 2003
Kwame Brown	2002	2013	F	6-11	270	March 10, 1982
Larry Brown*	1968	1972	G	5-9	160	September 14, 1940
Leon Brown	1947	1947	F	6-3	190	October 12, 1919

Figure 8

Again, using BeautifulSoup, the HTML and the text for the entire list of players was temporarily imported, and then the player-in-question's full name was searched through the list. If the players name was found in the list, an embedded link to their corresponding page could be pulled out, as can be seen in Figure 9, which displays the developer-tool view of the HTML behind the players name in the list:

Figure 9

Multiple players in the NBA have had the same name, so searching by their name alone was not guaranteed to provide accurate results. For additional verification, once a player's supposed page URL had been located, the draft year of the player was also confirmed in the biography section of the player's page. If the draft year matched, the required advanced stats were collected from the page. If the draft year did not match, the script looked for another instance of the players name in the list from the players page, and then continued trying until the correct player page was found.

As the advanced stats were located for each player, they were stored in a new advanced stats dataframe. If a player hadn't played a single minute in the NBA yet, they would not qualify for any of the advanced stats, and their information would remain blank. Once the data collection was completed, the stats dataframe was merged with the draft dataframe. Figure 10 shows a piece of the final data collection/draft dataframe:

Final Data Collection Dataframe

Strengths	Weaknesses	Outlook	Notes	OverTxt	Has_Page	PER	WS_48	VORP
All-around player w…	Long and wiry ath…	Incoming …	Measured…	Brow…	yes	11	0.063	0.5
Solidly built and s…	Washington does n…	Considere…	Declared…	nan	yes	nan	nan	nan
Highly refined, old…	Lacks the overall…	nan	NBA doct…	nan	yes	16.2	0.113	2.7
Strengths: Broad sh…	Weaknesses: Still…	nan	nan	nan	yes	14.5	0.101	0
Jefferson is a wiry…	Needs to continue…	nan	nan	Jeff…	yes	13.2	0.071	-0.2
Parker is on anothe…	Defense and posit…	nan	A Mormon…	Park…	yes	16.3	0.078	2.5
A fabulous athlete…	Even though he ha…	nan	His on-c…	nan	yes	15.6	0.054	0.9
6’7 explosive wing …	His floor game is…	Incoming …	Measured…	Cam …	yes	nan	nan	nan
Warren is a big tim…	The obvious weakn…	nan	Measured…	Warr…	yes	nan	nan	nan
Fits the prototype …	Okoro is a work i…	Auburn fr…	Measured…	Okor…	yes	9.5	0.077	-0.8

Figure 10

Basic draft and player stats were pulled with the NBA_API. This API crawls the nba.com website for official data that provides easy access from programming environments. Data like player name, draft year, draft position, career minutes, points, rebounds and assists per game were all available here. This was mainly use for creating labels for a supervised dataset.

To build a supervised labeled dataset, a formulaic approach is used. 4 bust labels are made via stats that can be compared for accuracy. The following formulas are used for different label theories.

Formulaic Approach:

- ***Bust 1:*** TBD for new players; >6K Minutes played Then “N” Otherwise “Y”

- ***Bust 2:*** TBD for new players; Bscore > 52.50762 Then “N” Otherwise “Y” Bscore= MinPG*(0.8)+PPG+RPG*(1.8)+APG*(1.8) * DraftMultiplier

- ***Bust 3:*** Adds MID Label; Bscore > 59.86251366 = “N” Otherwise Bscore <= 46.49273 Then “Y”

- ***Bust 4:*** TBD for players with insufficient stats; Ascore = 0.5zPER + 0.3zWS/48 + 0.2*zVORP > 0.277004 = “N” Otherwise “Y”

Bust1 makes the simplest assumption if you got minutes, the team liked how you were playing and therefore are not a bust. This may miss players who are injured or had high expectations because they were drafted early. Bust 2 uses basic stats to make sure the player is contributing on court. This includes “MinPG” (playing enough minutes in each game), “PPG” (are you scoring each game), “RPG” (getting rebounds in each game), “APG” (getting assists in each game). Bust 3 is the same as bust 2 but adds a MID label for average players. This makes the label perhaps surer of itself or more confident so models may receive less noisy data near the decision boundary. Bust 4 is the more advanced formula using comprehensive stats that judges the overall performance of players. It uses “PER” (Player Efficiency Rating), “WS/48” (Win Shares per 48 minutes) it may overweight big men who play less minutes but have good per-minute production and “VORP’ (Value Over Replacement Player) captures a player's contribution compared to an average replacement. “Z” in this formula is a way to balance each stat, a common scale with a mean of 0 and standard deviation of 1. A Bust label comparison test will be performed below that trains models on 545 players (have Y/N labels from each formula).

After these techniques models can be built to predict if players are a bust. The models used in the exercise for this prediction are multinomial naive bayes (MNB), Sentiment, Bernoulli, SVM, Decision Trees including Random Forest.

Sentiment Analysis and Score Assignment.

The dataset utilized in this portion of the study was compiled through a combination of web scraping and data integration techniques. The primary objective was to gather comprehensive pre-draft analysis information for NBA players, including their strengths,

weaknesses, outlook, and other related attributes. The dataset was obtained from the website nbadraft.net, which provides in-depth analysis and profiles of NBA draft prospects.

To collect this data, a Python script was developed to automate the process of scraping player-specific analysis pages on the website. The following key steps were involved in data collection:

Draft History Data: The script accessed the NBA draft history through the nba_api, fetching information about drafted players, including their names and draft years. This served as the foundation for selecting players for analysis.

Filtering the Draft History: To ensure a relevant dataset, the draft history data was filtered to include only records from the year 2000 onwards, as this was deemed the most pertinent period for analysis.

Web Scraping Analysis Pages: The script iterated through the filtered draft history dataset, constructing URLs for each player's analysis page on nbadraft.net. These URLs were used to access the pre-draft analysis content for each player.

Beautiful Soup Parsing: Beautiful Soup, a Python library for web scraping, was employed to parse the HTML content of each player's analysis page. The script specifically targeted sections of interest, including NBA comparisons, strengths, weaknesses, outlook, and notes.

Data Storage and Update: The collected analysis data for each player was stored in a dictionary and then written to a text file named player_dict.txt. Furthermore, the original draft history dataframe was updated to include the collected analysis data for each player.

Sleep Timer: To ensure ethical and considerate web scraping practices, a sleep timer was incorporated into the script. This timer imposed a delay of 60 seconds between each web request to avoid overwhelming the website's server.

Data Cleanup and Additional Scraping: The script performed various data cleanup and additional scraping tasks to address unique issues associated with different player analysis pages, such as handling variations in text formats and extracting an "overall" category when present.

NBA Player Statistics: Towards the end of the script, an attempt was made to collect NBA player statistics, including Player Efficiency Rating (PER), Win Shares per 48 minutes (WS/48), and Value Over Replacement Player (VORP). These statistics were extracted by constructing specific URLs based on player names.

Data Storage: The collected player statistics were stored in a dataframe named player_stats_DF and exported to a CSV file for further analysis and research.

The resulting dataset combines web scraping techniques with meticulous data manipulation to provide comprehensive pre-draft analysis information for NBA players. It contains 1425 rows and 24 columns. The columns include descriptive values such as player names, team, and season as well as the analytical commentary to be used in this study in columns: “Strengths”, Weaknesses” and “Outlook”.

Figures 11 and 12 show the first 5 rows of the resulting dataset prior to transformation and cleaning techniques were employed.

	PERSON_ID	PLAYER_NAME	SEASON	ROUND_NUMBER	ROUND_PICK	\
0	1641705	Victor Wembanyama	2023	1	1	
1	1641706	Brandon Miller	2023	1	2	
2	1630703	Scoot Henderson	2023	1	3	
3	1641708	Amen Thompson	2023	1	4	
4	1641709	Ausar Thompson	2023	1	5	

	OVERALL_PICK	DRAFT_TYPE	TEAM_ID	TEAM_CITY	TEAM_NAME	...	\
0	1	Draft	1610612759	San Antonio	Spurs	...	
1	2	Draft	1610612766	Charlotte	Hornets	...	
2	3	Draft	1610612757	Portland	Trail Blazers	...	
3	4	Draft	1610612745	Houston	Rockets	...	
4	5	Draft	1610612765	Detroit	Pistons	...	

	First Name	Last Name	Suffix1	Suffix2	Overall	NBA Comparison	\
0	Victor	Wembanyama	NaN	NaN	101.0	Ralph Sampson	
1	Brandon	Miller	NaN	NaN	95.0	Paul George	
2	Scoot	Henderson	NaN	NaN	97.0	Derrick Rose/Ja Morant	
3	Amen	Thompson	NaN	NaN	93.0	Latrell Sprewell	
4	Ausar	Thompson	NaN	NaN	93.0	Trevor Ariza	

Figure 11 – NBA Draft Post 2000 Data Unprocessed

	Strengths	\
0	A generational talent … The sky is the limit a...	
1	An extremely talented 6’9 SF/PF with a lanky f...	
2	Henderson is a 6’2 190 guard with outstanding ...	
3	Thompson is a 6’7 210 perimeter player who is ...	
4	Thompson is a 6’7 215 wing with the physical t...	

	Weaknesses	\
0	He has bulked up considerably, but he still ne...	
1	Though he has solid athleticism, he lacks grea...	
2	3-point shot needs improvement (27 3FG%); adde...	
3	Jump shot is a big concern currently (25 3FG%)...	
4	Can be too reliant on his athleticism; doesn’t...	

	Outlook	\
0	Victor Wembanyama is probably the most hyped t...	
1	Incoming Alabama freshman … 2022 McDonald’s Al...	
2	NaN	
3	NaN	
4	NaN	

	Notes
0	NaN
1	NaN
2	NaN
3	Measured 6’ 5.75” barefoot, 8’ 7.50” standing ...
4	Measured 6’ 5.75” barefoot, 8’ 8.00” standing ...

[5 rows x 24 columns]

Figure 12 – NBA Draft Post 2000 Data Unprocessed - Continued

To analyze the text data, computerized processing methods were used to convert text data into vectorized data, from which computer algorithms can extract word frequencies and conduct further analysis. Prior to vectorization, however, the contents from the variables “Strengths”, Weaknesses” and “Outlook”, were preprocessed with the following actions to produce more accurate results during modeling:

- Special Characters and Punctuation: regular expressions (regex) were used to remove any characters that are not alphanumeric or spaces from the text, including numerical digits.
- Convert Text to Lowercase: all text was converted to lowercase to ensure uniformity in text representation. This step was done to avoid any issues related to case sensitivity during analysis.
- Tokenization: The text was tokenized using NLTK's word_tokenize function. Tokenization breaks the text into individual words, which are referred to as tokens.
- Stopwords Removal: Stopwords are common words that do not add much meaning to the text (e.g., 'the', 'and', 'is'). The function removes stopwords using NLTK's set of English stopwords, retaining only meaningful words in the text.
- Removal of Nan Values: Missing values can hinder a model's ability to learn from the data and make accurate predictions. The rows containing nan values were dropped as in this case there was only one occurrence in the entire dataset.

	Strengths_Preprocessed	Weaknesses_Preprocessed	Outlook_Preprocessed
0	unique player tremendous frame ability face ba...	lack great length four 61075 wingspan solid el...	projected possible midsecond round pick 2021 n...
1	high basketball iq greatest strength strong pe...	average athlete lacking great speed leaping ab...	NaN
2	crafty scorer find hole defense passing skill ...	small even point guard ability get inside larg...	NaN
3	physically gasol great upper body strength lik...	unlike pau marc average athlete heavy legged p...	NaN
4	NaN	NaN	NaN

Figure 13 – Strengths, Weaknesses and Outlook After Preprocessing

To conduct a comprehensive Sentiment Analysis, sentiment scores were systematically assigned to specific columns within the dataset, namely, "Strengths_Preprocessed," "Weakness_Preprocessed," and "Outlook_Preprocessed." This intricate process was facilitated

through the utilization of Python's Vader Sentiment library, which encompasses the Sentiment Intensity Analyzer.

Before embarking on the Sentiment Analysis journey, it was imperative to address missing or NaN (Not-a-Number) values within the dataset to ensure the integrity of the analysis. This endeavor primarily involved the "Strengths" and "Weaknesses" columns, whereas the "Outlook" column harbored an extensive number of missing values, surpassing the 1000 mark. The strategy employed for handling missing values was to systematically drop rows that contained NaN values in either the "Strengths" or "Weaknesses" columns.

Following the removal of all rows featuring NaN values in the "Strengths" and "Weaknesses" columns, the dataset remained robust, comprising a total of 1042 data points. Each of these data points corresponded to a distinct draft player, thereby offering a comprehensive foundation for the subsequent Sentiment Analysis.

Having completed the data cleaning and preprocessing, the NBA draft dataset was now ready for exploratory analysis. Two parallel studies took place while exploring the dataset for capturing the positive and negative sentiments of the NBA Drafts analysis reports. In the first exploratory analysis, the columns Strengths_Preprocessed and Weaknesses_Preprocessed were combined into a single string. In the second version, the column Weaknesses_Preprocessed was evaluated by itself.

While analyzing the sentiment scores assigned to the "Strengths_Weaknesses_Combined," it became evident that the dataset exhibited a substantial skew toward positive scores. This trend is clearly illustrated in the bar chart presented in Figure 14, where most scores fall within the range of 0.76 to 1.00. Conversely, when examining the sentiment scores solely for "Weaknesses_Preprocessed," a more balanced distribution is observed, albeit with a slight skew toward the positive end, as depicted in Figure 15.

In addition to these visualizations, Figure 16 introduces WordClouds that showcase the most frequently occurring words based on their associated sentiment scores. These WordClouds offer a visual representation of the prominent terms within the dataset, shedding light on the prevalent themes and sentiments.

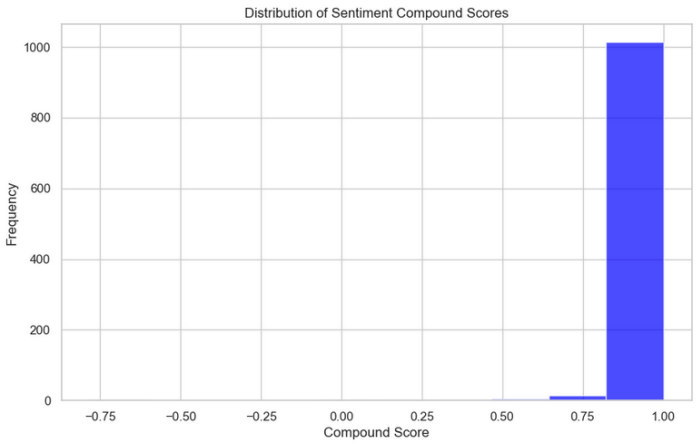


Figure 14 – Distribution of Sentiment Scores - Strengths_Weaknesses_Combined

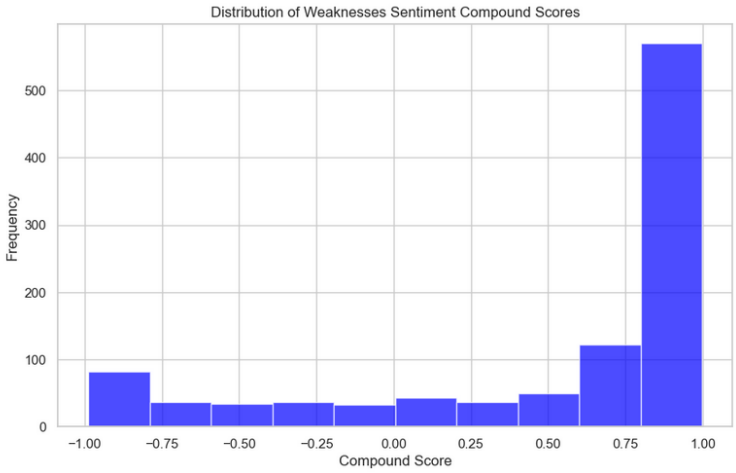


Figure 15 – Distribution of Sentiment Scores - Weaknesses Preprocessed

Figure 16 – WordClouds based on Positive and Negative Sentiment Scores using Weaknesses Preprocessed

The objective of the sentiment analysis is to determine if negative reviews vs positive reviews by the analysts can be used as predictors to determine if a player will be successful or a "bust". Figure 17 shows that with the Weaknesses only scores, most of the players which can be considered "busts" based on their career statistics, have sentiment scores that are positive and above 0.75. Figure 18 shows detailed scores distributions for each scenario, where "bust" is a yes, no, or to be determined.

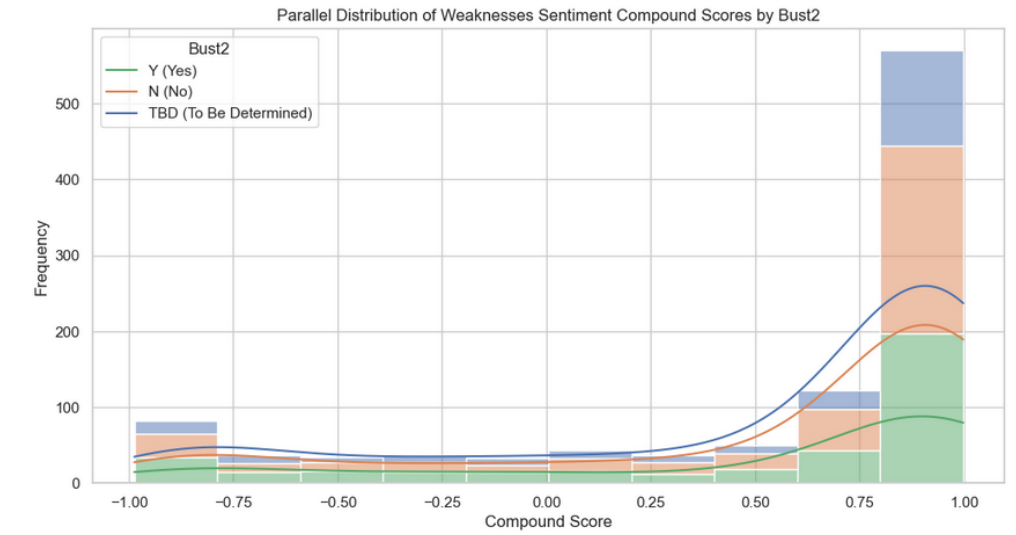


Figure 17 – Sentiment Scores Distributions for Bust2 Criteria – Weakenesses_Compound Scores

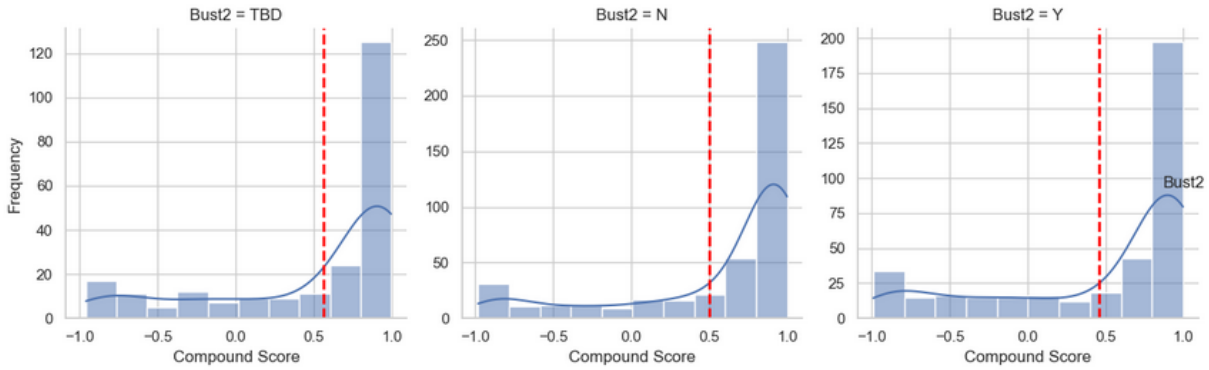


Figure 18– Weakness Compound Scores Distribution of Bust = Yes, Bust2 = N, and Bust2 = Y

Figures 19 and 20 underscore the severity of the skewness in the data when combining the Strenghts and Weakenesses columns.

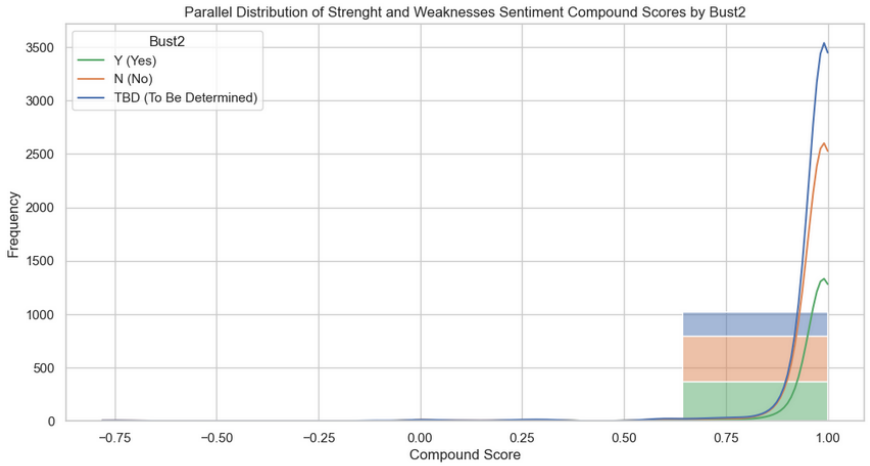


Figure 19 - Sentiment Scores Distributions for Bust2 Criteria – Strengths_Weakenesses_Compound Scores

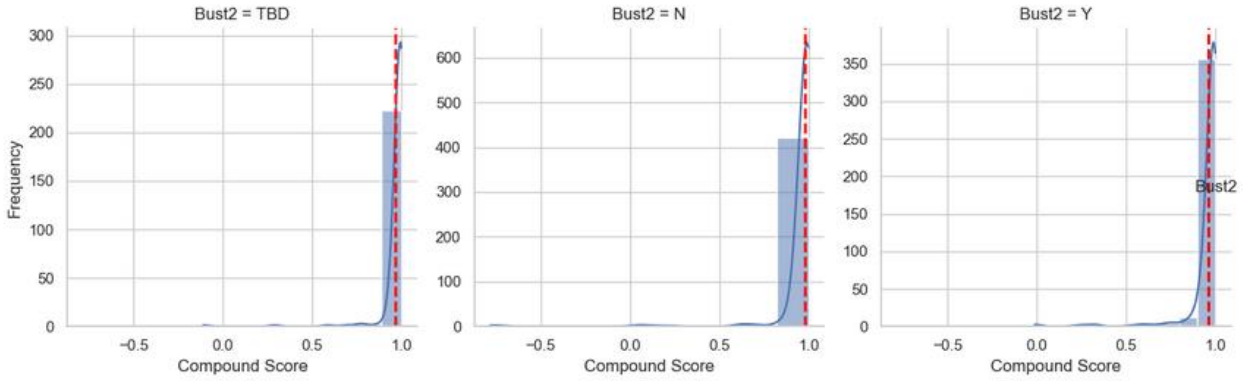


Figure 20 – Sentiment Scores Distributions for Bust2 Criteria – Strengths_Weakenesses_Compound Scores

With the sentiment scores at hand the top 10 most positive and top 10 most negative player reviews can be obtained from the dataset. Figures 21 and 22 show them with along with their respective NBA players and bust or no bust labels.

Top 10 Most Positive Scores:

	PLAYER_NAME	Bust2	Weaknesses_Preprocessed	Weaknesses_Preprossed_Compound
0	Jalen Green	TBD	green good shooter yet sniper beyond arc still...	0.9979
1	RJ Barrett	N	may peaked degree high school clearly top kid ...	0.9977
2	Marvin Bagley III	Y	lack wingspan big men class continue fill fram...	0.9973
3	James Wiseman	TBD	obvious elephant room go draft played minute c...	0.9967
4	Kyle Kuzma	N	kuzma show great deal talent skill still need ...	0.9964
5	Chase Budinger	N	strong oneonone player area game focus end flo...	0.9963
6	Charles Bassey	TBD	need work shooting range consistency hesitates...	0.9962
7	Victor Claver	Y	need add strength although he tough doesnt see...	0.9959
8	Johnny Davis	TBD	asked goto scorer last season necessity aggres...	0.9953
9	Tyrese Maxey	TBD	struggled efficiencyconsistency freshman kentu...	0.9948

Figure 21 – Top 10 Most Positive Scores and Analysis Reviews

```
Top 10 Most Negative Scores:
          PLAYER_NAME Bust2  \
1041     Willie Warren     Y
1040         AJ Price     N
1039      Kemba Walker     N
1038     Chris Richard     Y
1037  DeMarcus Cousins     N
1036    Luke Harangody     N
1035    Isaiah Jackson   TBD
1034  Danilo Gallinari     N
1033     Grant Jerrett     Y
1032    Dakari Johnson     Y

                              Weaknesses_Preprocessed  \
1041  warren suffers poor shot selection asked numbe...
1040  high level nba athlete quickness fine price do...
1039  measurement 61 shoe win remains undersized com...
1038  may struggle keep speed nba game doesnt great ...
1037  cousin lack maturity mental focus evident nega...
1036  luke biggest weakness toughest obstacle overco...
1035  point viewed project next level weighs 200 lb ...
1034  high level european player always equal contri...
1033  jerrett failed prove mediocre player college l...
1032  johnson ceiling relatively low essentially saw...

      Weaknesses_Preprossed_Compound
1041                         -0.9866
1040                         -0.9846
1039                         -0.9743
1038                         -0.9739
1037                         -0.9729
1036                         -0.9720
1035                         -0.9709
1034                         -0.9699
1033                         -0.9685
1032                         -0.9628
```

Figure 22 – Top 10 Most Negative Scores and Analysis Reviews

Models and Methods.

TF-IDF Vectorizer

For the sentiment analysis of the NBA Drafts dataset, the TF-IDF vectorization method was employed. TF-IDF, which stands for "Term Frequency-Inverse Document Frequency," is a technique used to transform textual data into numerical features suitable for machine learning. This approach plays a crucial role in converting text data into a format that can be utilized by machine learning algorithms. It works by calculating a score for each term in the text, indicating its significance in a specific document relative to the entire dataset.

The term frequency (TF) component of TF-IDF measures how frequently a term appears in a document, giving insight into the importance of that term within that document. Conversely,

the inverse document frequency (IDF) component quantifies the rarity of a term across all documents. Terms that are rare and appear in only a few documents are assigned higher IDF scores, highlighting their unique nature and potential importance. By combining the TF and IDF scores, the TF-IDF vectorizer generates a numerical representation for each document, where term importance is weighted based on its frequency within the document and rarity across the dataset.

Text Preprocessing: Lemmatization and Stemming

In addition to employing TF-IDF vectorization, the text data underwent preprocessing steps known as lemmatization and stemming. These techniques were applied to enhance the quality and consistency of the textual content before feeding it into the machine learning models.

Lemmatization: Lemmatization involves reducing words to their base or dictionary forms to normalize variations of words. For instance, "running" and "ran" would be lemmatized to "run." This process helps ensure that different inflections of a word are treated as the same term, leading to better feature representation and improved model performance.

Stemming: Stemming, on the other hand, involves removing prefixes and suffixes from words to reduce them to their root forms. For example, "running" and "runner" would both be stemmed to "run." While stemming is more aggressive than lemmatization, it can sometimes result in words that are not actual words, but the technique is useful in simplifying words to their core meanings.

Vader Sentiment Intensity Analyzer

The VADER (Valence Aware Dictionary and sEntiment Reasoner) Sentiment Intensity Analyzer is a powerful tool for assessing the sentiment or emotional tone of a piece of text. It operates on the principle of lexicon-based sentiment analysis, which means it relies on pre-

compiled dictionaries of words and their associated sentiment scores. These lexicons contain thousands of words, each tagged with a polarity score, indicating whether the word is positive, negative, or neutral, as well as an intensity score, which quantifies the strength of the sentiment.

VADER's magic lies in its ability to handle complex sentiment expressions, including negations and context-based sentiment shifts. It assigns a sentiment polarity score to each word in a text, considering both the individual word's sentiment and the context in which it appears. This is crucial for understanding phrases like "not good," where the negation changes the overall sentiment. VADER also considers capitalization and punctuation for added context. After evaluating all the words and phrases in a text, VADER calculates an overall compound sentiment score, which represents the text's overall sentiment intensity. This compound score can range from -1 (extremely negative) to +1 (extremely positive), with 0 indicating a neutral sentiment. Researchers and analysts often use this compound score to determine the sentiment of a text document accurately, making VADER a valuable tool for sentiment analysis in various applications, from social media monitoring to customer feedback analysis.

Model 1: Random Forest Ensemble (3 Cross Validation)

Random Forest combined with TF-IDF vectorization was used to predict the positive, negative, or neutral sentiments for the player's "Weaknesses_Preprocessed" analysis. Random Forest is an ensemble learning algorithm that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) of the individual trees. It is particularly effective in handling complex relationships and avoiding overfitting.

At the core of the Random Forest algorithm are individual decision trees. Decision trees are hierarchical structures that break down a dataset into smaller and more manageable subsets by asking a series of binary questions based on feature attributes. These questions are

designed to effectively split the data into different classes, ultimately leading to a classification decision at the tree's leaves.

The decision tree works by recursively splitting the data into subsets, with each split being determined by a feature and a threshold. The algorithm aims to find the best feature and threshold that results in the purest subsets, where samples within each subset predominantly belong to a single class (label). The process continues until a predefined stopping criterion is met, such as a maximum depth or a minimum number of samples in a leaf node.

While decision trees have the potential to model complex relationships in the data, they are prone to overfitting, capturing noise in the data and leading to poor generalization to unseen data. Random Forest builds on the strength of decision trees while addressing their limitations. Instead of relying on a single decision tree, Random Forest creates an ensemble of multiple decision trees, each trained on a different subset of the data and with some randomness introduced.

Model 2: Supervised Vector Machines (SVM + 3 Cross Validations)

In Model 2, Support Vector Machines (SVM) with linear kernels were employed for sentiment analysis, and the model's performance was assessed using 3-fold cross-validation. SVM is a powerful supervised learning algorithm used for classification tasks. It works by finding the optimal hyperplane that best separates data points belonging to different classes. In this context, SVM was utilized to classify sentiments in the "Weaknesses_Preprocessed" text data. By optimizing the hyperplane's parameters, SVM aims to maximize the margin between different sentiment classes, enhancing its classification accuracy.

Model 3: Supervised Vector Machines (SVM + Lemmatization + 3 Cross Validations)

Building upon Model 2, Model 3 incorporated lemmatization as a preprocessing step before employing SVM with linear kernels. Lemmatization is a text normalization technique that reduces words to their base or dictionary forms. By lemmatizing the text data, common variations of words are transformed into their root form, reducing dimensionality, and potentially improving the model's performance. Like Model 2, Model 3 underwent 3-fold cross-validation to evaluate its effectiveness in sentiment classification.

Model 4: Supervised Vector Machines (SVM + Stemming + 3 Cross Validations + Bigrams)

Model 4 followed a similar approach to Model 3 but utilized stemming instead of lemmatization as a preprocessing step. Stemming involves removing suffixes or prefixes from words to obtain their root forms, which can help reduce the vocabulary size and improve the SVM model's performance. Stemming may be more aggressive than lemmatization, potentially leading to different results. As with the previous models, Model 4 underwent 3-fold cross-validation to assess its ability to classify sentiments effectively.

Model 5: Latent Dirichlet Allocation (LDA) – Topic Modeling

In Model 5, a different and insightful approach to text analysis was employed, focusing on topic modeling using Latent Dirichlet Allocation (LDA). Unlike the previous models, which were designed for sentiment prediction, LDA is an unsupervised machine learning technique with broader applications. It's primarily used to reveal hidden topics within a collection of text documents, making it particularly relevant for uncovering the underlying themes in the context of NBA draft player analysis.

LDA operates on the principle that each document in the dataset is a mixture of various topics, and each topic is, in turn, a mixture of specific words or terms. By examining the patterns of word co-occurrence across documents, LDA effectively identifies these latent topics and

quantifies their prevalence within the dataset. This approach goes beyond sentiment analysis and enables a deeper exploration of the content contained within the NBA Drafts Analysis text data.

So, how can LDA help predict whether NBA draft players will be successful ("non-busts") or underperform ("busts")? While LDA itself doesn't provide direct predictions about players' outcomes, it offers valuable insights into the characteristics and themes associated with player weaknesses. By identifying latent topics within the strengths and weaknesses' commentaries, analysts can gain a nuanced understanding of the types of weaknesses that are commonly mentioned and their relative importance. This information can then be used as a feature in predictive models or as part of a broader analysis to assess whether certain weakness themes are indicative of future success or challenges in the NBA. In essence, LDA serves as a powerful tool for uncovering hidden patterns and trends within textual data, which can inform more informed decision-making in player evaluation and selection processes.

Naïve Bayes Analysis.

Data Preparation and Cleaning.

For analysis to be performed on the text data contained in each scouting report, it would now need to be broken up into a document-term matrix (DTM), which takes every single word, phrase, or combination of characters in a piece of text, breaks them down into individual tokens, and then counts how frequently each token appears in each document (each scouting report in this case would be a different document). This was accomplished in two ways: using the CountVectorizer (CV) and TfidfVectorizer (TF) tools from the scikit learn libraries. The CV tool performs just a simple raw count of how many times each word appears in each review. The TF tool produces a normalized value that reflects the importance and rarity of a word by comparing

how often it appears in each document versus how often it appears across the entire collection of documents. Words that are more relevant to a particular document have a TF score closer to 1, while words that are less relevant for a particular document will have a TF closer to zero. Figure 23 below shows excerpts of two DTMs side-by-side, one created using CV and the other created using TF:

CV Vs. TF Dataframe

CountVectorizer

Index	abilty	able
0	1	1
1	0	0
2	0	0
3	0	1
4	0	1
5	0	1
6	0	0
7	0	1
8	0	1
9	0	1

TfidfVectorizer

Index	abilty	able
0	0.17642	0.0515207
1	0	0
2	0	0
3	0	0.0561702
4	0	0.0654126
5	0	0.0505995
6	0	0
7	0	0.0368484
8	0	0.0418245
9	0	0.0685906

Figure 23

In addition to single word/term DTMs being created, DTMs were also created using bi-grams, meaning words were coupled together with the word that appeared directly before or after them. Figure 24 below shows an example of a bi-gram DTM:

Bi-gram DTM

Index	ability attack	ability average	ability ball	ability better	ability block
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0.0480301	0	0	0	0
9	0	0	0	0	0

Figure 24

During the data collection process, various fields of text were collected that could potentially be analyzed, but the most consistent data was found in the ‘Strengths’ (STR) and ‘Weaknesses’ (WKN) data. DTMs were created using just the STR column data, just the WKN column data, and then using the STR and WKN (SAW) data combined. In addition to being broken up by the column of text, the analysis was also performed against all 4 of the different ‘bust’ labels that had been established. Depending on how the ‘bust’ label was calculated, some players qualified for certain labels but may not have qualified for others. As each model was run against each label, only players that qualified for that particular label were included in the analysis, meaning the models for each label had slightly different row counts. Figure 25 shows an excerpt from a DTM created using the raw STR data to be analyzed against the ‘bust’ label:

STR DTM: Bust 1 Label

Index	97	98	99	aaron	aau	aba
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

Figure 25

Each column in the above DTM should represent a word that was found in one of the pre-draft scouting reports, but as can be seen, some of the column headers contain numbers instead, or were a collection of characters that could not be interpreted as words. The data would need to go through additional cleaning in order to remove columns/words that were not going to significantly contribute to the analysis.

In the first step of the cleaning process, contractions (don’t, won’t, couldn’t, etc.) needed to be broken up into their original two respective words to make sure the cleaning process didn’t remove punctuation and leave behind words like “couldn”. The name of the player was removed

then removed from their scouting report. Any words/columns containing numbers needed to be removed since they would not help contribute to textual analysis. Instead of words like “enjoy” and “enjoying” or “play” and “plays” counting as separate words, the text data would need to be lemmatized and plural words needed to be converted to their singular form to ensure there were as few word variants as possible. Any stop words needed to be removed, since these are extremely common words that appear so frequently, they would contribute to the analysis. Any words less than 4 characters or more than 13 characters also needed to be removed, since these words may be too common or too unique (respectively) to help differentiate one topic from another.

In the next step of the cleaning process, the now-cleaned up DTM was examined to see if any columns/words appeared too frequently or did not appear frequently enough. If a word appeared too frequently across a majority of the scouting reports, it would not be contributing to a report’s uniqueness, therefore not helping to establish any patterns in the data that could be used for analysis. At the other end of the spectrum, if a word did not appear frequently enough, it too would not be contributing to patterns in the data that would help with the analysis. With this all-in mind, any words that appeared in 99% of the scouting reports or less then 1% of the scouting reports were all removed.

In the final step of the cleaning process, each DTM was inputted into a function that went through each of its columns and compared them to each other to determine if they were highly positively or negatively correlated, since Naïve Bayes models make the assumption that all independent variables are not correlated. If two variables were found to be highly positively correlated (above .75), or highly negatively correlated (below -.75), whichever variable of the two was less correlated to the ‘bust’ label being used was dropped.

Figure 26 below displays how the number of columns/words changed in the DTM’s before and after each cleaning step was applied:

Column Change Info After Each Cleaning Step (Single Word)

Index	analysis	Raw Col Total	Raw-Step1 Diff	% Change1	Step1 Total	Step1-2 Diff	% Change2	Step2 Total	Step2-3 Diff	% Change3	Final Total
0	col_saw_1	8899	3288	0.369	5611	4129	0.736	1482	6	0.004	1476
1	col_saw_2	8873	3272	0.369	5601	4125	0.736	1476	6	0.004	1470
2	col_saw_3	8188	3039	0.371	5149	3628	0.705	1521	7	0.005	1514
3	col_saw_4	8935	3338	0.374	5597	4029	0.72	1568	9	0.006	1559
4	col_str_1	7015	2620	0.373	4395	3256	0.741	1139	7	0.006	1132
5	col_str_2	6997	2609	0.373	4388	3256	0.742	1132	8	0.007	1124
6	col_str_3	6390	2393	0.374	3997	2831	0.708	1166	9	0.008	1157
7	col_str_4	7020	2656	0.378	4364	3158	0.724	1206	9	0.007	1197
8	col_wkn_1	6280	2395	0.381	3885	2969	0.764	916	7	0.008	909
9	col_wkn_2	6257	2384	0.381	3873	2962	0.765	911	7	0.008	904
10	col_wkn_3	5781	2216	0.383	3565	2605	0.731	960	6	0.006	954
11	col_wkn_4	6398	2447	0.382	3951	2981	0.754	970	10	0.01	960
12	Mean Change	7252.75	2721.42	0.376	4531.33	3327.42	0.735	1203.92	7.583	0.007	1196.33

Figure 26

The DTMs created from the raw, unaltered data, contained on average about 7253 columns. Once the first cleaning step was applied (general data cleanup), that number was reduced on average by about 2721 columns, for a reduction percentage of about 38%, resulting in about 4531 remaining columns per DTM. When step 2 of the cleaning process was applied (remove words that were too frequent/not frequent enough), the number of words in the DTMs were reduced by about 3327 words on average, or about 74%, leaving about 1204 words/columns per DTM. When the final cleaning step was applied (removing correlated variables), only about 8 columns on average were removed from each DTM, equaling a .7% change and leaving about 1196 columns per DTM. Figure 26 below also shows an in depth look at how the number of columns/words change after each cleaning step:

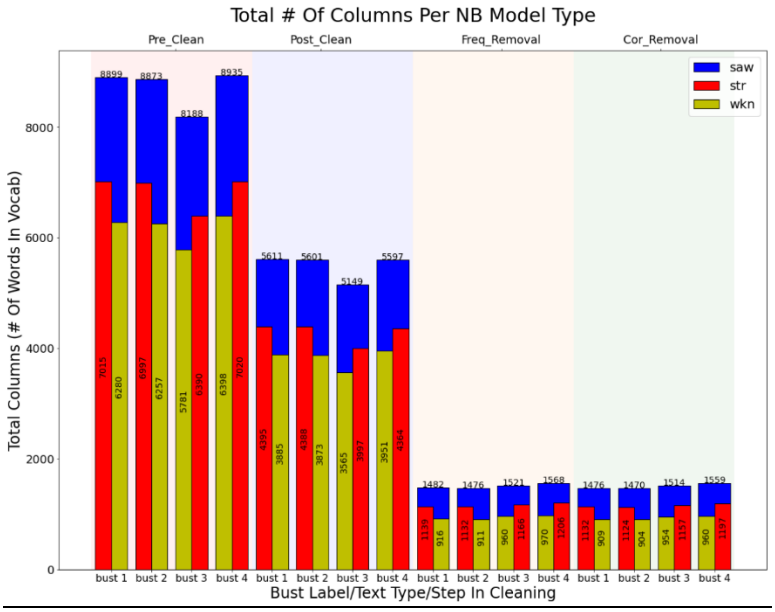


Figure 26

Figure 27 below displays how the number of words/columns changed for the bi-gram DTMs as the cleaning steps were applied:

Column Change Info after Each Claning Step (Bigrams)

Index	analysis	Raw Col Total	Raw-Step1 Diff	% Change1	Step1 Total	Step1-2 Diff	% Change2	Step2 Total	Step2-3 Diff	% Change3	Final Tota
0	BI_col_saw_1	108433	25317	0.233	83116	80851	0.973	2265	13	0.006	2252
1	BI_col_saw_2	107983	25228	0.234	82755	80511	0.973	2244	13	0.006	2231
2	BI_col_saw_3	94233	23115	0.245	71118	68563	0.964	2555	16	0.006	2539
3	BI_col_saw_4	115144	26546	0.231	88598	86052	0.971	2546	17	0.007	2529
4	BI_col_str_1	74735	18672	0.25	56063	54697	0.976	1366	7	0.005	1359
5	BI_col_str_2	74438	18619	0.25	55819	54461	0.976	1358	8	0.006	1350
6	BI_col_str_3	64226	16775	0.261	47451	45913	0.968	1538	9	0.006	1529
7	BI_col_str_4	78375	19387	0.247	58988	57461	0.974	1527	10	0.007	1517
8	BI_col_wkn_1	57537	16600	0.289	40937	40281	0.984	656	7	0.011	649
9	BI_col_wkn_2	57282	16522	0.288	40760	40110	0.984	650	7	0.011	643
10	BI_col_wkn_3	50065	15117	0.302	34948	34193	0.978	755	6	0.008	749
11	BI_col_wkn_4	61510	17655	0.287	43855	43118	0.983	737	7	0.009	730
12	Mean Change	78663.4	19962.8	0.26	58700.7	57184.2	0.975	1516.42	10	0.007	1506.42

Figure 27

The bigram DTMs created from the raw, unaltered data, contained on average about 78663 columns. Once the first cleaning step was applied (general data cleanup), that number was reduced on average by about 19963 columns, for a reduction percentage of about 26%, resulting in about 58,701 remaining columns per DTM. When step 2 of the cleaning process was applied (remove bigrams that were too frequent/not frequent enough), the amount of

bigrams in the DTMs were reduced by about 57184 bigrams on average, or about 98%, leaving about 1516 bigrams/columns per DTM. When the final cleaning step was applied (removing correlated variables), only about 10 columns on average were removed from each DTM, equaling a .7% change and leaving about 1506 columns per DTM. Figure 28 below also shows an in depth look at how the number of columns/bigrams change after each cleaning step:

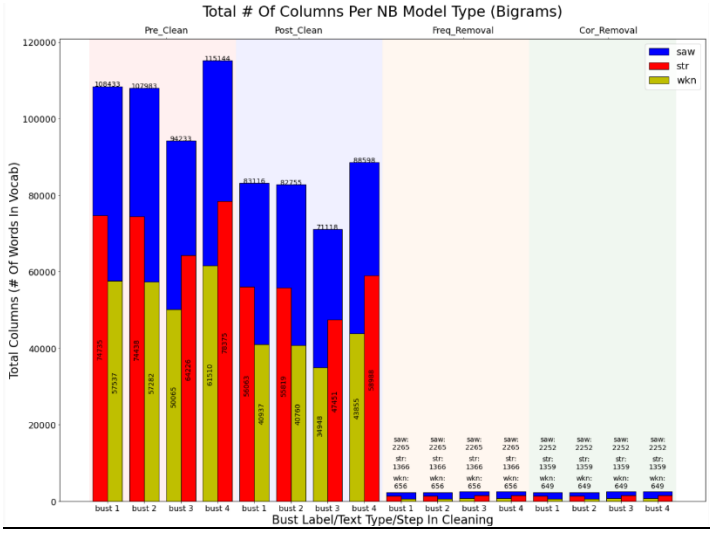


Figure 28

NB Methods Used.

This portion of the analysis explored the ability of naïve bayes (NB) models to predict whether players were going to be busts based on their textual pre-draft scouting reports. NB uses a probabilistic approach to build models. It determines the probability or likelihood that a certain datapoint would fall under a certain label based on the values of the variables or features associated with that data point. The basic NB algorithm can be seen in Figure 29:

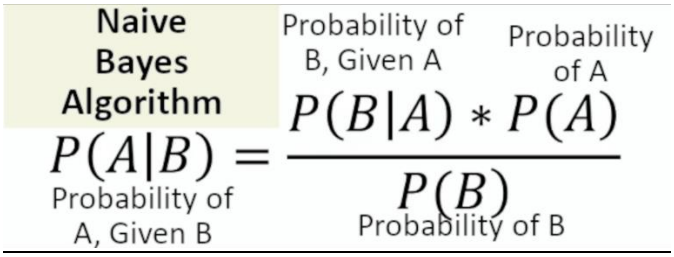


Figure 29

There are multiple variations of NB models, but for this analysis, multinomial NB (MNB) and Bernoulli NB (BNB) were used. A key difference between MNB and BNB is the type of data that can be used by each NB technique. MNB uses continuous number counts or frequencies of the given features in the dataset. BNB uses Boolean/binary data, which only looks at whether the features exist for each data point. Figure 30 shows an excerpt from a dataframe used to create a MNB model side-by-side with an excerpt from a dataframe used to create a BNB model:

MNB Vs. BNB Dataframes

MNB

Index	ability	able
0	1	1
1	3	0
2	0	0
3	2	1
4	0	1
5	3	1
6	3	0
7	2	3
8	5	2
9	0	2

BNB

Index	ability	able
0	1	1
1	1	0
2	0	0
3	1	1
4	0	1
5	1	1
6	1	0
7	1	1
8	1	1
9	0	1

Figure 30

In the MNB dataframe, under the “ability” column, the number 3 appears in row 1, meaning the word “ability” appeared three times in the scouting report in row 1. Row 3 in this column contains a 2, meaning “ability” appeared twice in the row 3 scouting report. Under the

column “able”, there is a 3 in row 7, meaning the word “able” appeared seven times in this scouting report. When the BNB method is used, since Boolean/binary data is needed instead of the continuous counts seen in the MNB data, the “binary” option is used when running the CV function on the data. This results in only zeros “0” and ones “1” for each cell value in the BNB DTM dataframe, “0” representing that the given word was not used in the document, and “1” representing that it did appear in the document. That is why in the BNB dataframe from Figure 11, row 1 under “ability” only contains a 1 instead of a 3, row 3 contains 1 instead of a 2, and row 7 under “able” contains a 1 instead of a 3.

The MNB and BNB models were created and run using the 3 different scouting report text inputs (str,wkn,saw), both vectorizer types (CV/TF), the text from each cleaning step (pre-clean: “pre”, post-clean: “cle”, frequent/non-frequent terms removed: “fre”, correlated variables removed: “cor”), the 4 different bust labels, and single words versus bigrams. This translated to 384 unique models that were created. Each model was cross-validated 10 times, for a total of 3,840 total NB models created and tested.

Long Term Short Term (LSTM) and Convolutional Neural Network (CNN) Deep Learning Models.

Data Preparation and Cleaning.

The dataset, which contains player attributes, was loaded and filtered to focus on rows where the 'Bust' column is marked as either 'Y' (Yes) or 'N' (No), indicating whether a player turned out to be a "bust". To provide a holistic view, the 'Strengths' and 'Weaknesses' columns were combined into a single text feature. This merged text data was then tokenized, and sequences were padded to ensure consistent input length across all data points. The target variable, 'Bust', was transformed into a binary format using label encoding. Subsequently, the

processed data was divided into training and validation sets, ensuring a structured approach to model training and evaluation.

Model Building.

A systematic approach to model building was adopted, experimenting with various deep learning architectures:

- **LSTM Model:** Utilizing the power of LSTM cells, which are adept at handling sequence data like text, separate inputs for 'Strengths' and 'Weaknesses' were fed into the model (Figure 31).

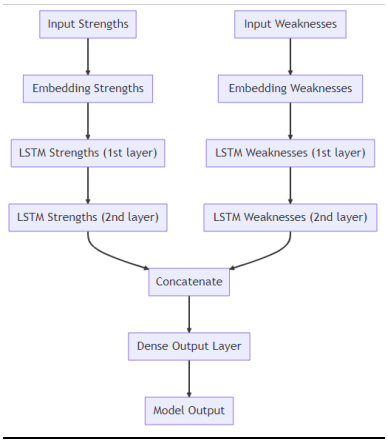


Figure 31.

- **CNN Model:** Transitioning from LSTMs, a Convolutional Neural Network was used, leveraging the capability of CNNs to capture local patterns in data (Figure 32).

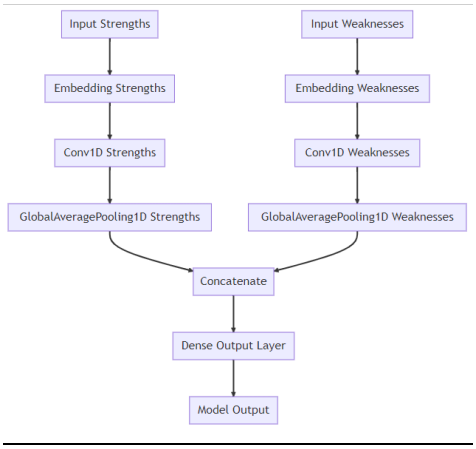


Figure 32

- **CNN with Word2Vec:** This model integrated the pre-trained Word2Vec embeddings, aiming to enhance the model's understanding of the text by leveraging pre-existing semantic relationships between words (Figure 33).

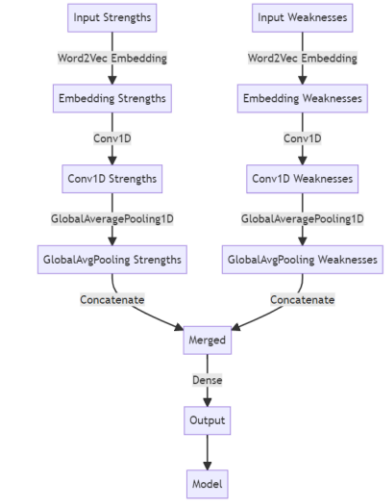


Figure 33

- **CNN with GloVe:** In a similar vein, the GloVe embeddings were employed with a CNN architecture, capitalizing on the rich semantic information encapsulated in the GloVe model (Figure 34).

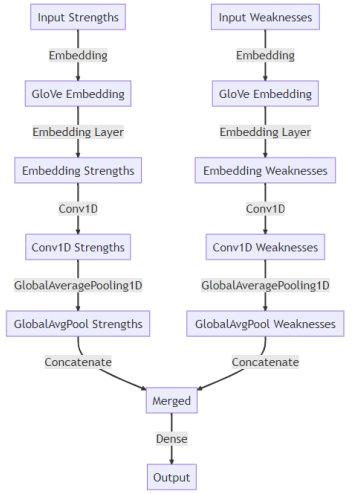


Figure 34

Each model, after receiving data through embedding layers, channeled it through architecture-specific neural layers (either LSTM or CNN). The processed data was then passed to a dense layer for the binary classification task.

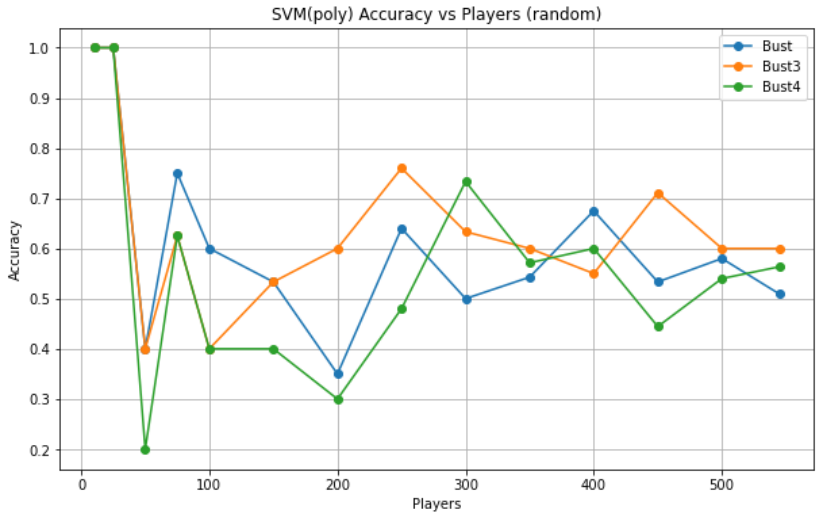
Training and Evaluation.

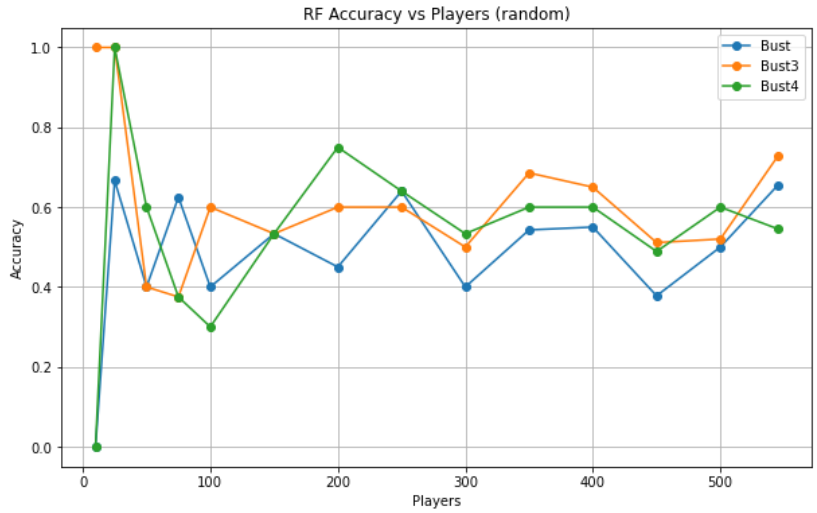
The models were compiled using the Adam optimizer with binary cross-entropy as the chosen loss function. During training, performance metrics were stored epoch-wise to monitor the models' learning trajectories. Post-training, the epoch yielding the highest validation accuracy was identified and duly highlighted.

RESULTS

Multinomial Naive bayes (MNB), Sentiment, Bernoulli, SVM, Decision Trees Random Forest.

Bust label comparison testing consists of up to 545 players. Each of these players has a Y/N label for bust of not. This filters out all players with TBD/MID which includes players with insignificant stats and makes Bust 2 irrelevant but can cause imbalance which is discussed further below. Two types of models are included in this comparison SVM (poly) and Random Forest.



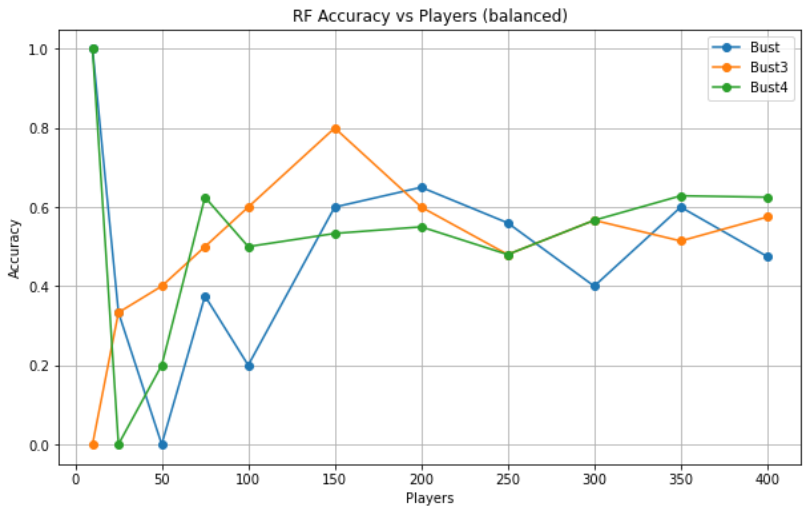


Results above show how accuracy can vary greatly as players are fed into the model. The highest SVM accuracy comes from Bust 3 when fed 250 players ~75%. For RF, Bust 4 does impressively at 200 players ~74% while Bust 3 does well at all 545 ~71%. These results also highlight how more training data could hurt accuracy. This can be a balance of underfitting

and overfitting while dealing with noisy text data. Also, training balance or a skew of labels should be analyzed as well. The table below shows % of how many Y/N labels remained in each.

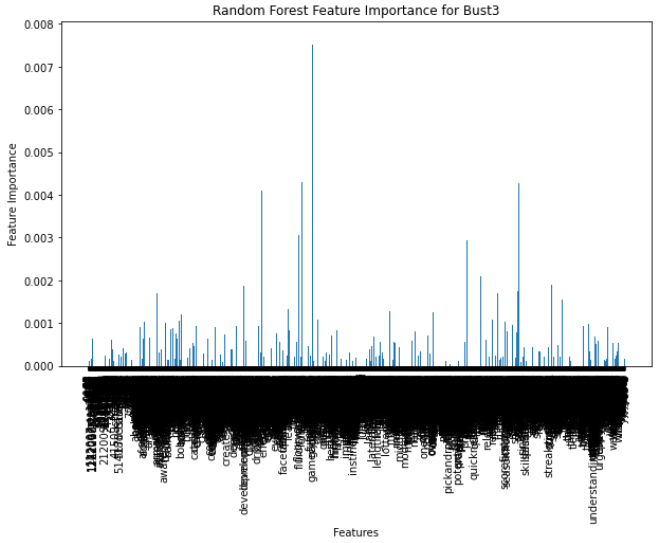
545	Bust1	Bust3	Bust4
Y	0.488073	0.388991	0.458716
N	0.517431	0.616514	0.546789

Bust 3 did well in accuracy but if balance turns out to be important it may not be the best choice for predictions as it had a 61/39 ratio. Another option is to only feed in a balanced set of labels and test the model's accuracy. Below is how random forest did when using balanced input.

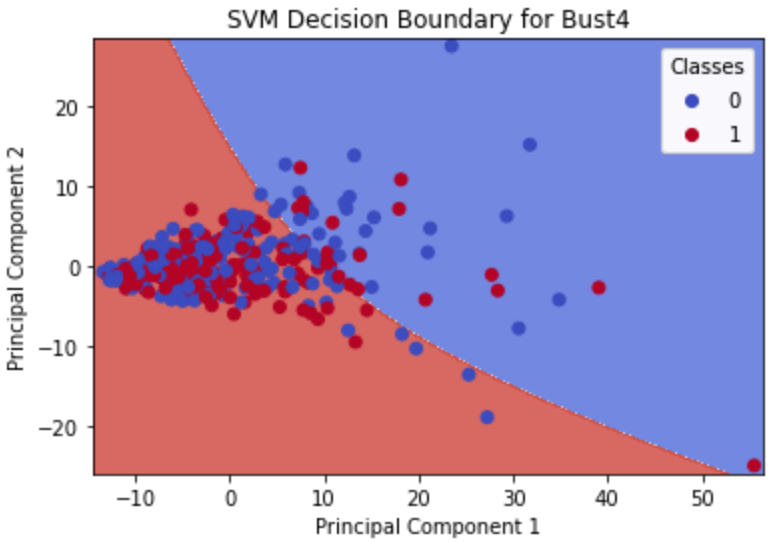


Bust 3 does quite well here as well when fed 150 players ~80%. With 150 players again we must worry about underfitting. Bust 4 label does the best when fed the most players 400 at ~62%. Let's take an exploratory look at this ~80% model by plotting this RF models features below:

Rank	Feature	Importance Score
1	hard	0.0077
2	game	0.0075
3	ball	0.0069
4	post	0.0063
5	look	0.0062
6	jumper	0.0061
7	strength	0.0059
8	go	0.0055
9	year	0.0053
10	point	0.0052

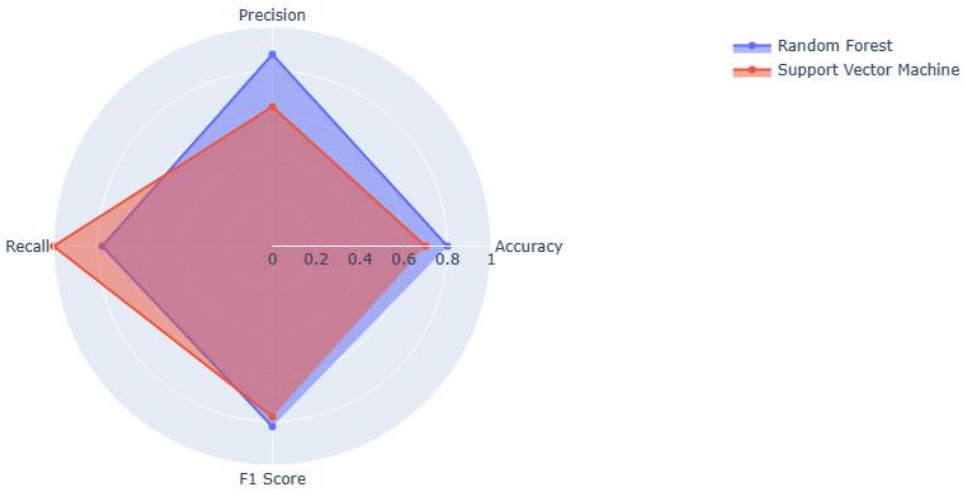


Some interesting features, “hard” may indicate a players struggle or a scouts difficulty to discern skill. The accuracy of this model is elusive, currently the testing size is 10% with 80% accuracy. When raised to 20% the accuracy drops drastically to 50%. When, 15% then 69% accuracy. Is this another sign of over-fitting? Feature selection could be very important here. Including irrelevant features or words can lead to overfitting. Let’s next take a look at a SVM model that takes 400 players. One way to examine an SVM model is a decision boundary plot.



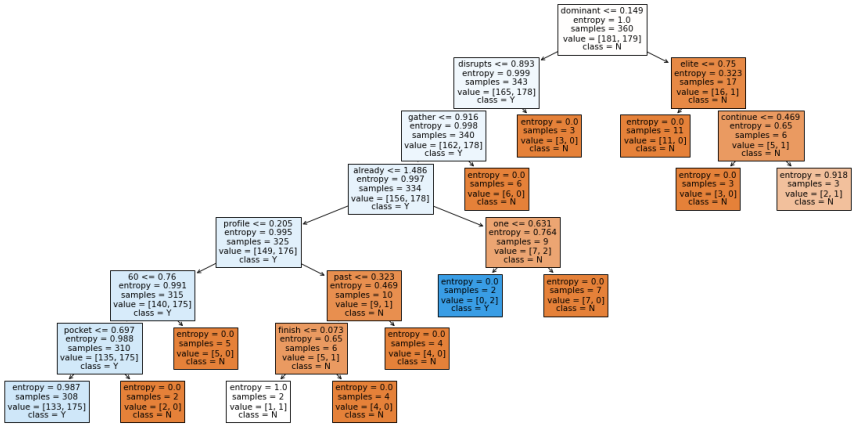
1 is a bust in the plot above. Results show the disparity between predictions as the red bust side has many more dots appearing on that side of the boundary.

Comparing these latest models can be illustrated below on a spider chart to see how they are predicting from statistics derived from a confusion matrix.



In the results shown above, the SVM model achieved a higher recall (it correctly identified all actual positives) but had lower precision compared to the Random Forest model. The Random Forest model had a slightly higher precision but a lower recall. The F1 score, which considers both precision and recall, was similar for both models and suggests that they have a similar overall performance.

To visualize how the models may be making decisions, decision trees are a useful model to attempt which lets us peek behind the curtain. After trial and error in an attempt for good accuracy, 400 players and 65% accuracy were achieved for a decision tree finely tuned tree shown below: (max_depth=7, min_samples_split=3, min_samples_leaf=2, max_features='sqrt', criterion='entropy', class_weight=None, splitter='random', random_state=42):

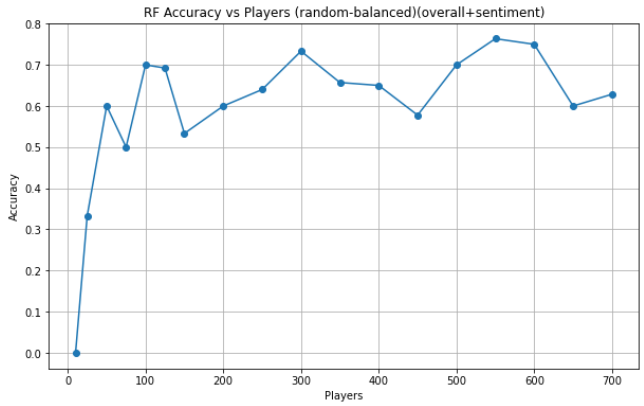


Important Features	Value
dominant	0.23083507181890603
one	0.13385869319303556
gather	0.12655817905005867
profile	0.1265280238096473
disrupts	0.06204258038593006
already	0.05247041103292486
pocket	0.04691915165915066

Results show key words from the scouting report that lead towards a bust decision. Dominant appears as the biggest deciding factor that plays a big role in separating the different

classes in the decision tree. The importance values are relative, so you can compare the values to see which features provided more predictive power compared to others. But the exact values don't have an exact interpretation.

Some of the most accurate and biggest models come from supplementing the scouting report data with numerical metrics. Below is how RF is performing when adding overall rating data and sentiment scores derived from the scouting reports:



As you can see above, we can reach 600 players and keep above 70% accuracy. Supplementing the data proves a useful way to enlarge the model and keep accuracy, but does still tail-off at 650-700 players likely due to overfitting.

Finally, let’s try to apply this model by making a prediction using the above model on some players drafted this year. They haven’t played yet in the league so why? Well, we can evaluate how the model does over the year by watching these players play for fun. Below is a sampling of players and whether the model predicts them as a bust:

Player	Predicted Bust
Victor Wembanyama	N
Jalen Pickett	Y
Jalen Wilson	Y
Scoot Henderson	Y
Julian Strawther	Y
Leonard Miller	N
Jaime Jaquez Jr	Y
Kris Murray	N
Marcus Sasser	Y
Isaiah Wong	Y
Trayce Jackson-Davis	Y
Mouhamed Gueye	Y
Sidy Cissoko	Y
Mojave King	Y

Sentiment Analysis.

TF-IDF Vectorizer Results.

Figure 35 portrays the initial 30 features or words from the processed vocabulary, derived through the TF-IDF vectorizer applied to the reviews data is presented. It also showcases the first 10 sample TF-IDF scores associated with each feature or word, after the meticulous cleansing and pre-processing steps.

```
First 30 Vocabulary Words:
['aac' 'aahs' 'aaron' 'aau' 'abandoned' 'abilities' 'abilitiesplays'
 'ability' 'abilitydoesnt' 'abilties' 'able' 'abnormal' 'abort'
 'aboveaverage' 'abovetherim' 'absent' 'absolute' 'absolutely' 'absorb'
 'absorbing' 'abundance' 'abuse' 'abused' 'abysmal' 'academic'
 'academically' 'academics' 'academy' 'acb' 'acc']

First 10 Sample Vectorization Score Columns:
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.06024411 0.         0.        ]
 [0.         0.         0.         ... 0.0635197  0.         0.        ]
 ...
 [0.         0.         0.         ... 0.02719611 0.         0.        ]
 [0.         0.         0.         ... 0.03215775 0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

Figure 35 – First 30 Vocabulary Words/Features and First 10 Vectorization Score Columns

Model 1: Random Forest Ensemble (3 Cross Validation) Results:

While predicting sentiment for the NBA Drafts data, several different scenarios were tested. Given that the sentiment scores were less skewed for the “Weaknesses” evaluations performed by the scouts and analysts, all models were applied the “Weaknesses_Preprocessed” variable. In scenario 1, the entire dataset containing 1042 rows was used. In scenario 2 a composite of records containing 220 top positive scores, 220 top negative scores, and 130 (all that was available) neutral scores were used. Scenario 3 was made of only the top 200 most negative scores and top 200 most positive scores. For scenarios 1 and 2, three categorical labels were used to classify the sentiment scores. For neutral scores, the label 0 was assigned. For negative and positive scores, the labels 1 and 2 were assigned respectively. Scenario 3 only had positive and negative labels which were classified as 0 for negative and 1 for positive sentiments.

Model 1.1 Random Forest 3CV Scenario 1 Results:

The sentiment classification model achieved an overall accuracy of approximately 65.07%, indicating that it correctly predicted the sentiment (positive, negative, or neutral) of the "Weaknesses_Preprocessed" analysis for about 65.07% of the player profiles in the dataset, on average. This suggests that the model performs reasonably well in categorizing sentiments. However, when examining the fold-wise accuracies, some variations across different subsets of the data were observed.

In the first fold, the model achieved an accuracy of around 64.39%, showcasing its ability to perform adequately on one portion of the dataset. In the second fold, the accuracy was slightly lower, at approximately 64.03%, suggesting consistent but not significantly improved performance. The third fold yielded the highest accuracy among the three, at about 65.70%. While the model's performance is relatively stable across the folds, it's important to note that there is still room for enhancement. Further inspection of each classification type shows that the

model performs well for Label 2 but struggles with Labels 0 and 1, particularly in terms of recall (Figure 36).

	precision	recall	f1-score	support
Label0	1.00	0.03	0.06	34
Label1	0.83	0.08	0.14	66
Label2	0.65	0.99	0.79	177
accuracy			0.66	277
macro avg	0.83	0.37	0.33	277
weighted avg	0.74	0.66	0.54	277

```
Sentiment Classification Model Accuracy: 0.6507177033492823
Fold 1 Accuracy: 0.6438848920863309
Fold 2 Accuracy: 0.6402877697841727
Fold 3 Accuracy: 0.6570397111913358
```

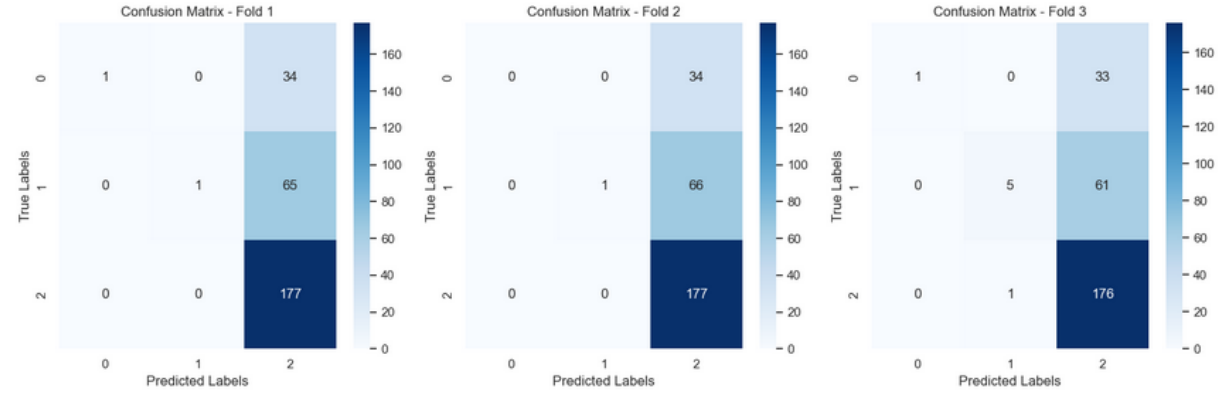


Figure 36 – Results and Confusion Matrix Model 1.1 Random Forest 3CV Scenario 1

Model 1.2 Random Forest 3CV Scenario 2 Results:

The results for Random Forest scenario 2 indicate that the distribution of sentiment labels in the dataset is somewhat imbalanced, with Label2 having the highest count (220 samples), followed by Label1 (220 samples), and Label0 with the lowest count (130 samples). This imbalance could impact the model's performance, especially for the minority class (Label0). The imbalance for the neutral class is due to the limitations in availability of neutral

classifications. In this study, neutral labels were assigned to scores ranging from 0 to 0.6. However, most of the reviews, analysis and commentary provided by the NBA scouts and analysists were polarized by being strongly positive or strongly negative, thus making the neutral classification a challenge for the models overall.

The complete accuracy of the sentiment classification model for this sample data is 0.5175, indicating that the model correctly predicts the sentiment label for approximately 51.75% of the samples. However, accuracy alone might not provide a complete picture, given the class imbalance. Fold-wise Accuracy: Examining the fold-wise accuracy, we see that Fold 1 and Fold 3 have the same accuracy of 46.05%, while Fold 2 has a slightly lower accuracy of 44.08%. These variations in fold-wise accuracy suggest some inconsistency in model performance across different subsets of the data.

The classification report provides a more detailed breakdown of the model's performance for each sentiment label. For Label0, the precision is high (1.00), indicating that when the model predicts Label0, it is usually correct. However, the recall is low (0.03), indicating that the model misses many actual Label0 instances. This results in a low F1-score (0.05) for Label0. For Label1, the precision is moderate (0.45), indicating that the model's predictions for Label1 are relatively accurate. The recall is higher (0.71), indicating that the model captures a significant portion of the true Label1 instances. This leads to a relatively higher F1-score (0.55) for Label1. For Label2, both precision and recall are moderate (0.47 and 0.47, respectively), leading to a balanced F1-score of 0.47.

	precision	recall	f1-score	support
Label0	1.00	0.03	0.05	36
Label1	0.45	0.71	0.55	59
Label2	0.47	0.47	0.47	57
accuracy			0.46	152
macro avg	0.64	0.40	0.36	152
weighted avg	0.59	0.46	0.40	152

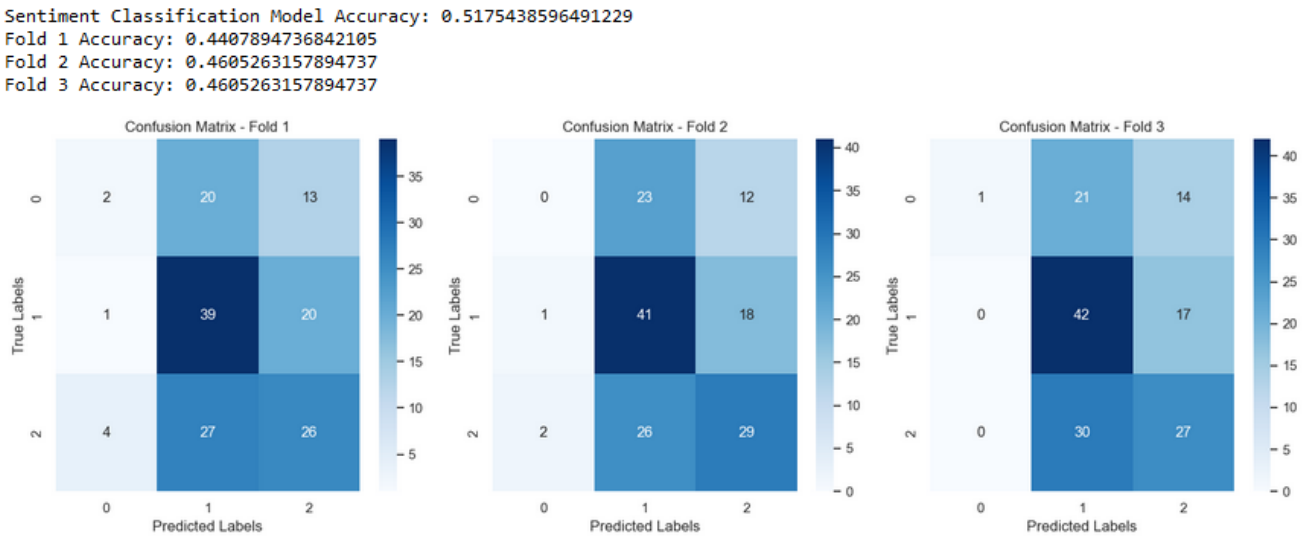


Figure 37 – Results and Confusion Matrix Model 1.2 Random Forest 3CV Scenario 2

In summary, the model performs reasonably well for Label1 and Label2, with higher precision and recall for these classes. However, the performance for Label0 is poor, primarily due to low recall. The overall model accuracy is influenced by the class imbalance, and it's important to consider the specific objectives of the sentiment classification task when evaluating these results. Further analysis and potentially addressing class imbalance are necessary to improve model performance.

Model 1.3 Random Forest 3CV Scenario 3 Results:

The results for Random Forest classification in the scenario of distinguishing between negative (Label0) and positive (Label1) appears to be relatively balanced. The overall accuracy of the sentiment classification model is quite high at 86.25%. This indicates that the model correctly predicts whether a player's "Weaknesses_Preprocessed" analysis is negative or positive for the majority of samples. The accuracy for each fold shows some variation but remains relatively high across all folds. Fold 1 and 2 have the highest accuracies at 85.05%,

followed by Fold 3 at 80.19%. The classification report provides detailed metrics for each sentiment label:

For Label0 (negative sentiment), both precision and recall are balanced at around 0.84 and 0.73 respectively. This indicates that the model accurately identifies negative sentiments and effectively captures most of the true negative instances. The F1-score for Label0 is also 0.78, reflecting a good balance between precision and recall.

For Label1 (positive sentiment), both precision and recall are similarly balanced at around 0.77 and .87. This suggests that the model performs well in identifying positive sentiments and captures most of the true positive instances. The F1-score for Label1 is also 0.82, indicating a good overall performance in distinguishing positive sentiments (Figure 38).

In summary, the Random Forest model performs well in distinguishing between negative and positive sentiments. It demonstrates relatively high accuracy and balanced precision and recall for both sentiment labels.

	precision	recall	f1-score	support
Label0	0.84	0.73	0.78	52
Label1	0.77	0.87	0.82	54
accuracy			0.80	106
macro avg	0.81	0.80	0.80	106
weighted avg	0.81	0.80	0.80	106

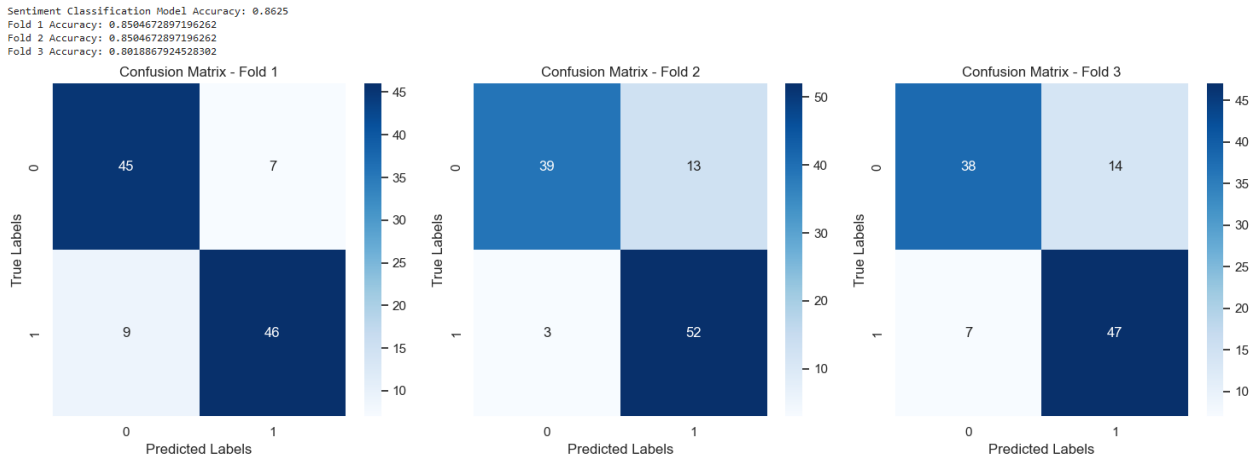


Figure 38 – Results and Confusion Matrix Model 1.3 Random Forest 3CV Scenario 3

Model 2. Supervised Vector Machines (SVM Linear Kernel + 3 Cross Validations)

Results

The results for SVM classification in the third scenario appear to be relatively balanced. The overall accuracy of the sentiment classification model is the highest thus far at 92.5%. This indicates that the model correctly predicts whether a player's "Weaknesses_Preprocessed" analysis is negative or positive for most of the samples. The accuracy for each fold shows some variation but remains relatively high across all folds. Fold 2 has the highest accuracy at 89.72%, followed by Fold 1 at 88.79%, and Fold 3 at 82.08%. The classification report provides detailed metrics for each sentiment label:

For Label0 (negative sentiment), both precision and recall are balanced at around 0.81. This indicates that the model accurately identifies negative sentiments and effectively captures most of the true negative instances. The F1-score for Label0 is also 0.82, reflecting a good balance between precision and recall.

For Label1 (positive sentiment), both precision and recall are similarly balanced at around 0.83. This suggests that the model performs well in identifying positive sentiments and captures most of the true positive instances. The F1-score for Label1 is also 0.82, indicating a good overall performance in distinguishing positive sentiments (Figure 39).

In summary, the SVM model with a linear kernel performs remarkably well in distinguishing between negative and positive sentiments. It demonstrates high accuracy and balanced precision and recall for both sentiment labels. These results suggest that the model is effective in classifying sentiment in the context of negative versus positive sentiments for the "Weaknesses_Preprocessed" analysis of NBA draft players.

	precision	recall	f1-score	support
Label0	0.81	0.83	0.82	52
Label1	0.83	0.81	0.82	54
accuracy			0.82	106
macro avg	0.82	0.82	0.82	106
weighted avg	0.82	0.82	0.82	106

```
Sentiment Classification Model Accuracy: 0.925
Fold 1 Accuracy: 0.8878504672897196
Fold 2 Accuracy: 0.897196261682243
Fold 3 Accuracy: 0.8207547169811321
```

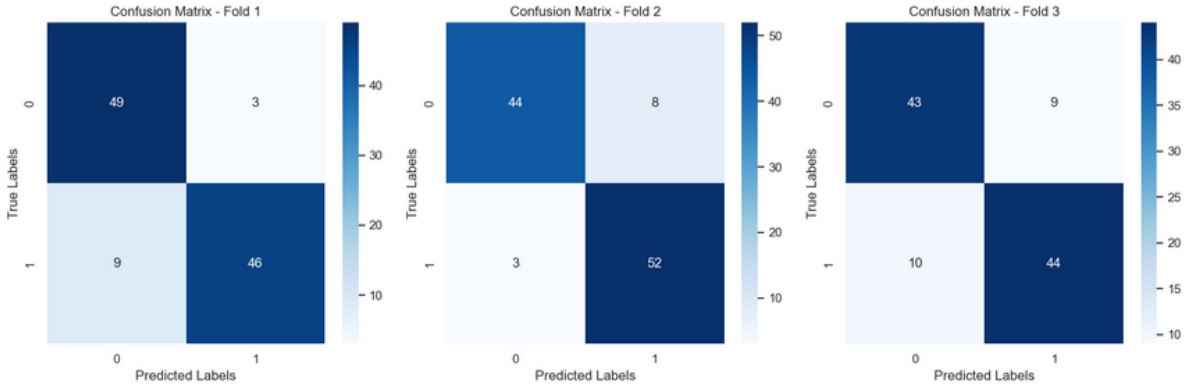


Figure 39 – Results and Confusion Matrix Model 2 SVM 3CV Scenario 3

Model 3: Supervised Vector Machines (SVM + Lemmatization + 3 Cross Validations) Results

Although no significant improvements were observed after the addition of lemmatization, model 3 produced impressive results. The detailed results for the SVM with a linear kernel and lemmatization in a 3-fold cross-validation scenario for sentiment classification, focusing on distinguishing between negative (Label0) and positive (Label1) sentiments are as follows:

Model Accuracy: The overall accuracy of the sentiment classification model is very high at 91.25%. This indicates that the SVM model with a linear kernel and lemmatization correctly predicts whether a player's "Weaknesses_Preprocessed" analysis is negative or positive for most samples.

Fold-wise Accuracy: The accuracy for each fold varies slightly but remains high across all folds. Fold 2 has the highest accuracy at 93.46%, followed by Fold 1 at 87.85%, and Fold 3 at 82.08% (Figure 40).

These results demonstrate that the SVM model with a linear kernel and lemmatization performs exceptionally well in distinguishing between negative and positive sentiments. It achieves a very high overall accuracy and balanced precision and recall for both sentiment labels, further highlighting its effectiveness in sentiment classification for the "Weaknesses_Preprocessed" analysis of NBA draft players.

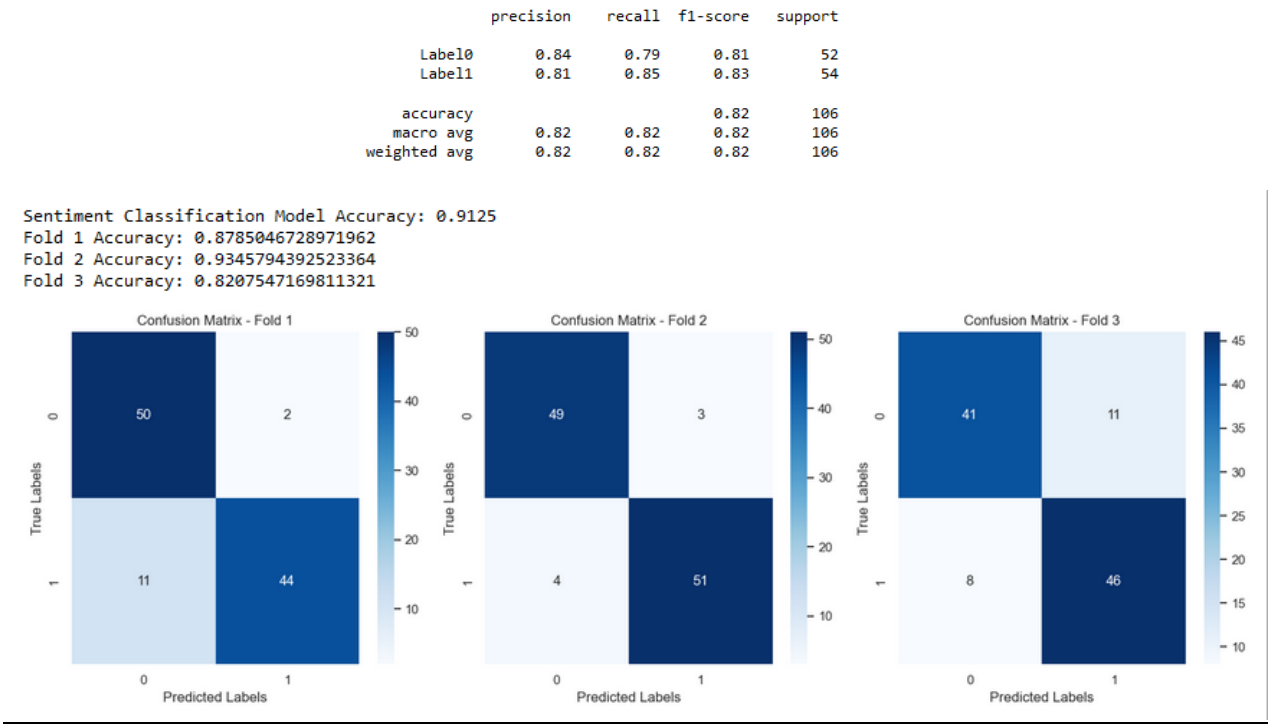


Figure 40 – Confusion Matrix heatmap for SVM 3CV + Lemmatization

Model 4: Supervised Vector Machines (SVM + Stemming + 3 Cross Validations + Bigrams) Results

The results for the SVM with a linear kernel, stemming, and bigrams in a 3-fold cross-validation scenario for sentiment classification produced a very high overall accuracy of the

sentiment classification of 85%. This indicates that the SVM model with a linear kernel, stemming, and bigrams correctly predicts whether a player's "Weaknesses_Preprocessed" analysis is negative or positive for most samples. The accuracy for each fold varies slightly but remains high across all folds. Fold 2 has the highest accuracy at 87.85%, followed by Fold 3 at 83.96%, and Fold 1 at 83.18% (Figure 41).

The classification report for Label0 shows a precision of 0.84, indicating that the SVM model with stemming and bigrams accurately identifies negative sentiments. The recall for Label0 is 0.83, suggesting that the model captures most of the true negative instances. The F1-score for Label0 is 0.83, reflecting a good balance between precision and recall. For Label1 the precision is 0.84, indicating that the model performs well in identifying positive sentiments with stemming and bigrams. The recall for Label1 is 0.85, suggesting that the model effectively captures most of the true positive instances. The F1-score for Label1 is 0.84, indicating a good overall performance in distinguishing positive sentiments with stemming and bigrams.

These results, although not as good as the results from the simple SVM with linear kernel, demonstrate that the SVM model with stemming, and bigrams performs exceptionally well in distinguishing between negative and positive sentiments. It achieves a very high overall accuracy and balanced precision and recall for both sentiment labels, further highlighting its effectiveness in sentiment classification for the "Weaknesses_Preprocessed" analysis of NBA draft players.

	precision	recall	f1-score	support
Label0	0.84	0.83	0.83	52
Label1	0.84	0.85	0.84	54
accuracy			0.84	106
macro avg	0.84	0.84	0.84	106
weighted avg	0.84	0.84	0.84	106

```
Sentiment Classification Model Accuracy: 0.85
Fold 1 Accuracy: 0.8317757009345794
Fold 2 Accuracy: 0.8785046728971962
Fold 3 Accuracy: 0.839622641509434
```

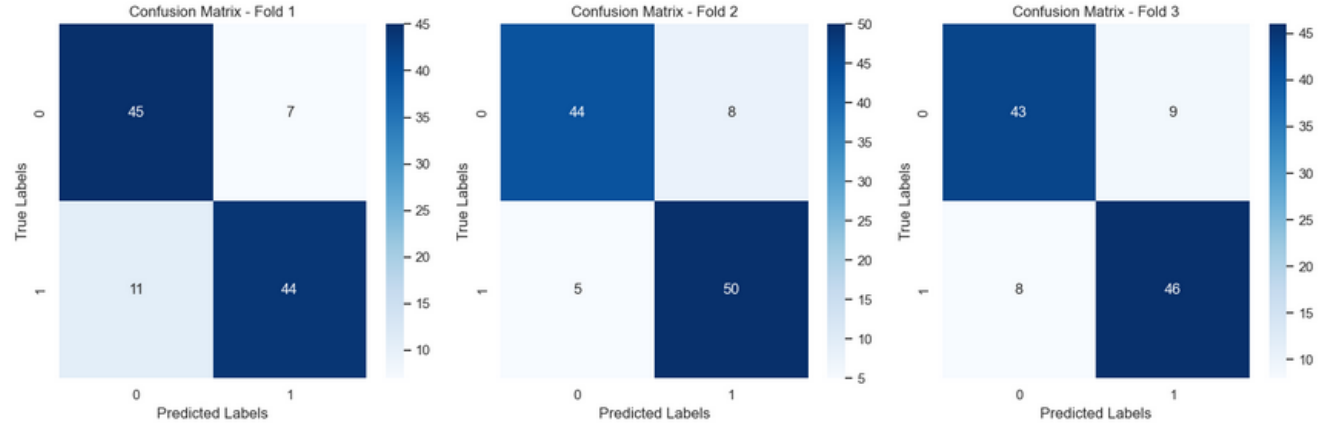


Figure 41 – Confusion Matrix heatmap for SVM 3CV + Stemming + Bigrams Results:

Sentiment Analysis Summary of Results:

Scenario 1: Original Data with all 1042 rows highly skewed towards positive sentiment

Model Type: Random Forest ensemble with 3 Cross Validations TF-IDF vectorization.

Accuracy: Achieved an accuracy of approximately 65%, indicating a moderately successful classification model.

Sentiment Labels: Distinguished between three sentiment labels: Label0 (negative), Label1 (neutral), and Label2 (positive).

Precision and Recall: Displayed varying levels of precision and recall for each sentiment label. Notably, Label2 (positive) had the highest precision and recall.

Scenario 2: Sample Data with 570 rows still skewed towards positive and negative sentiment scores (220 label2,220 label1, and 130 label0)

Model Type: Random Forest with TF-IDF vectorization and imbalanced sentiment labels.

Accuracy: Achieved an accuracy of around 46%, indicating a lower-performing model due to imbalanced data.

Sentiment Labels: Classified into three sentiment labels: Label0, Label1, and Label2.

Precision and Recall: Displayed imbalanced precision and recall, with Label2 (positive) having the highest values.

Scenario 3: Sample Data with top 200 negative and top 200 positive sentiment scores

Model Types: Random Forest with 3CVs and SVM with linear kernel and various preprocessing techniques (lemmatization, stemming, and bigrams).

Accuracy: Achieved consistently high accuracy across all preprocessing conditions, ranging from 82% to 91%.

Sentiment Labels: Focused on binary classification (negative vs. positive) due to preprocessing variations and data constraints.

Precision and Recall: Demonstrated balanced precision and recall for both negative and positive sentiments in all scenarios.

Overall Conclusions:

Random Forest vs. SVM: SVM consistently outperformed Random Forest in sentiment classification, particularly when distinguishing between negative and positive sentiments.

Imbalanced Data Challenge: Scenarios 1 and 2 struggled due to imbalanced sentiment labels, resulting in lower accuracy. This highlights the importance of addressing class imbalance in classification tasks.

Preprocessing Matters: Preprocessing techniques such as lemmatization, stemming, and bigrams did not significantly improve model accuracy and balance in SVM-based sentiment classification.

Binary vs. Multi-label Classification: The choice of binary (negative vs. positive) or multi-label (negative, neutral, positive) classification should depend on the research question and dataset characteristics. Binary classification with SVM yielded the best results.

In summary, SVM models with appropriate preprocessing techniques, especially when focusing on binary sentiment classification, proved to be the most effective in accurately classifying sentiment in the NBA draft data. The choice of preprocessing techniques and the handling of imbalanced data are critical considerations for improving sentiment analysis model performance.

Sentiment Analysis Scores vs. "Bust" Frequency Correlation Analysis:

To explore potential correlations between sentiment scores derived from NBA Draft analysis data and a player's "bust" status, we selected the top 50 players with the most positive sentiment scores and the top 50 players with the most negative sentiment scores for comparison. We then examined the distributions based on four different criteria used to classify a player as a "bust" or not.

The analysis revealed interesting insights. While some players with high positive sentiment scores were classified as "busts," the prevalence of "busts" was notably higher among players with top negative sentiment scores. This observation suggests that sentiment analysis of scouts' and analysts' reviews could offer valuable additional insights for predicting a draft player's likelihood of becoming a "bust" (Figures 42 and 43).

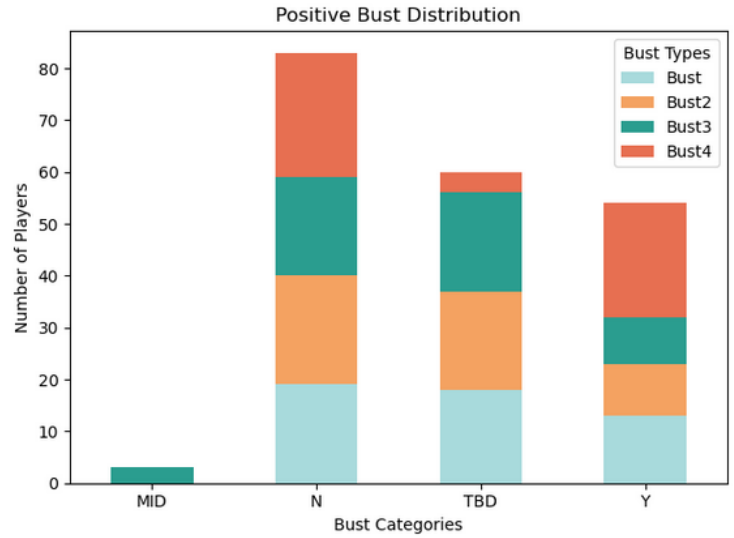


Figure 42 – Positive Bust Distribution – Top 50 Sentiment Scores from Weaknesses_Preprocessed

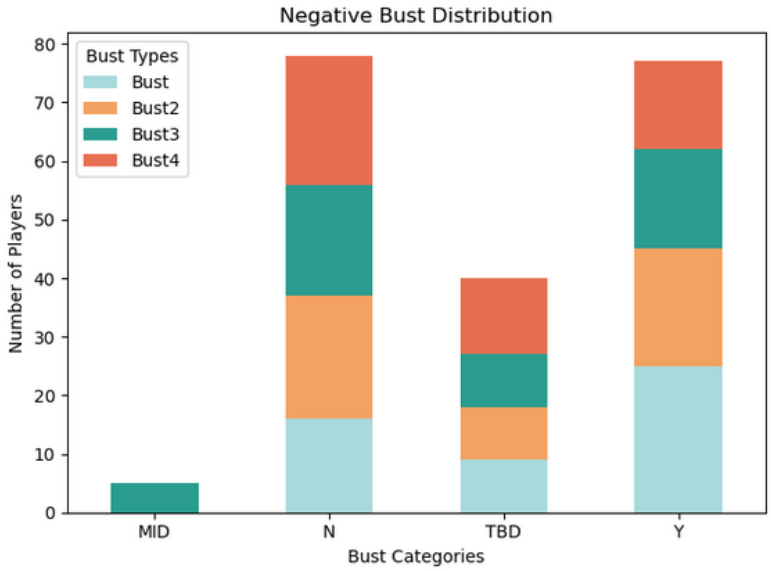


Figure 43 – Negative Bust Distribution – Top 50 Sentiment Scores from Weaknesses_Preprocessed

Model 5: Latent Dirichlet Allocation (LDA) – Topic Modeling Results

In Model 5 adopted several approaches to unveil hidden topics within NBA Draft player analyses and commentaries. In one approach, all reviews, and commentaries from the "Strengths," "Weaknesses," and "Outlook" columns were combined into a single text string and stored in a column named "Concatenated_Preprocessed." In the second approach, only commentaries from the "Weaknesses" column were used due to their balanced representation of negative and positive player assessments.

The primary goal was to analyze word co-occurrence patterns in these texts to identify latent topics and assess their prevalence across the dataset. This method allowed for a deeper understanding of the underlying themes in the "Weaknesses_Preprocessed" textual data. The same data cleaning and preprocessing techniques from the sentiment analysis were applied.

Initially, when the LDA model was applied to the combined "Strengths," "Weaknesses," and "Outlook" columns, it produced ten topics, but these exhibited repetitive themes and words, suggesting a need for further refinement. To optimize the model's performance and identify the most appropriate number of topics for the dataset, a grid search was employed. In this grid search, the LDA model was trained multiple times, each time with a different number of topics ranging from 5 to 50 possibilities (Figure 44).

The key to understanding how the grid search for coherence scores works lies in the coherence score itself. Coherence scores are a measure of how interpretable and semantically meaningful the topics are. Higher coherence scores indicate that the topics are more distinct and representative of the underlying themes in the text data. During the grid search, the LDA model is trained repeatedly with varying numbers of topics, and for each iteration, the

coherence score is calculated. The grid search systematically explores the entire range of topic numbers, and as the number of topics increases, the coherence score is tracked.

Figures 45 and 46 visually represent this process, with Figure 45 specifically showing the top 15 topic coherence scores. The blue line in the coherence score graph demonstrates how the coherence score changes as the number of topics increases. Analysts look for a point on this graph where the coherence score exhibits a sharp increase, followed by a decrease or stabilization. This point, often referred to as an "elbow" point, indicates the ideal number of topics. The sharp drop in the blue line (coherence scores) serves as a clear indicator of where this optimal number of topics can be found. This approach ensures that the selected topics are both informative and coherent, ultimately enhancing the utility of the LDA model's results for further analysis and interpretation.

Figure 46 displays the results from PyLDAvis. PyLDAvis is a Python library that provides an interactive visualization of topic models, particularly those generated using Latent Dirichlet Allocation (LDA). It helps users to explore and understand the topics within a text corpus by visualizing the relationships between topics and the terms that define them. The topic bubbles in the PyLDAvis represent topics as circles or bubbles on the screen. The size of each bubble corresponds to the prevalence of the topic in the corpus.

The inter-topic distance map arranges the topic bubbles in 2D space based on their similarity. Similar topics are placed closer to each other, allowing you to identify topic clusters. When a topic bubble is clicked, PyLDAvis displays a word cloud that shows the most relevant terms for that topic. The terms are selected based on their probability of appearing in the topic.

PyLDAvis also provides a topic matrix with a list of terms associated with each topic, along with their relevance scores and displays a bar chart for each topic, showing the distribution of that topic across the documents in your corpus. In the NBA Draft data case, topic

8 has the most prevalence and words like “good”, “game”, “ball”, “shot”, and “ability” have the highest frequencies.

```
Topic #1: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #2: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #3: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #4: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #5: good game ball great ability shot player nba shooting level
Topic #6: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #7: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #8: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #9: thing hits amazing character agile learn desire perfect mcdonalds competitor
Topic #10: thing hits amazing character agile learn desire perfect mcdonalds competitor
```

Figure 44 – 10 Topics from Concatenated Preprocessed Results

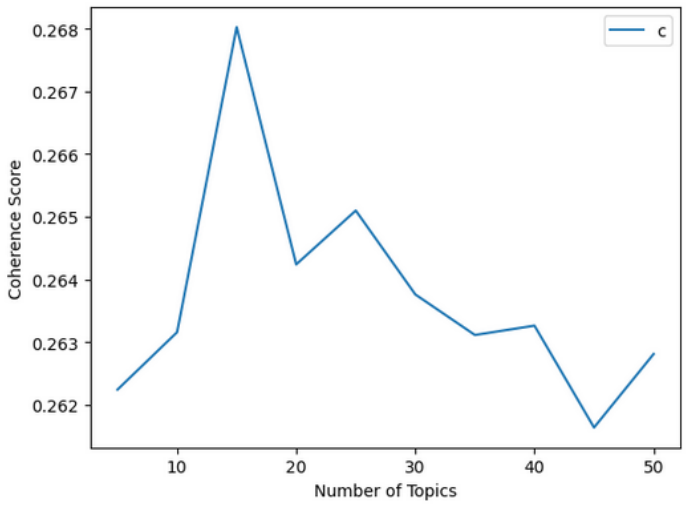


Figure 45 – 15 Topics from Concatenated Preprocessed Results Using Coherence Scores

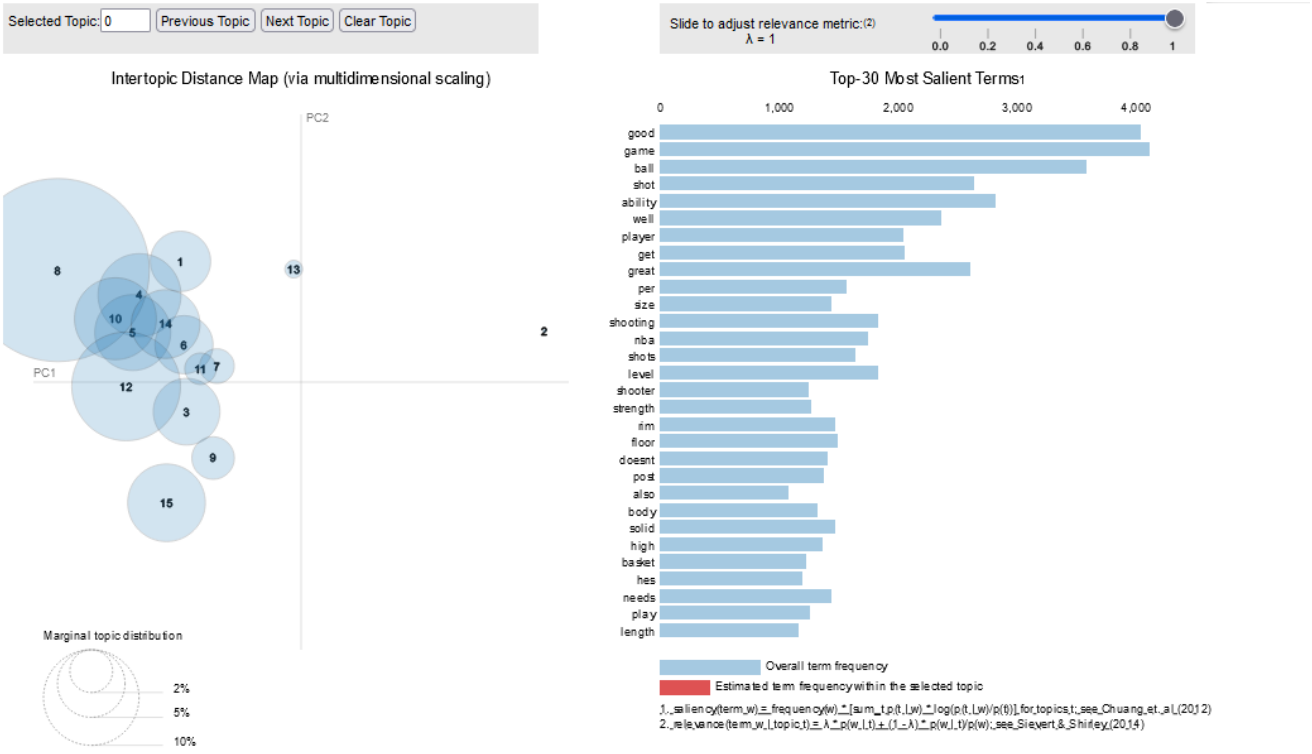


Figure 46 – LDA visualization (PyLDAvis) with 15 Topics from Concatenated Preprocessed Results Using Coherence Scores and Distribution of Most Frequent Words

For scenario 2, using only the reviews and commentaries from the "Weaknesses_Preprocessed" column, the first 10 topics were more distinguishable and thus offered more insights into what the main ideas were (Figure 47). The application of a coherence scores search grid yielded 25 total topics which are shown in Figure 48 along with top words and scores.

```
Topic #1: battle load distance exploit williams armour incident smith underdeveloped jones
Topic #2: battle load distance exploit williams armour incident smith underdeveloped jones
Topic #3: smith armour association distance 2018 nbpa favor clip creativity load
Topic #4: battle load distance exploit williams armour incident smith underdeveloped jones
Topic #5: shot need game lack ball time nba player doesnt level
Topic #6: battle load distance exploit williams armour incident smith underdeveloped jones
Topic #7: battle exploit williams facilitator underdeveloped league weakness appears scorer playing
Topic #8: incident load distance fight credit risk williams court carry deliberate
Topic #9: 69 pushed injury upper question size body strength doesnt battle
Topic #10: jones flash skillset plenty championship 18 12 deliberate minute vertical
```

Figure 47 – 10 Topics from Weaknesses_Preprocessed Results

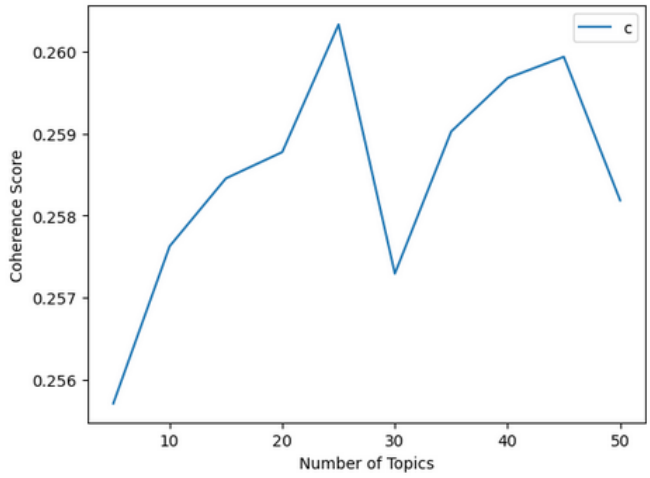


Figure 48 – 25 Topics from Concatenated Preprocessed Results Using Coherence Scores

The PyLDAvis visualization below highlights that for the second scenario, out of the 25 topics found, the distance between bubbles is much smaller when compared to the first scenario. This indicates the degree of similarity or dissimilarity between topics. Specifically, it represents the inter-topic distance, also known as the topic distance map. When two bubbles are positioned close to each other on the PyLDAvis visualization, it suggests that the topics they represent share similar terms and are more related to each other. Conversely, if two bubbles are farther apart, it indicates that the topics they represent are less similar and have fewer overlapping terms. In other words, topics that are close to each other may indicate subtopics or themes within a broader topic, while isolated topics might represent unique or niche themes within the corpus.

For the second scenario the words like “shot”, “need”, “game”, “lack”, “get”, “time” and “player” are the most frequent.

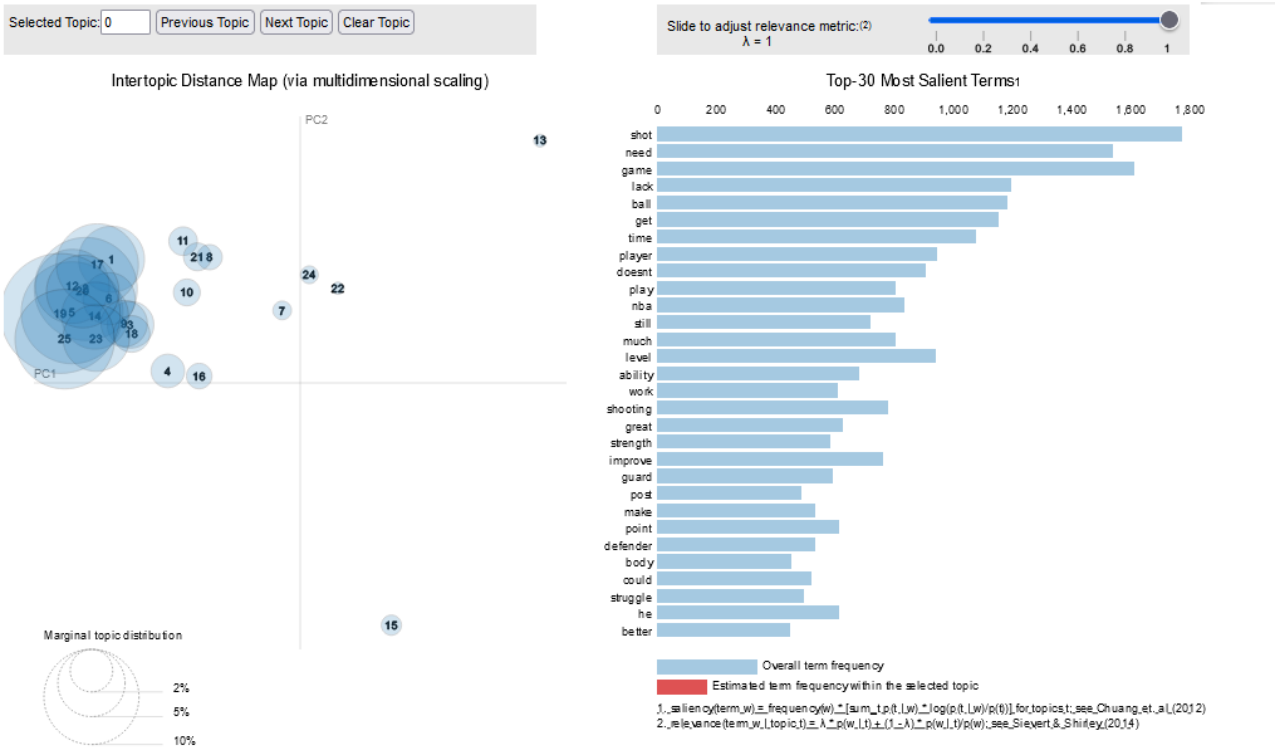


Figure 49 – LDA visualization (PyLDAvis) with 25 Topics from Concatenated Preprocessed Results Using Coherence Scores and Distribution of Most Frequent Words

Next, the overall top 10 words from each of the 25 topics observed from the “Weaknesses” analysis was plotted in a series of mini wordClouds as shown in Figure 50.

Figure 50 – WordClouds per Topic – top 10 Words from Weaknesses_Preprocessed

The heatmap below (Figure 51) provides an overview of topics with the most significant overlapping coherence scores. In this heatmap, darker colors signify stronger relationships between the topics, indicating a higher degree of word similarity and thematic correlation. For instance, Topics 7, 13, and 15 serve as illustrative examples of topics exhibiting pronounced correlations among their constituent words, suggesting shared themes and subject matter.

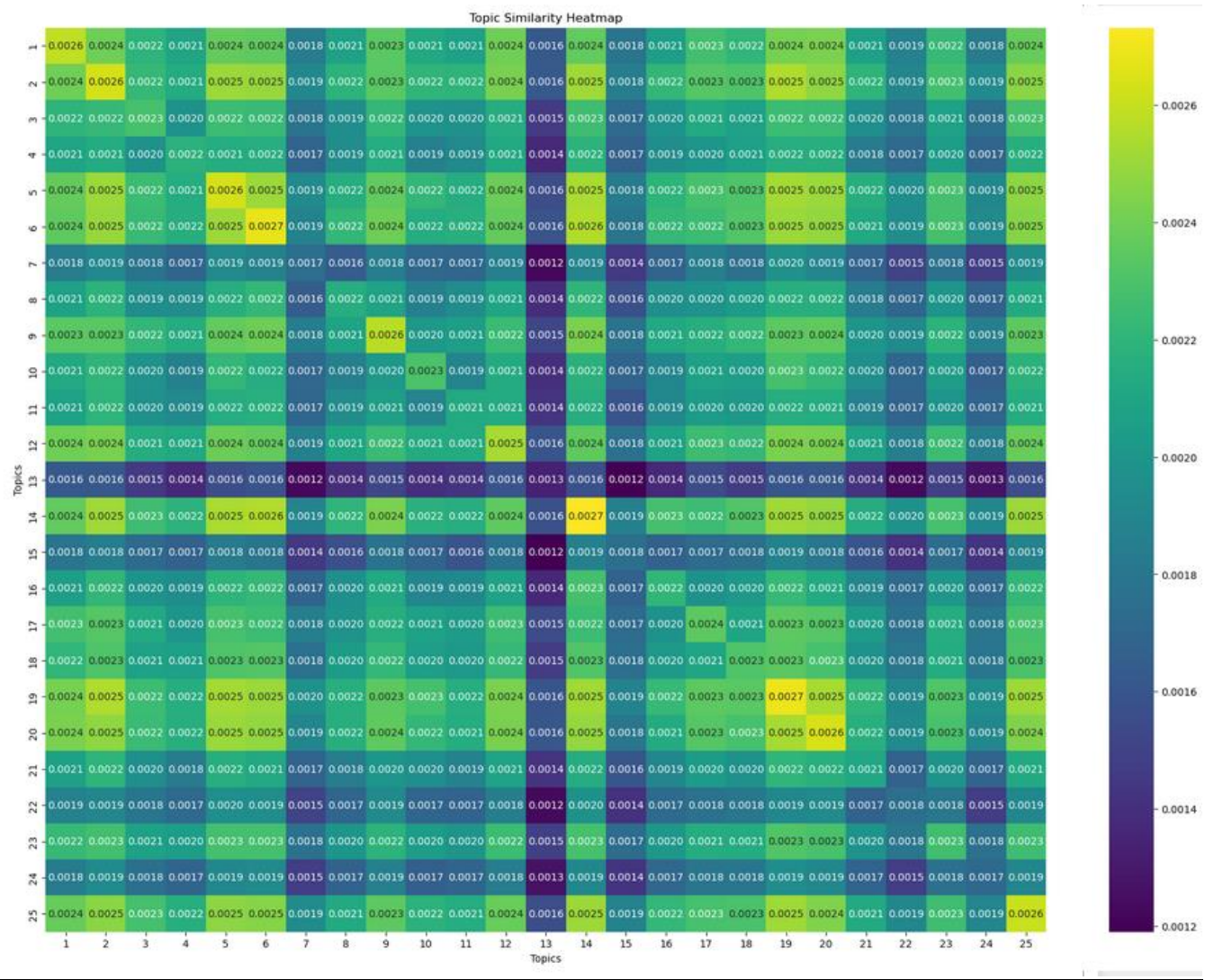


Figure 51 – Coherence Scores Heatmap of 25 Topics from "Weaknesses_Preprocessed

LDA Summary of results:

The best version of the model identified 25 distinct topics based on the "Weaknesses_Preprocessed" text data. A few key takeaways can be analyzed from these topics:

Common Themes: Several topics revolve around common themes like the need to improve various aspects of the players' game, such as shooting, ball-handling, and overall skills. These topics also touch upon the player's physical attributes, including strength and athleticism.

Game and Skills: Topics #1, #2, #3, and #6 focus on aspects related to the players' performance in the game, including shooting, scoring, and defense. It appears that the player is analyzed in terms of their skill development and gameplay.

Strength and Physical Attributes: Topics #4, #9, and #15 emphasize the importance of strength, body control, and physicality in the players' game. This suggests a focus on the players' physical attributes and their impact on performance.

NBA and Professional Level: Topics #2, #3, #5, #7, and #17 mention the NBA and professional basketball, indicating that the players' readiness for the NBA or a similar level is a recurring theme.

Shooting and Shooting Range: Topics #10, #12, #19, and #20 highlight the players' shooting abilities and range, including three-point shooting and midrange skills. Improving shooting seems to be a key focus.

Ball Handling and Passing: Topics #13, #16, #21, and #22 touch on the players' ball-handling, passing, and playmaking abilities. These topics suggest an evaluation of the player's ability to distribute the ball and make plays.

Naïve Bayes.

Across all 4 steps in the cleaning process, the models built using the ‘Bust4’ label had the highest accuracy on average for all the different model variations, although none were very significant. The mean accuracies for all models during the pre-clean, post-clean, and freq-removal steps using ‘Bust4’ was about 58%, while the models during the cor-removal step was slightly higher at 59%. Figure 52 displays the accuracies for all labels across the 4 cleaning steps:

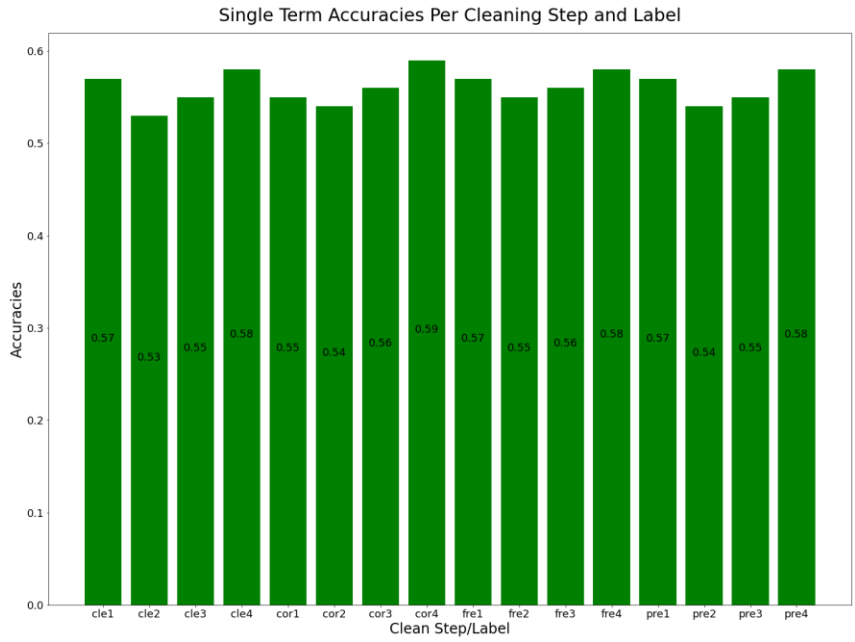


Figure 52

The highest accuracy for models using the 'Bust4' label was built using CV, just the str text, before any cleaning was done, with MNB, for an accuracy of about 62%. Figure 53 shows a confusion matrix created from the predictions made by a model using all of the above listed variations:

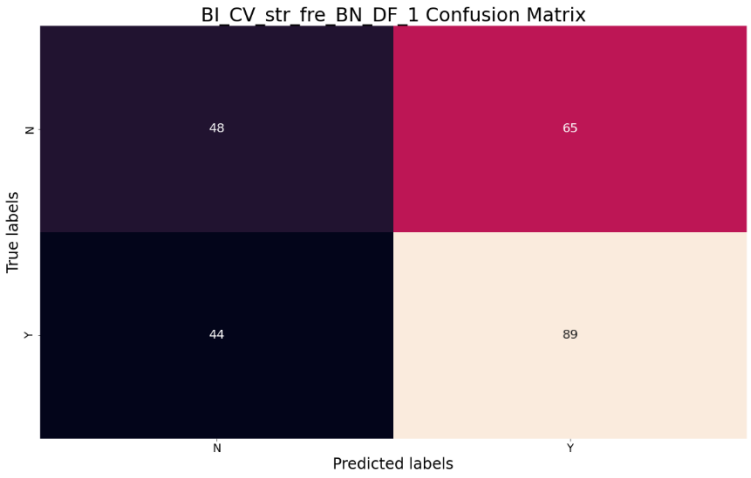


Figure 53

The label that produced the third highest accuracies, on average, was label 3, with an average accuracy of about 55% across all models that were built using it. The model using label 3 that had the highest accuracy was created using CV, str text, freq_removal, and BNB, with an accuracy of about 56%. Figure 54 shows a confusion matrix created from the predictions made by a model using all of these variations:

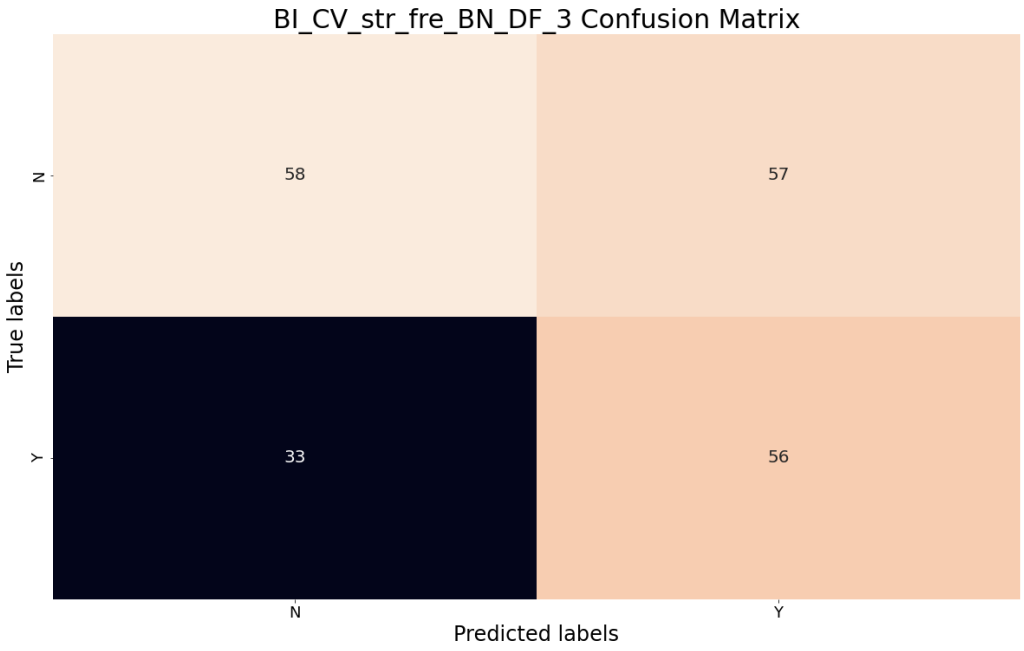


Figure 54

Finally, the label that produced the lowest accuracies, on average, was label 2, with an average accuracy of about 54% across all models that were built using it. The model using label 2 that had the highest accuracy was created using CV, str text, fre_removal, and MNB, with an accuracy of about 57%. Figure 55 shows a confusion matrix created from the predictions made by a model using all of these variations:

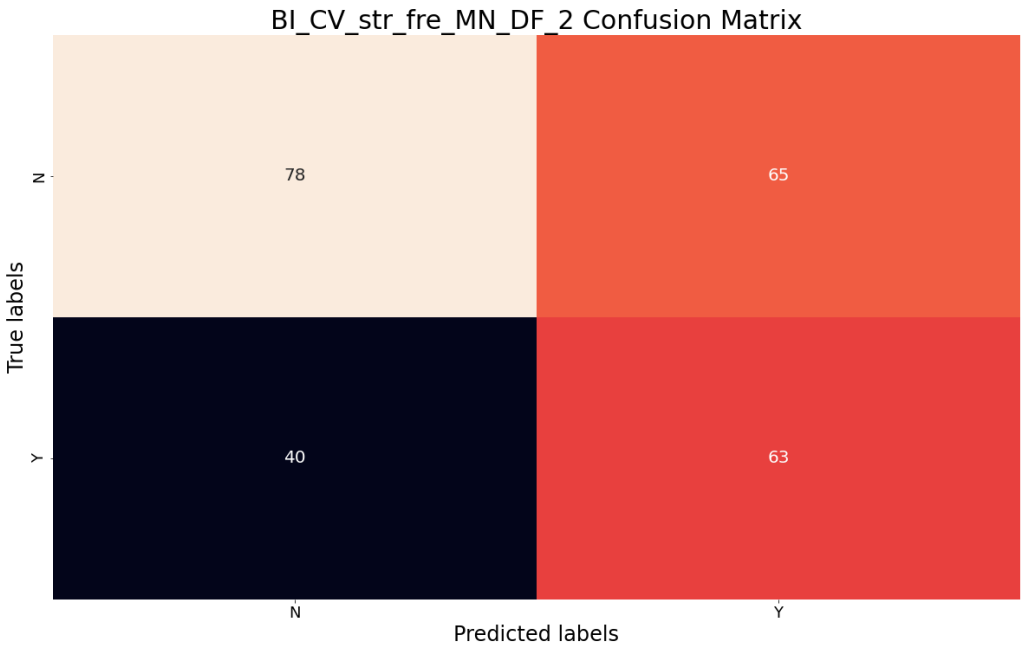


Figure 55

Long Term Short Term (LSTM) and Convolutional Neural Network (CNN) Deep Learning Models.

Model Results.

LSTM Model.

The LSTM model returned an accuracy of 52.59% at epoch 2, at which training and validation loss leveled out and did not improve or get worse (Figure 56).

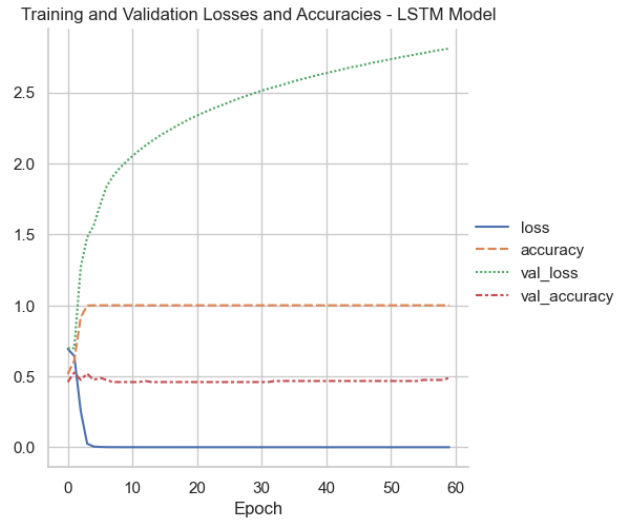


Figure 56

CNN Model (plain).

The plain CNN model (no embeddings) resulted in an accuracy of 60.00% at epoch 15, at which the training and validation loss leveled out at this point, neither improving or getting worse; similar to the behavior of the LSTM model (Figure 57).

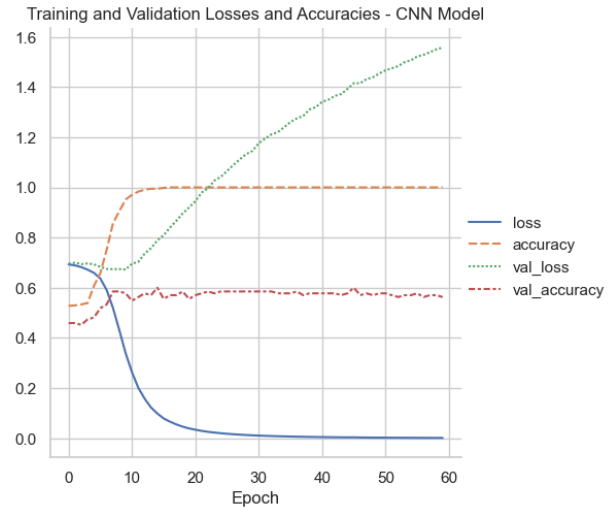


Figure 57

CNN Model (GloVe embeddings).

Once Glove embeddings were introduced to the CNN model, the behavior of the model during training changed significantly. The CNN model with GloVe embeddings returned an accuracy of 62.96 at epoch 28, however, it began to diverge at this epoch and the accuracies lessened at each subsequent epoch (Figure 58).

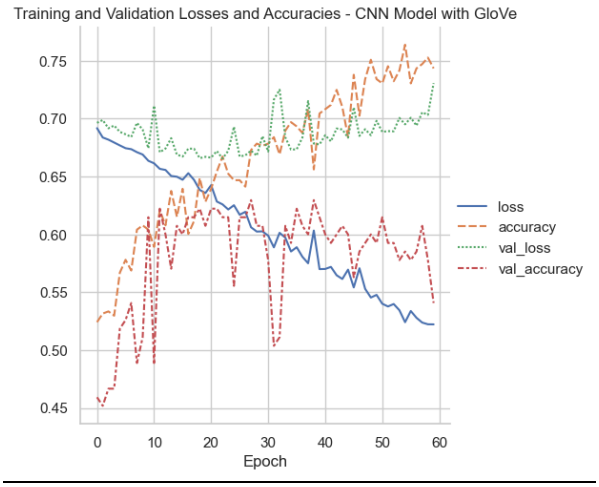


Figure 58

CNN Model (Word2Vec embeddings).

A similar behavior was observed in the CNN model when Word2Vec embeddings were introduced. This model returned an accuracy of 63.70% at epoch 29 (a similar point in time as the GloVe CNN model), and diverged in a similar pattern to the GloVe embedded model, but not as radically (Figure 60).

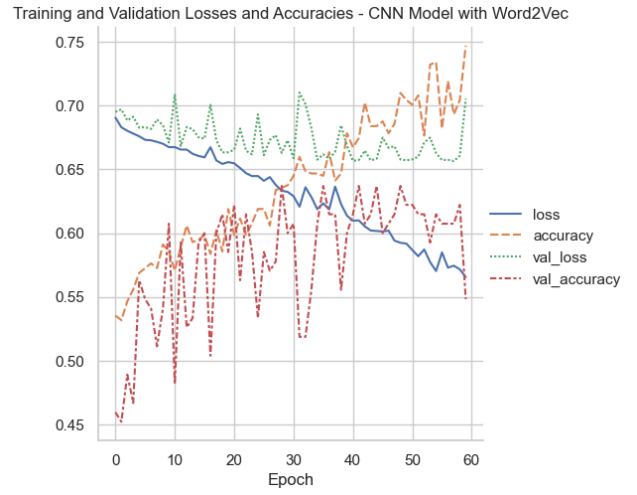


Figure 60

Discussion on Deep Learning Models.

The choice of multiple neural network architectures (LSTM and CNN) and embedding mechanisms (Word2Vec and GloVe) underscores a comprehensive attempt to address the prediction challenge. While LSTMs are inherently designed for sequential data like text, the efficacy of CNNs, traditionally reserved for image data, has been established in capturing localized textual patterns. The inclusion of pre-trained embeddings, such as Word2Vec and GloVe, arms the models with an intricate understanding of word semantics, potentially enhancing generalizability.

The behavior of the divergence in the CNN models with GloVe and Word2Vec embeddings (Figures 61 and 62) could be explained by an imbalance in the dataset. However, after conducting an ad-hoc analysis of the training/validation patterns in both of these models using cross-fold validation, it was concluded that this possibility was unfounded.

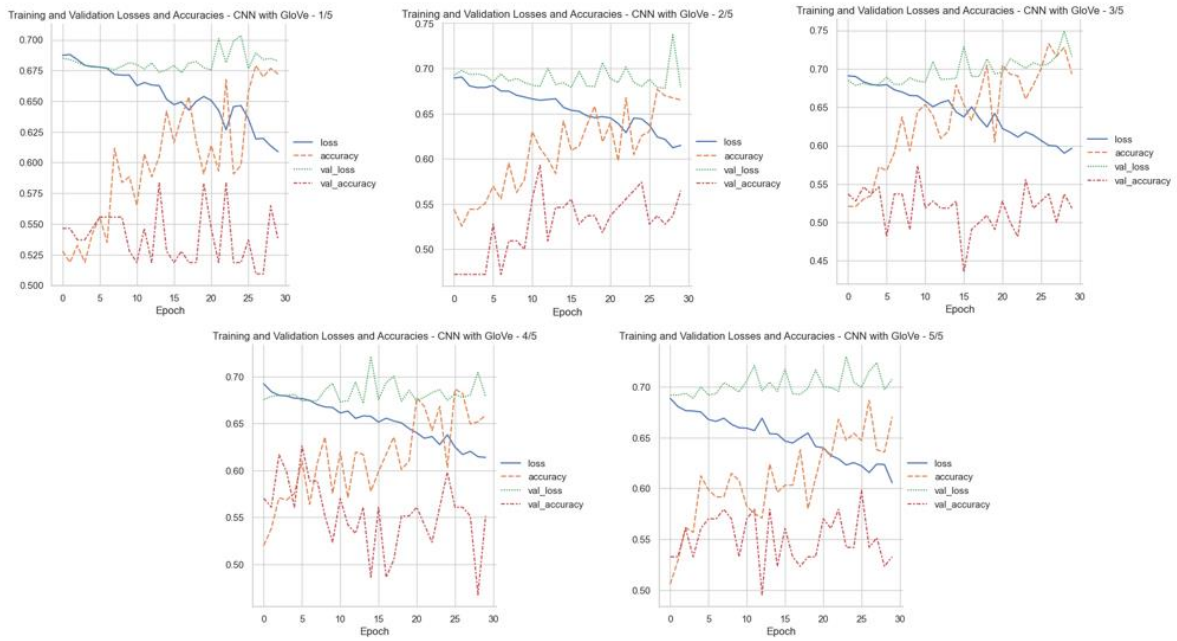


Figure 61

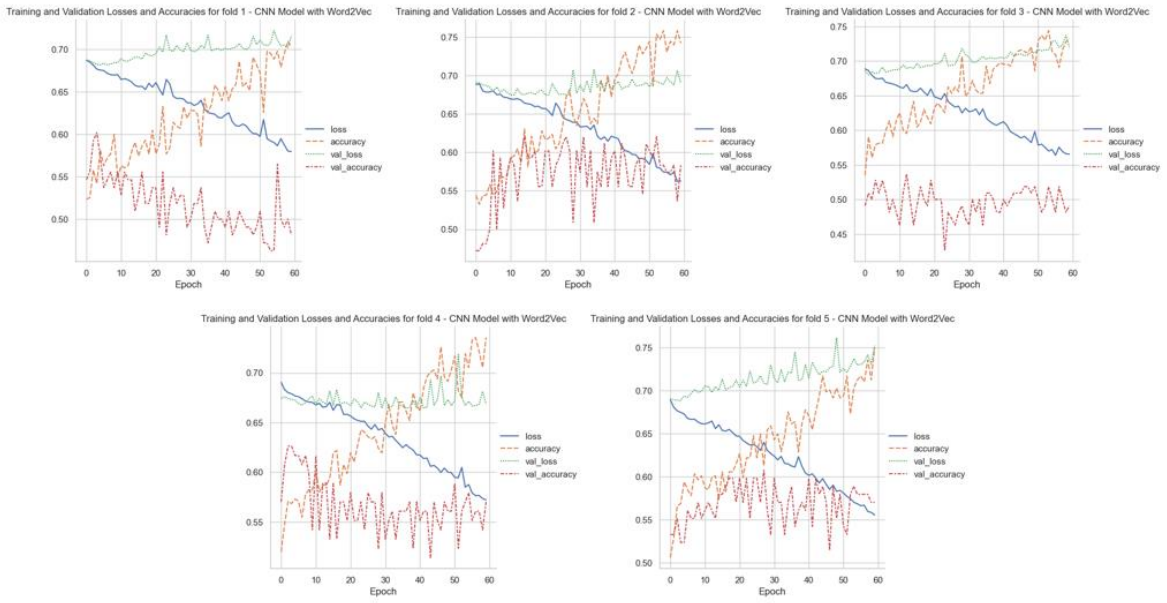


Figure 62

From the above figures, it can be concluded that the training/validation behaviors were remarkably similar across all folds, indicating that the datasets are, indeed, balanced.

While the models demonstrated promise, the achieved accuracy, ranging between 55% and 65%, invites introspection. The unpredictability inherent in forecasting a player's career trajectory based on subjective evaluations, potential data imbalances, and the presence of noisy data could influence outcomes. However, with cross-fold validations of the models, the results were not found to be imbalanced in a way that would reflect that the validation sample was represented in an unbalanced way.

CONCLUSION

The world of professional basketball scouting has always been a blend of art and science. In recent years, we've witnessed a significant shift towards a more analytical approach, emphasizing data-driven insights. This study, while comprehensive in its scope, serves as an introductory exploration into the vast potential that modern scouting methods offer. The insights gleaned underscore the dynamic balance required between traditional expertise, with its nuanced understanding of the game, and the precision of cutting-edge analytical tools. What emerges is a clearer picture of the future of player evaluation, especially within the critical juncture of the NBA Draft.

As we pave the way forward, one cannot help but acknowledge the myriad of untapped data sources that remain largely unexplored. From the pulse of social media sentiments, which can offer a window into a player's public perception, to intricate biometric data detailing physical responses during high-pressure situations, the reservoir of information is vast. Each of these sources, when harnessed appropriately, has the potential to revolutionize our understanding of

player capabilities. Integrating such diverse datasets promises richer, more layered insights, allowing for a nuanced evaluation that encompasses both athletic prowess and off-the-court dynamics.

The next frontier of sports analytics, we believe, lies in interdisciplinary collaboration. By forging partnerships with experts in fields like behavioral psychology, sociology, and even neuroscience, we can delve deeper into the psyche of players. It's not just about how a player shoots or defends, but also about how they react under pressure, their adaptability to changing game dynamics, and their interpersonal relationships within a team framework. By understanding these complex interplays, our scouting reports can evolve to be more holistic, capturing the essence of a player beyond just their statistical outputs.

With the rapid advancements in automated scouting methodologies, there emerges a parallel need for transparency and ethical considerations. As we harness algorithms and machine learning, it's of paramount importance to ensure that these tools are devoid of biases and offer a fair evaluation. Moreover, we must remember that at the core of every data point, every statistic, is an individual – a player with dreams, aspirations, and emotions. Our methods, while rigorous, should also be empathetic, acknowledging the human stories that unfold on the basketball court.

In conclusion, as we stand at this exciting crossroad in the realm of player scouting, the path ahead is rife with both challenges and opportunities. The hope is that with a judicious mix of traditional wisdom, innovative analytical techniques, and a dash of empathy, the future of sports analytics will not only be transformative but also deeply respectful of the athletes it seeks to evaluate.