

Out-Classing the NFL Draft: Using NLP and Machine Learning on NFL Scouting Reports to Predict Rookie Year Success

Sammy Muench, Gabe Schwartz, Jacqueline Kloner, Derin Acaroglu

Senior Capstone Project May 2025

Advisor: Professor Alva Couch

Table of Contents

Table of Contents.....	1
Introduction.....	2
Background and Motivation.....	2
Objective.....	2
Overview of Approach.....	2
Proof of Concept.....	3
Data Processing Pipeline.....	4
Data Collection.....	4
Prompt Engineering and LLM/NLP Utility.....	7
Scaling.....	7
Football Relevance.....	8
Output Processing.....	9
Finalized Dataset.....	11
Model Training and Evaluation.....	15
Defining Success.....	15
Model Specifics.....	15
Results and Discussion.....	16
Positional Summaries and Patterns.....	17
Offensive Guard.....	17
Defensive Back.....	20
Running Back.....	22
Feature Summaries and Success Traits.....	22
All-Position Summary and Draft Comparison.....	24
Limitations and Concerns.....	26
Challenges and Lessons Learned.....	27
Process.....	27
How can we make it more efficient.....	27
How can we incorporate more analysis.....	27
Product.....	27
How can we improve the quality of the product in the future.....	27
Challenges.....	28
Data Processing.....	28
Modeling.....	28
Conclusions and Future Work.....	29
Future Studies.....	29
References.....	30

Introduction

Background and Motivation

The NFL Draft is an annual event where professional American football teams pick players from college in inverse order - the weakest team goes first. Despite the months of preparation leading up to the draft, predicting long-term player success remains a major challenge. For example, only 27% of first-round wide receivers drafted between 2000 and 2019 signed a second contract with the team that drafted them. This means nearly three-quarters failed to meet expectations. Similar statistics hold across other positions, revealing a major gap in talent evaluation in the NFL.

While teams rely on a combination of film study, NFL combine performance, and in-person interviews, these methods remain largely qualitative and are often subjective. Factors such as team fit, coaching environment, and team schemes further complicate proceedings. However, the pre-draft scouting reports found on NFL.com provide a rich source of expert opinion that has historically gone unanalyzed pre-draft. If we can extract meaningful information from this text and link it to the draft, we may be able to improve many teams' draft day decisions.

Objective

The goal of this project is to investigate whether the language used in pre-draft scouting reports can help predict a player's future NFL performance. Specifically:

- Whether certain phrases or traits commonly used by scouts directly correlate with success in the NFL.
- Whether Natural Language Processing (NLP) techniques can transform these unstructured reports into structured features that are predictive.
- Whether machine learning models trained on this data can outperform, or match traditionally used draft analysis techniques.

By addressing these questions, we hope to provide a new tool that enhances existing scouting processes.

Overview of Approach

We developed a modeling pipeline that combines Natural Language Processing and player performance data. The approach is as follows:

- **Collecting Data**

We scraped pre-draft scouting reports from NFL.com for a variety of positions dating back to 2014. These reports contain written blurbs by NFL experts assessing each player's strengths and weaknesses.

- **Categorizing into Traits**

Using large language models (e.g., Google Gemini), we categorized the freeform text from scouting reports into position-specific traits. For instance, for wide receivers we looked for qualities like route running, hands, separation and for offensive lineman we scouted for pass protection. We labeled the traits as positive or negative depending on the phrasing of the text.

- **Defining Success**

To evaluate rookie-year performance, we used Pro Football Focus (PFF) grades, which score players on a 0–100 scale based on snap by snap performance.

- **Modelling**

We wanted to predict whether a player's PFF score was above or below the median for their position (binary classification). Logistic regression was used as our baseline model, with cross-validation strategies accounting for year-based grouping to avoid any data leaks.

Proof of Concept

Due to data processing constraints, we limited our initial analysis to a subset of positions, including offensive guards (OG), centers (C), running backs (RB), cornerbacks (CB), and defensive linemen (DL). For each of these, we extracted 1500-2500 scouting profiles, depending on availability and completeness.

Our proof of concept focused on whether position-specific traits extracted from text correlated with a player outperforming the median PFF grade in their rookie year. Even in this simplified binary task, our models showed signs of success. Verbal cues in scouting reports (comments about play recognition or athleticism) were weakly but meaningfully correlated with higher performance grades.

While the small dataset and the obvious subjectivity in both text and evaluation metrics posed challenges, we concluded that the language used in scouting reports does contain valuable information. If scaled up, we thought that this approach could help teams identify “diamonds in the rough” and reduce the uncertainty in early-round picks.

Data Processing Pipeline

In order to consolidate all of our data processing and transformations into one place, we developed a data pipeline, which takes the raw data from NFL.com and converts it into our final feature set. With only two manually changed global variables at the top, we could rerun the entire pipeline for each new position/category set combination. Below, we explore how this data was collected in the first place; then, we explore our complex pipeline.

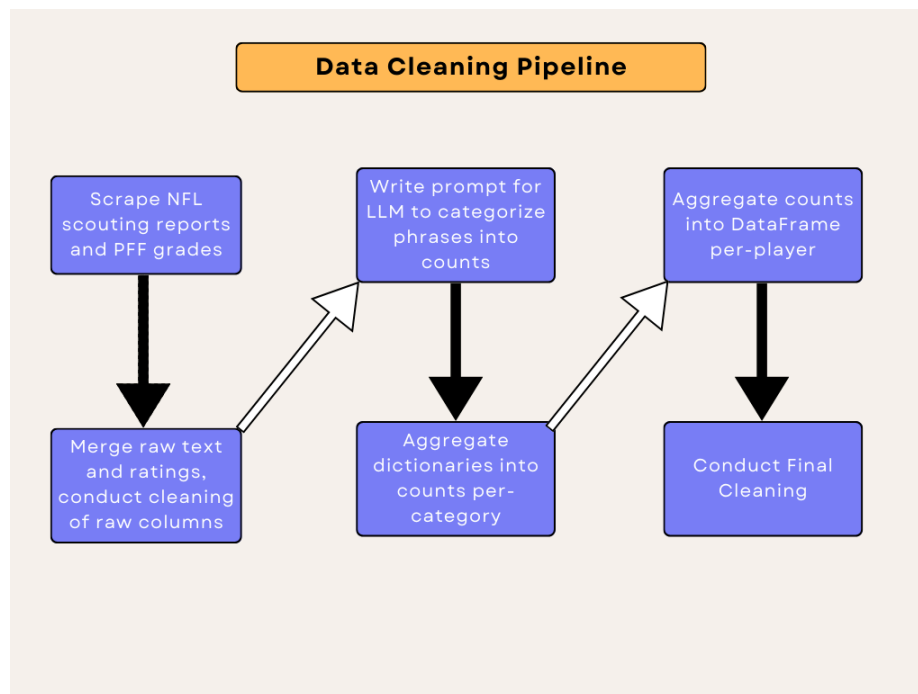


Figure 1: Comprehensive Data Processing Pipeline

Data Collection

We used two primary sources of data: NFL.com for our X-features, and Pro Football Focus (PFF) for our Y-response variable. PFF creates an industry-standard player-quality scoring metric, where the company employs analysts to give each player a score from -2 to +2 (in 0.5 increments) for each snap that they are in the game. These scores are then accumulated into a score from 0-100 over the course of a season, where 'average' is considered to be around 60. For context, in football, a snap is the act of the center giving the ball to the quarterback to start a play, so 'the number of snaps played' is another way of saying 'the number of plays during which a player was on the field'.

Fortunately, PFF allows users to download spreadsheets containing all of their seasonal player scores, but only behind a paywall. Our advisor, Dr. Alva Couch, was willing to sponsor us with the \$100 to access this data, which we are extremely grateful for.

The NFL.com data was not so simple. This required us to build a web-scraping algorithm using

BeautifulSoup, which perfectly grabbed the data we needed and stored it in a CSV. NFL.com is built in a very navigable way. There is a draft board page which shows all prospects in order of their own grading system with applicable filters. This page is shown below in Figure 2.

Filter

QB

College

Status

2025

Search by Last Name

Q

Reset filters









Player	Status	Position	Team	Class	Grade
 <div>Cameron Ward</div> <div>MIAMI</div>	Rnd 1, Pick 1	QB	Tennessee Titans	Senior	6.39
 <div>Shedeur Sanders</div> <div>COLORADO</div>	Rnd 5, Pick 6	QB	Cleveland Browns	Senior	6.30
 <div>Jaxson Dart</div> <div>MISSISSIPPI</div>	Rnd 1, Pick 25	QB	New York Giants	Senior	6.17
 <div>Tyler Shough</div> <div>LOUISVILLE</div>	Rnd 2, Pick 8	QB	New Orleans Saints	Senior	6.16
 <div>Quinn Ewers</div> <div>TEXAS</div>	Rnd 7, Pick 15	QB	Miami Dolphins	R-Junior	6.15
 <div>Jalen Milroe</div> <div>ALABAMA</div>	Rnd 3, Pick 28	QB	Seattle Seahawks	R-Junior	6.14
 <div>Kyle McCord</div> <div>SYRACUSE</div>	Rnd 6, Pick 5	QB	Philadelphia Eagles	Senior	6.13
 <div>Dillon Gabriel</div> <div>OREGON</div>	Rnd 3, Pick 30	QB	Cleveland Browns	Senior	6.10

Figure 2: NFL.com draft board

The URL for this exact setup is

<https://www.nfl.com/draft/tracker/prospects/qb/all-colleges/all-statuses/2025?page=1>. Note how all of the filters appear in the URL (e.g. qb, 2025). We had our web-scraper loop through a list of every possible position, inserting the position acronyms into the URL for each position. Inside of this first loop, we would run another loop which iterated from 2014 (the first year of data on NFL.com) to the current year minus 1 to avoid incomplete seasons, and insert this value into the URL as well. Inside of this year's loop, we would first scrape the number of pages of players that there were for that position, and then start a third loop which would replace the page number in the URL after parsing through the players. Each player in the list was stored with a link to their individual page, so the scraper added each url to a list instantiated inside of the first loop (by position).

Once the 2nd and 3rd loops had concluded for a specific position, the scraper would then open each URL in that position's stored list to gather data for each player, then output it to a CSV. To visualize this process for extra clarity, examine the diagram below:

[list of all positions]

Loop 1: Loop through positions in list:

[empty list of URLs]

Loop 2: Loop through every year from 2014 to now:

Go to URL, find number of pages for year/position combo

Loop 3: Loop through each page of players:

Add each player's URL to URL list

Conclude Loop 3

Conclude Loop 2

Loop 4: Loop through all URLs in list:

Go to URL

Scrape necessary data

Store in CSV labeled for current position

Conclude Loop 4

Conclude Loop 1

Figure 3: Web-Scraper Loop Diagram

Unfortunately, this was not a simple process. It did not take long to figure out that NFL.com changes the names of their HTML fairly often to prevent scrapers like ourselves from gathering their data too easily. This meant that we had to fix up the code from the proof of concept to match the new HTML, and then do it again when we needed to go back for more data later.

When the scraper reached an individual player's analysis page, there were 3 main sections of information. The first 2 are pictured below in Figure 4.

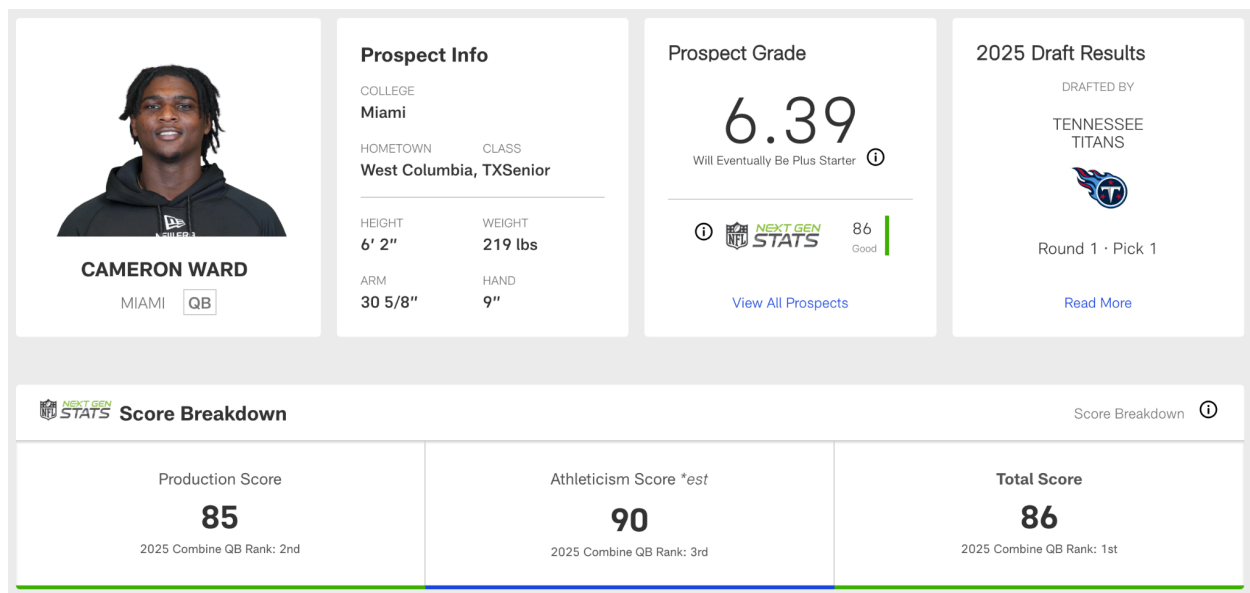


Figure 4: Cam Ward NFL.com page

In the top section, we are scraping the name, body measurements converted to decimals, prospect grade, and we scraped the draft position multiplied to exact pick number $((\text{round} - 1) * 32 + \text{pick})$ so that we could use it later to compare with our results. In the second section, the NFL Next Gen Stats section, we scraped the score and the rank for each section, all converted to integers. While the main concept behind this project is to use NLP, we recognized that all of this data is valuable to find a result, and we can run tests both with and without this data to determine the quality of the NLP features.

The third and final section of the NFL.com player page contains the pre-draft analysis paragraphs which are at the core of our concept. Each player has an 'Overview', 'Strengths', and 'Weaknesses' paragraph about them, written very densely with football terminology. Figure 5 below shows an example of what these sections look like.

Overview

Gunslinger with good size, a big arm and the mobility to help out his offensive line. Ward can read the full field and operates with average decision-making and processing quickness. Like a shortstop, he rips sidearm rockets that fit into tight windows on all three levels, but his delivery and mechanics cause inconsistencies with placement and accuracy. He is fairly consistent regardless of the coverage scheme he sees, but figuring out disguised coverage on the pro level will take time, and it is not a given he will develop that skill. He looks to strike it rich with aggressive, vertical throws; for better efficiency, he needs to learn to mine for gold with combo reads and rhythm throws. While he has the ability to move the sticks with his legs, he's more of a pocket passer than a dual-threat quarterback. Pocket mobility helps him extend and make plays out of structure, but the longer he's off-schedule, the spottier his decision-making can get. With a patient plan and a nurturing offensive coordinator who can accentuate his physical tools while regulating the feast-or-famine elements of his play, Ward could become a good NFL starter inside of his first contract.

Figure 5: Cam Ward overview paragraph

Once all of this data is collected, we stored it in a CSV for each position, with rows for each player and columns for each piece of collected data.

Prompt Engineering and LLM/NLP Utility

As mentioned, our goal was to utilize Natural Language Processing (NLP) to turn the analysis paragraphs like Figure 5 into measurable and meaningful data. There were three main difficulties that we came across during this task: scaling, football relevance, and output processing. Let's walk through each issue and our solutions.

Scaling

In order to break down the text into usable data, we needed to identify important phrases and then categorize them into groups which we predetermine (by position). Our vision was to have a feature for each category with a score, such that a player would have the features: {cat1: +3, cat2: -1}, etc.

The most accurate and consistent way to do this is by manually going through players by hand and categorizing the phrases ourselves, and then evaluating the text in front of us and scoring it. However, it was barely even reasonable to ask us to do this for one player, who may have 30-60 phrases to categorize, let alone thousands of players with tens of thousands of phrases. We used NLTK, which takes in human text and returns positive, negative, neutral, and compound scores between -1 and 1.

Football Relevance

The decisions to automate the NLP process brought forth a new challenge: phrase identification. Off-the-shelf NLP methods tended to work poorly for our task. The first method that we wanted to try was the ‘Bag of Words’ (BoW) method. For this method, we would need to build a lexicon of all words that exist in all of the text blurbs, find a way to filter these down to only meaningful words, and then count the number of times each word appears for each player and use the count as the feature for our models. While effective for general sentiment analysis with a small word set, we quickly understood that this would not be effective for our use case. Not only would our feature vectors be massive, but the context of the words would be missing. For example, we could take two phrases: “mediocre pass catcher” and “makes defenders look mediocre”. With BoW, these two versions of ‘mediocre’ couldn’t be distinguished. Finally, any words that didn’t exist in the training set would be disregarded completely by the model.

Naturally, the next method to try was the ‘nGrams’ method, which is the same as BoW but uses groups of consecutive words up to size n. This solves the problem of context and ambiguity, but it worsens the feature vector sizing concern, and once again, any new phrases (which are even more likely than new words) will be ignored by the model.

As mentioned, we wanted each player to end up with a score for each of our predetermined categories. These attempts at BoW and nGrams were detours, and were our first attempts at getting any sort of automated NLP. After recognizing their shortcomings, we went deeper into our tool bags to find something that would be able to match our desired output format.

We implemented a package called SpaCy, which monitors linguistic rules and tags parts-of-speech in a way that returns us meaningful phrases without being limited by the number of words. SpaCy would return us phrases that were more interpretable, such as “who lacks star-caliber power”, and would skip over any nGrams that did not have linguistic structure.

Then, the plan was to take the phrases for each player from each paragraph and input them into Google Gemini. The LLM here would serve as a human mind, being able to place each phrase into one of the categories we give it. The Gemini output would return a dictionary for each player for each paragraph, which holds a phrase as the key and lists the relevant categories as the value, with a (+) or (-) next to each to indicate a base interpretation of sentiment. For example, we might see {“has big hands but doesn’t know how to use them”: [“body (+)”, “catching (-)”]}. From this output, we would create a new dictionary for each player with a key for each category, and a concatenation of all phrases relevant to that category as the value. Then, we could send that text blurb into NLTK’s

polarity score generator and be returned a value representing the sentiment for that player surrounding that category.

However, this plan was short-lived, as NLTK failed for the same reason that BoW/nGrams didn't work: NLTK does not know football context. If NLTK reads the phrase "a monstrous wide receiver," it will give it a negative score because of the word 'monstrous,' even though this is a positive phrase in football context.

This complication led us to aggregate the (+) and (-) values for each category, since an LLM like Gemini has the unique ability to interpret this football-specific context. This method would output a positive or negative integer for each category, relating to the number of positive phrases (+1 per) and negative phrases (-1 per) identified.

Output Processing

It was at this point that we reached our final challenge with the NLP step in our process. Gemini was not consistent with its output formatting, leading us to tediously clean and fix errors that never should have been made. For example, sometimes, a (+) would not have parentheses around it, or a quotation mark would be placed on the wrong side of a comma, destroying the python dictionary format.

We did our best to fix the prompt to stabilize the formatting, but the issues prevailed, forcing us to think outside the box. Our final solution was to consolidate this entire processing step into the Gemini prompt. We eliminated SpaCy, and we eliminated the aggregation of (+) and (-) from the LLM output. Instead, we prompted our AI bot to split the phrases itself behind the scenes, categorize them and return us a dictionary with the final scores such that we didn't have to do any calculations ourselves.

A sample prompt is listed below:

""

You are to categorize a Python list of phrases I give you about a prospective NFL linebacker into the following categories. No problem if the phrase does not exactly match one of the keywords given; you are allowed some creative leeway in categorizing. If a phrase is not clearly attributable to one category, you may assign it two categories. When you find the right category, simply return it into a Python list format of the categories.

Here are the categories:

Play Diagnosis and Instincts: Pre-snap keys, post-snap reads, anticipation of run/pass and misdirection.

...

Competitive Motor and Effort: Hustle to the whistle, sideline-to-sideline urgency, finish mentality and consistent effort.

Instructions for output: Evaluate EVERY phrase in the blurb individually and categorize according to above definitions. Provide a SINGLE dictionary EXACTLY as formatted below, without any additional explanation or Python code around it. Use EXACTLY the category names listed above as keys. EVERY category must have a numeric score (0 if no phrases apply). Example of desired sample dictionary output:

```
{  
  "Play Diagnosis and Instincts": 3,  
  ...  
  "Competitive Motor and Effort": -1  
}
```

After you write the dictionary EXACTLY in the format listed above, please list your justification after for each phrase. Your justification should be listed after the close bracket of the dictionary.

Here is the blurb: [Insert Player Blurb Here]
"""

As mentioned, we engineered large prompts for Google Gemini's API to automatically classify our player blurbs with adequate football knowledge. This process required a Python loop with a backoff factor due to Gemini's API call constraints. As such, this was a very slow process with so many players. Additionally, we built three prompts for each position: a small, a medium, and a large, with 4, 9, and 20 categories respectively. This feature added flexibility in our pipeline, as we were unsure if few coarse-grained or many fine-grained variables would truly be predictive. We stored these prompts in properly labeled folders as txt files so they could easily be accessed by the script.

After receiving the prompt, the LLM would respond in the following way:

```
```python  
{
 "Play Diagnosis and Instincts": -2,
 ...
 "Competitive Motor and Effort": -1
}
```
```

****Justification:****

*****"Undersized outside linebacker lacking both starting experience and instincts for the position."**:**
This phrase negatively impacts "Play Diagnosis and Instincts". Lack of instincts is directly related to this category.

...

****"He has a natural feel for navigating traffic, but that won't be very valuable to him until he improves his play recognition."***: This phrase is a positive attribute related to "Pursuit Speed and Angle Play" (navigating traffic implies good pursuit and angles), but the caveat about needing to improve play recognition negatively affects "Play Diagnosis and Instincts".

"

This format of output was much easier for us to handle and convert from plain-text into a true python dictionary. We used the justification as an eye-test to ensure that the output was valid, and then eliminated it from the response to convert the rest into a usable format.

With that, our entire NLP process was now automated. After setting the POSITION variable in the pipeline and setting the number of variables to 'small', 'medium', or 'large', we could run the notebook all the way through the Gemini API call without touching the keyboard.

Finalized Dataset

After some basic cleaning and standardization of the features, we were ready to finalize the dataset for EDA and modeling.

We first split the data into X, Y, and INFO dataframes. We then set an index on the [player_id, player_name] columns such that we would always be able to connect data together to pull information. In Figure 6 below, we can see the columns that were stored in each of these tables:

| Table | X_df | Y_df | INFO_df |
|---------------------|--------------------------------------|--------------------------------------|--------------------------------------|
| Universal Info | Player Name | Player Name | Player Name |
| | Player ID | Player ID | Player ID |
| | Snaps Played Rookie Season NFL (PFF) | Snaps Played Rookie Season NFL (PFF) | Snaps Played Rookie Season NFL (PFF) |
| | Draft Position | Draft Position | Draft Position |
| Table-Specific Info | Height | PFF Grade | NFL.com URL |
| | Weight | | Draft Year |
| | NFL.com Grade | | Position |
| | Next Gen Production Score | | NFL Team |
| | Next Gen Production Ranking | | Analyst Name (who wrote paragraphs) |
| | Next Gen Athleticism Score | | Overview Paragraph |
| | Next Gen Athleticism Ranking | | Strengths Paragraph |
| | Next Gen Total Score | | Weaknesses Paragraph |
| | Next Gen Total Ranking | | SpaCy Phrases Overview |
| | 1 Column Per Category (4, 9 or 20) | | SpaCy Phrases Strengths |
| | | | SpaCy Phrases Weaknesses |
| | | | Gemini Output Overview |
| | | | Gemini Output Strengths |
| | | | Gemini Output Weaknesses |

Figure 6: X, Y, and INFO dataframe column labels

The only features that are input into the modeling processes are those in the X_df in blue coloring, i.e. the Table-Specific columns for X_df. It is also worthwhile here to mention that with some of our experiments, we ran models both with and without the top 9 features listed above, only using the

Gemini output categories. This allowed us to sometimes attempt to make the best model possible (using all features) and sometimes back off and simply evaluate the quality of the data we have generated (using only category scores).

Feature Exploration

After extracting features, we needed to make sure that these features would be suitable for modeling; after all, they are synthetic variables fully created by an AI chatbot. Following this idea, we modeled histograms for our language-based variables and were pleasantly surprised with their normal-like distributions. Attached are two charts with smooth, bell-shaped curves.

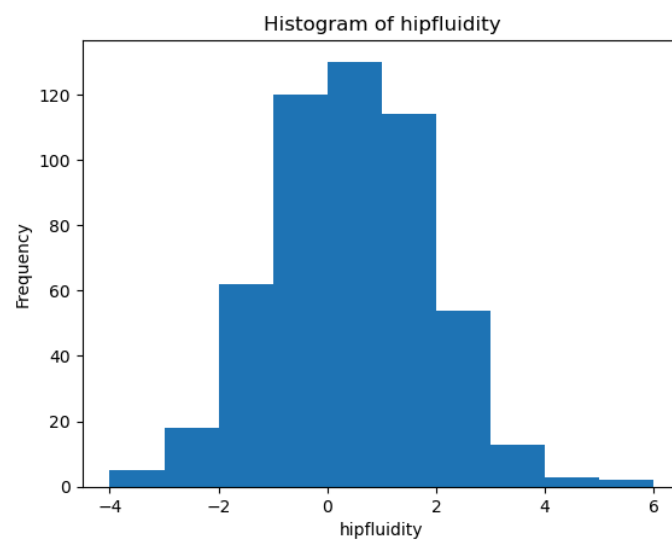


Figure 7: Histogram of Hip Fluidity (under our “cornerback” study). +1 is assigned to a player if they received a good grade, according to our criteria (explained below).

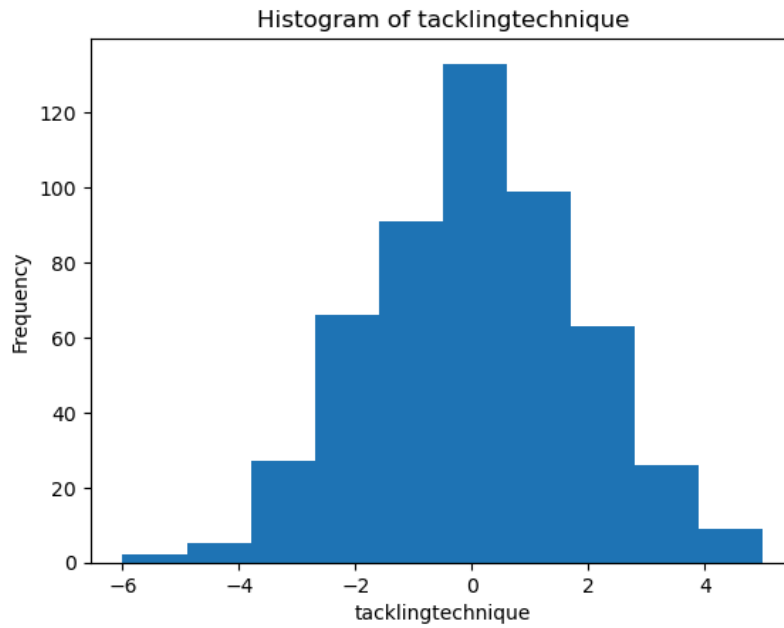


Figure 8: Histogram of Tackling Technique

After exploring variables at a 1-dimensional scale, we moved to see which variables are related to each other. We were pleased to find weak yet present relationships between features and some with the target. Attached are three plots of key 2-D scatterplots.

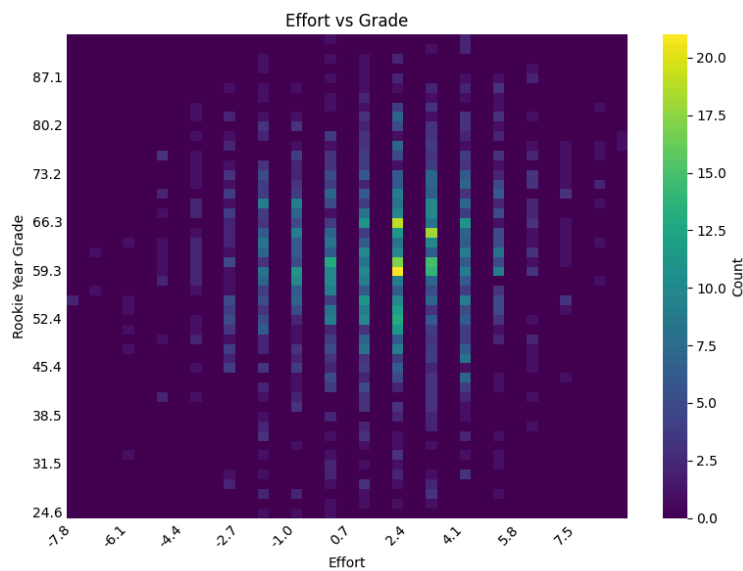


Figure 9: Effort vs PFF grade, our response variable. We observe a weak positive correlation.

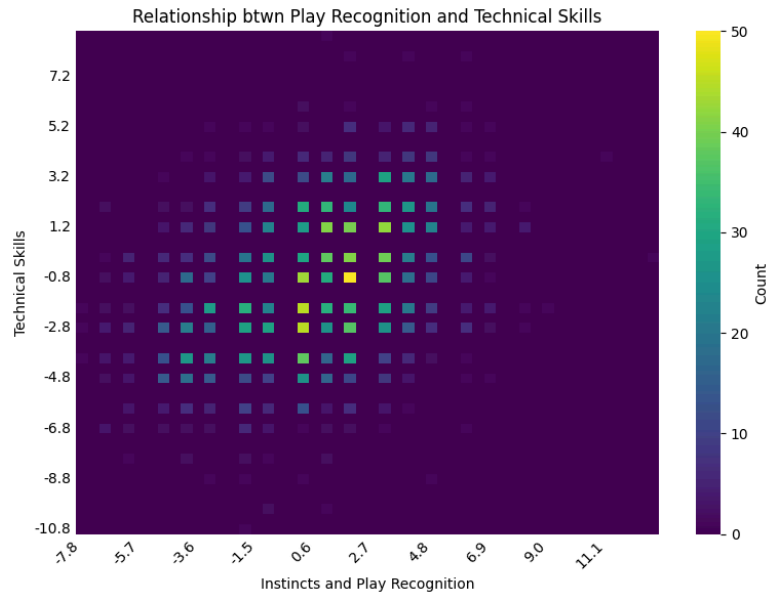


Figure 10: Play Recognition vs Technical Skills for all players. We observe a moderate positive correlation. We hypothesize that NFL players who have strong play recognition also have strong technical skills because both attributes relate to playing experience.

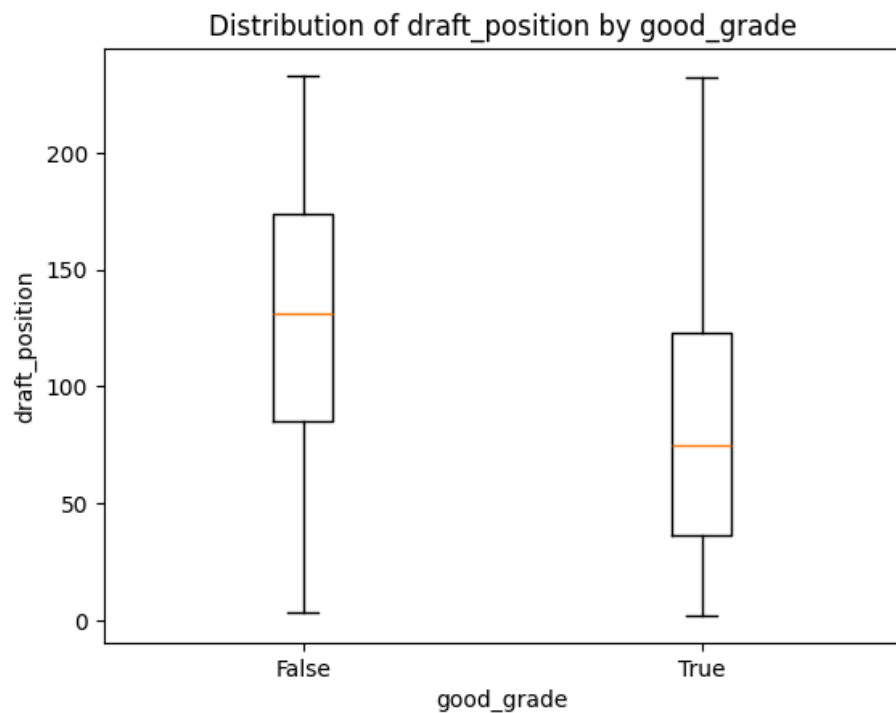


Figure 11: Distributions of draft position, conditioned on whether the player performed well in the NFL. Better players tend to be drafted earlier, which makes sense, but the relative width of each boxplot demonstrates how NFL general managers do not always pick the right players early in the draft. Great college players can perform poorly, and low-drafted players can perform well.

While there were no astounding correlations between our features, we are able to interpret these results to understand the quality of our data and gain some insights about our metrics.

Model Training and Evaluation

Defining Success

As we learned in our proof of concept, predicting exact player grades lack utility, as predictions often cluster around the mean. The added value of this project is predicting which players will succeed, not the grade of every player. As such, we moved forward, changing our modeling task to classification with an emphasis on precision.

Strictly viewing our project through a real-life lens, our model would likely be deployed in a scouting department in efforts to find great players to draft. As a result, our outcomes are relatively unaffected if our model misses a player who excels. However, if we identify a player we think will succeed, we advise our team to sign them, and if they fail, our team will have expended resources to get a player who did not contribute. This outcome directly reflects on our model; thus, we tuned our models to maximize *precision* (or to minimize the number of false positives). Even if our model predicts a low number of players to succeed, this outcome is a success if all of those players indeed perform well.

After fitting models, we observed AUROC curves and confusion matrices on our validation sets.

For most runs, we assigned our response variable—whether a player received a good rookie year grade— to “False” if a player played less than 100 snaps. This filter was applied to minimize the number of players who may have grades based on small sample sizes. Additionally, low snap counts usually correspond to teams not trusting their players enough to give them significant on-field time. This reason further motivated us to “punish” players who did not play a significant amount. A “good grade” was assigned based on whether a player scored above a 60 on PFF, which is above average.

Model Specifics

Logistic Regression and Random Forest models were run on the data. Random Forest was run to extract feature importance, and Logistic Regression was run for best results. Given that we created many datasets for each position, we extracted a wealth of results per-position.

Models were either split with a consistent random state or by year—trained on 2014-2021, val on 2022, test on 2023-2024. Most position-specific datasets had between 150-400 entries after cleaning; players were dropped if they did not have adequate pre draft data, which led to problems such as small sample sizes and test data. Our all-position dataset contained 2300 observations and as such suffered less from small data problems.

We used 5-fold cross-validation, either with a random state or by GroupKFold, grouping by year. We grouped by year because we split our data by year, so we wanted to keep our data as distinct as possible to maximize our ability to generalize to future years. This idea covers the case where a position's important skill set changes over time. We tuned C for Logistic Regression, and we tuned the number of trees and max depth for Random Forest models. After validating, we trained on the whole development set and tested once to see results after holding out data.

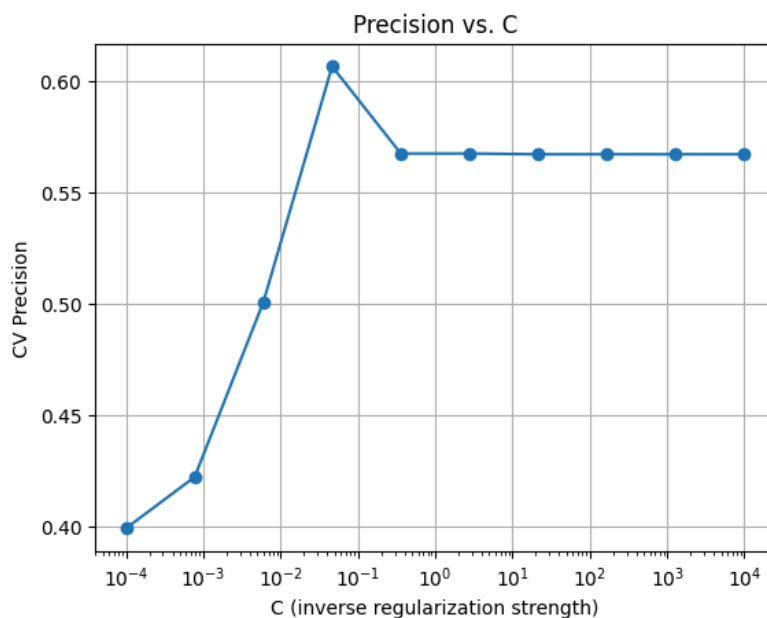


Figure 12: Precision vs C (regularization) for our heldout validation set. We see that $C=0.04$ is best.

Results and Discussion

As mentioned, we ran experiments on several positions individually, as well as on all positions together with more generalized categories. Additionally, we tested three different levels of categorization. While some of these tests certainly performed better than others, the results and takeaways are extremely promising, and lead us to believe that we can dig even deeper into the value of NLP in the future.

Before we dive deeper into these experiments, we note that there were some clear differences between model types. The 'large' category set size outperformed the 'small' category set size for every position by a wide margin, and outperformed the 'medium' as well, though by a lesser amount. We had discussed the possibility of overfitting by using too many categories and driving the model to analyze more data than we needed; however, it appeared that this was not the case. For a complex and noisy dataset like we had, the extra features help us refine the logistic regression curve.

Positional Summaries and Patterns

While we found great results with the Offensive Guard position, not every position we tried was as successful. As a reminder, due to a significant Gemini bottleneck, we only experimented with a small number of individual positions, so we could be missing some key results that would have been found with more time. That said, we observe below the Offensive Guards model with the 'large' list of categories and evaluate our key metrics.

Offensive Guard

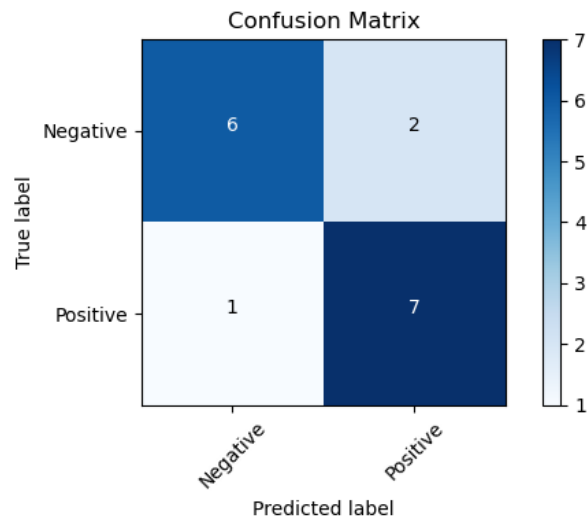


Figure 13: Confusion Matrix for GroupKFold offensive guard logistic regression on large category set

As shown in Figure 13, the model has maximized its precision at $7/9 \approx 0.78$. Note that there were only a total of 16 players in the test set (after data cleaning and filtering). Below, we can see the ROC graph for this model, which gives us an AUROC of 0.78, where 0.5 would be equivalent to random chance. Note that here, we used PFF = 55 as our cutoff instead of 60, as anything above 55 is right around NFL average, meaning the player is a decent enough draft pick with a league shortage of good offensive linemen.

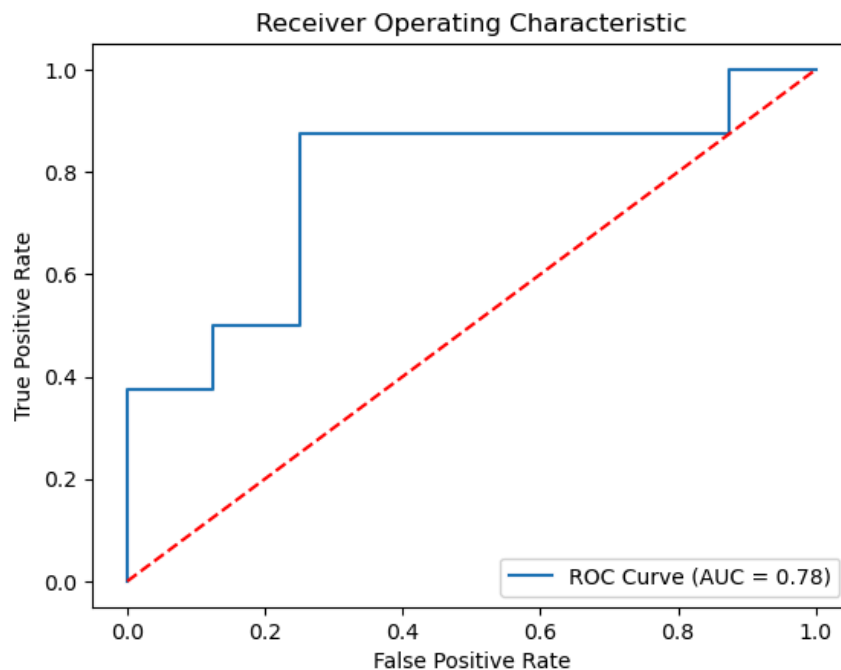


Figure 14: ROC curve for GroupKFold offensive guard logistic regression on large category set

In filtering, we eliminated all players who had not played 100 snaps their rookie season. The reason for this was to ensure that we weren't training on players who were injured their first year in the NFL. However, on the flip side, not playing 100 snaps could indicate low performance in practice, which would be important for our model. To evaluate these differences, we ran the offensive guard modeling script both with and without the filter, with 16 and 20 players respectively. The unfiltered version also gave an AUROC of 0.78, but had a slightly lower precision with 8 true positives and 3 false positives.

| y_test | y_pred | player_id | player_name | draft_position | y_test | y_pred | player_id | player_name | draft_position | snap_counts_block |
|--------|--------|-----------|------------------------|----------------|--------|--------|-----------|------------------------|----------------|-------------------|
| False | False | 77578.0 | Jordan McFadden | 149.0 | False | False | 77578.0 | Jordan McFadden | 149.0 | 162.0 |
| False | False | 99057.0 | Layden Robinson | 99.0 | False | False | 81785.0 | Nick Saldiveri | 97.0 | 18.0 |
| False | False | 99103.0 | Anthony Bradford | 102.0 | False | False | 99103.0 | Anthony Bradford | 102.0 | 659.0 |
| False | False | 100525.0 | Sataoa Laumea | 163.0 | False | False | 100525.0 | Sataoa Laumea | 163.0 | 355.0 |
| False | False | 129605.0 | Zak Zinter | 85.0 | False | False | 129605.0 | Zak Zinter | 85.0 | 231.0 |
| False | False | 131703.0 | TJ Bass | NaN | False | False | 131703.0 | TJ Bass | NaN | 344.0 |
| False | True | 77339.0 | Christian Haynes | 81.0 | False | True | 77339.0 | Christian Haynes | 81.0 | 167.0 |
| False | True | 102821.0 | OCyrus Torrence | 60.0 | False | True | 99057.0 | Layden Robinson | 99.0 | 602.0 |
| True | False | 146550.0 | Jackson Powers-Johnson | 44.0 | False | True | 102821.0 | OCyrus Torrence | 60.0 | 1307.0 |
| True | True | 59675.0 | Sidy Sow | 111.0 | True | False | 81933.0 | Jarrett Kingston | 199.0 | 1.0 |
| True | True | 77319.0 | Steve Avila | 37.0 | True | False | 101544.0 | Trevor Keegan | 165.0 | 35.0 |
| True | True | 78326.0 | Mason McCormick | 115.0 | True | False | 146550.0 | Jackson Powers-Johnson | 44.0 | 956.0 |
| True | True | 98261.0 | Christian Mahogany | 194.0 | True | True | 59675.0 | Sidy Sow | 111.0 | 772.0 |
| True | True | 158192.0 | Isaiah Adams | 71.0 | True | True | 77319.0 | Steve Avila | 37.0 | 1205.0 |
| True | True | 163815.0 | Dominick Puni | 87.0 | True | True | 78326.0 | Mason McCormick | 115.0 | 936.0 |
| True | True | 60818.0 | Andrew Vorhees | 204.0 | True | True | 98261.0 | Christian Mahogany | 194.0 | 144.0 |
| | | | | | True | True | 99084.0 | Nick Broeker | 205.0 | 3.0 |
| | | | | | True | True | 158192.0 | Isaiah Adams | 71.0 | 462.0 |
| | | | | | True | True | 163815.0 | Dominick Puni | 87.0 | 1078.0 |
| | | | | | True | True | 60818.0 | Andrew Vorhees | 204.0 | 268.0 |

Figure 15: Offensive guard player results with and without snap filter

First, evaluate the left screenshot in Figure 15. As discussed earlier, one of our primary goals was to be able to identify individual players as good who went late in the draft. This is the easiest way to provide value to NFL scouts and front offices. As we can see in the figure, we correctly identified both Christian Mahogany and Andrew Vorhees as good players, even though they were drafted 194 and 204. In fact, Christian Mahogany had the highest PFF rating in the entire dataset at 91.5 which means that this would have been a magical discovery for any NFL team pre-draft. On the other hand, with a cutoff at 55, we predicted that Christian Haynes and OCyrus Torrence would be good, while their true values were 48.5 and 54.9. Torrence can hardly count as a loss for us, as he was 0.1 away from our prediction being correct, but Haynes is unfortunately a bad miss that could have cost a team. That said, we are still very proud of these results.

Our discussion gets a bit more interesting, however, as we move to the right image in the figure, the dataset unfiltered by snap count. First of all, we add Nick Broeker to our list of late round success stories, although he only played three snaps with an aggregate PFF score of 57.5. We also added a false positive in Layden Robinson, who we had correctly predicted in the filtered version. His true PFF score was 43.6, which is an even bigger disappointment than Haynes.

We were a bit surprised to find that OCyrus Torrence actually led the entire test set in snap count at 1307 snaps played his rookie season. However, his PFF score was only a 54.9. Compare this to Mahogany, who only played 144 snaps but led all offensive guards in our entire dataset with a PFF score of 91.5. This begs the question: is PFF score actually the best metric to determine player quality? Does an increased snap count imply that a player is more impactful? Maybe the coach's eye test isn't perfect? Maybe a team has a shorter depth chart? Given the time and money that goes into generating PFF scores, and the general consideration that they are industry standard and high quality, it is safer to assume that they are in fact a good metric. However, we still have Nick Broeker, who received a 57.5 for the year on only three recorded snaps. Do we trust the PFF numbers or the coach's choices here? There is certainly an argument for both.

If we look a bit deeper into some of these players, the Lions have the top ranked offensive line in the NFL. Maybe Christian Mahogany only played 144 snaps because the players ahead of him are so successful already. On the other hand, OCyrus Torrence starts for the 8th ranked offensive line of the Bills, even with a statistically mediocre year. It could be the coach's preference, or maybe a lack of a better alternative, but it is much easier to trust that PFF scores are generated on a play-by-play basis than it is to throw our analysis out the window because the snaps played don't perfectly correlate.

Ultimately, it appears that our filtered version of the experiment gave slightly better results, but we may not have enough data yet to fully confirm that conclusion.

Defensive Back

While less impressive than our offensive guard set, we got great results with defensive backs, and with a lot more data in the test set. Here, we had 122 players in the test set, which possibly adds to the validity of our results.

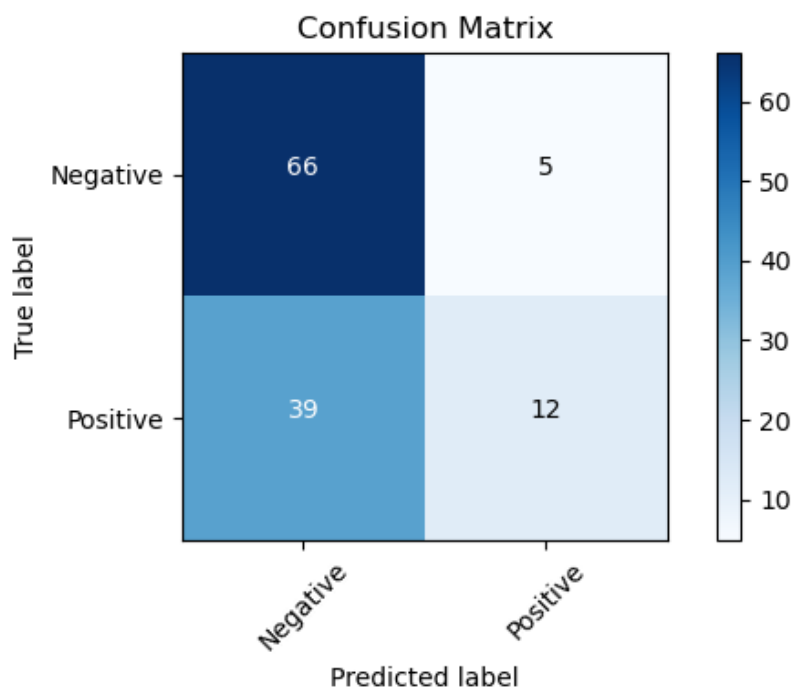


Figure 16: Confusion Matrix for GroupKFold defensive back logistic regression on large category set

As we can see in Figure 16, we were able to get a precision of $12/17 \approx 0.71$. Additionally, as seen below in Figure 17, we achieved an AUROC score of 0.64.

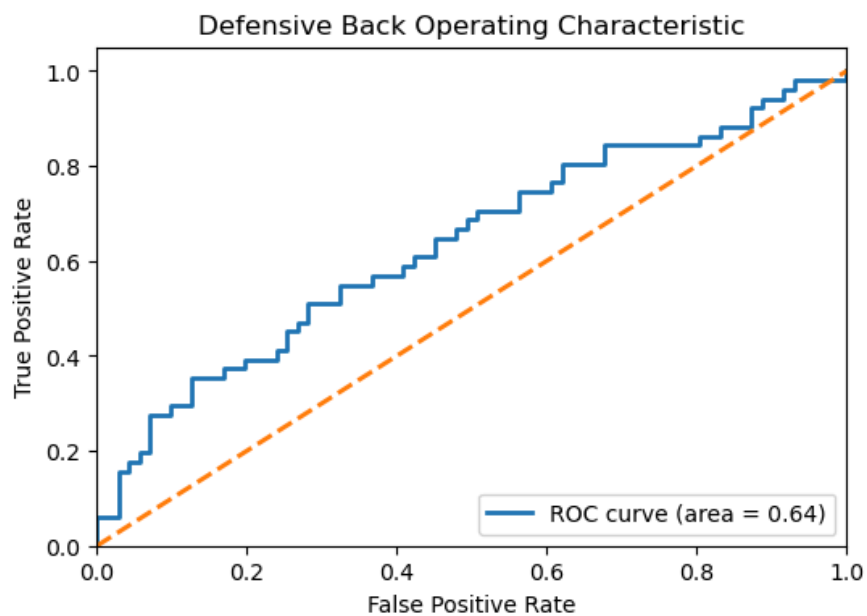


Figure 17: ROC curve for GroupKFold defensive back logistic regression on large category set

While the numbers are not quite as high as for offensive guards, we have more data to back it up, and fine tuned our cutoff point to achieve the best results, which in this case, was 64.

Running Back

Running backs are an example of a less successful experiment. Below is the confusion matrix for our best model:

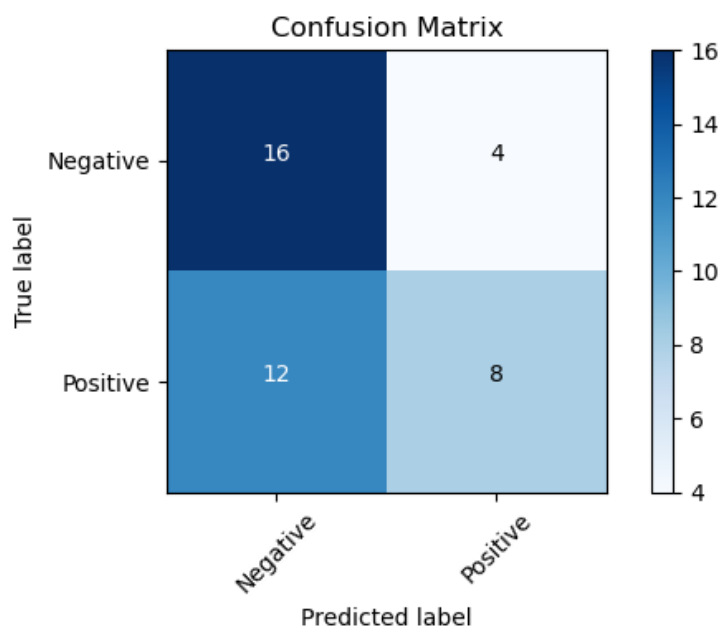


Figure 18: Confusion Matrix for GroupKFold running back logistic regression on large category set

As we can see, we have 40 players in our test set. Running backs' score distribution was shifted a bit higher than many other positions, with a median around 65, so our refined model used 66 as its cutoff, which gave us the best results. Our precision is $8/12 = 0.667$, which isn't bad, but is not as good as some of our other positions and models. Additionally, the AUROC score for this model peaked at around 0.56, which is only moderately better than random choice.

Unfortunately, we weren't able to find the same success in all of our positions. That said, we believe there are still important insights that can be used from all samples in our all-position models.

Feature Summaries and Success Traits

Once again, our most successful experiment was our Offensive Guard model, which recorded a precision of 0.78 and an AUROC of 0.78. Since we used a logistic regression model, we were able to plot the weights which the model assigned to each feature. Observe these results below in Figure 19:

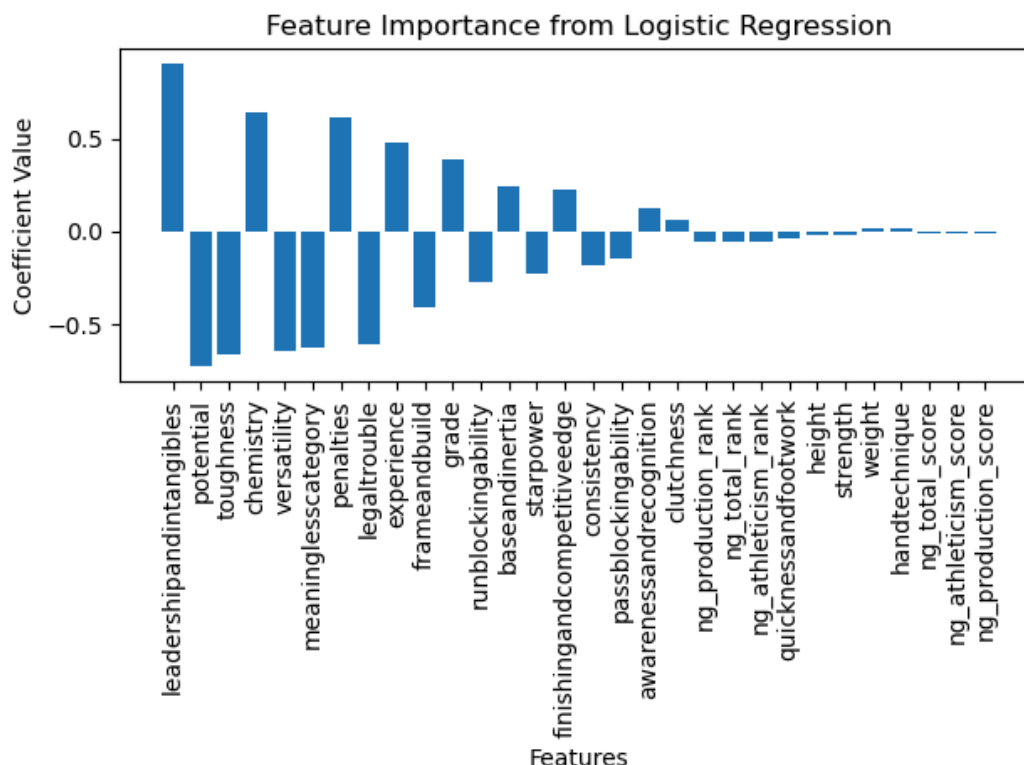


Figure 19: Feature weights for offensive guard GroupKFold logistic regression on large category set

One of our key metrics was to determine which features held the highest weights in a successful model. Since our features are categories representing real characteristics about a player, these feature weights can give us insight into which players have a better chance of being successful in the NFL, regardless of our exact model results. This is a good time to give a reminder that our dataset is not large enough for this chart to be conclusive, but it is certainly interesting to observe. Let's get into it.

Leadership and Intangibles is far and away the most important feature in this model's predictions. This may seem a bit shocking given that run blocking, pass blocking, and other performance based categories exist in this dataset. There are several reasons why we could be seeing the results that we are. The first is that maybe intangible characteristics are actually the most important for offensive guards. This case is further strengthened by the fact that the second largest positive coefficient is for the Chemistry category.

The main alternative reasoning as to why we are seeing these results (apart from overfitting or fortune), and probably the most likely, is that analysts' language surrounding different categories have different levels of conclusivity, or in other words, differing levels of ambiguity of interpretation. It is possible that the graph we are looking at only tells us that analysts are rarely wrong about leadership skills, but can often be wrong about other categories. It is extremely important to understand that our categories come from an interpretation of a single analyst's words as opposed

to ground truth characteristics. Chances are, our results would be a lot stronger if we could get a flawlessly prepared ‘true’ score for each player for each category instead of needing to infer it from an analyst report.

That said, the model left some results to be desired. We found ourselves unsurprised that Potential had a negative coefficient. It is often true in sports that those who are touted to have a high ceiling do not ever reach it, especially their rookie year. This probably indicates that even if they have lower potential, ‘NFL-ready’ players tend to be more successful in their first season in the NFL as opposed to ‘projects’ or players that need many years to improve.

There are several other interesting results here. Toughness and Versatility both have negative coefficients. The Next Gen statistics meant almost nothing. Both Run Blocking and Pass Blocking have negative coefficients. Frame and Build has a negative coefficient. We don’t have the answers to these, but we do understand that our model isn’t perfect, and once again, a good proportion of these results can be attributed to the discrepancy between our inferred category scores based on the writing of these analysts and the true characteristics of these players.

Finally, while the results are not quite as good, running this model using only the LLM category data still gives us an AUROC of 0.7, with a $5/8 = 0.625$ precision. This just about confirms to us that even though the model is not perfect, there exists value in the features which we have generated. For future studies of offensive guards, we hope to increase the dataset size to gain confidence in our predictions.

All-Position Summary and Draft Comparison

We ran the all-position models both using all of our features, as well as only the language data as discussed earlier. While our categories were more general in order to encompass traits from all positions, we were still able to find value in our results.

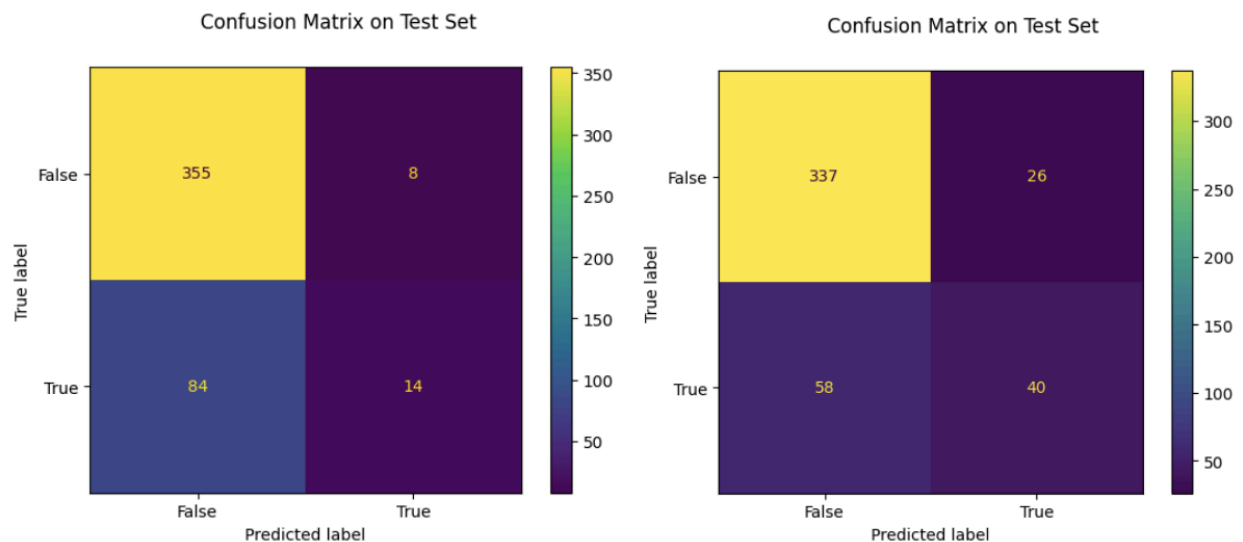


Figure 20: Confusion matrix for all positions combined, only language on left, all features on right

As we can see in Figure 20, we ran our models to maximize precision. It did not matter to us how many players were predicted to be poor as long as we had a high success rate for our positive predictions. On the left in the figure, we observe the experiment using only the LLM category features, and on the right, we observe the experiment using all of our features. Interestingly, our category-only feature set actually was able to peak at a higher precision, $14/22 = 0.636$ versus $40/66 = 0.606$. Also note that the AUROC scores for both were around 0.65-0.68.

While potentially promising, we must keep in mind that these precision values are not good enough to be put into production. The key takeaway here is that there is value hidden in our generated numbers, and that they are quality predictors, even when used independent from all other data. We know that we can improve our model and increase that precision over time.

Take a look at this set of graphics depicted below in Figure 21.

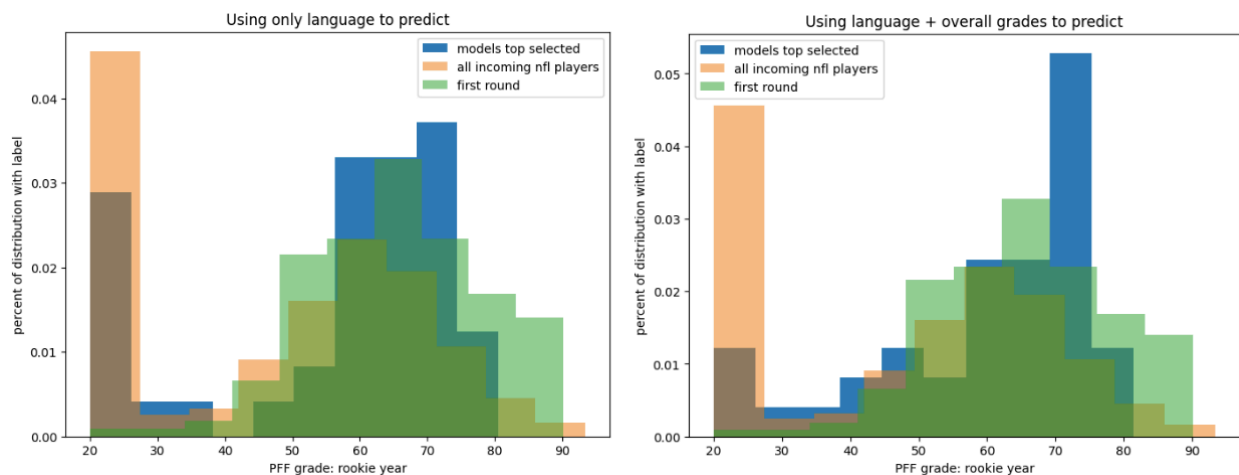


Figure 21: Histograms for all-position models; only language on left, all features on right. The y-axis is the percent of players who received the labeled grade on the X-axis in each population.

Once again, we see the model run on only the LLM category features on the left, and the model run on all features on the right. The x-axis contains 10-point buckets for true PFF scores, and the y-axis represents the density in each bucket. Think of these as a continuous PDF that is then bucketed into groups. Note that the spike at 20 is due to using 20 as a placeholder for players who did not play enough to get a PFF score. In the orange, we have the distribution of all incoming NFL players since 2014 (as limited by our data scraping). In the green, we see the distribution of all first round picks in the same time frame. In the blue, we see the distribution of our model's top 40 selected players.

Notice that in both histograms, our model's top 40 identifies a higher concentration of players in the 55-75 range than either of the other two distributions. The first round of the draft and the total

aggregation of incoming players both contain a higher density above and below this 20-point region. However, our model appears to be more consistent in the average to above-average range, particularly in the model using all features, where we see a remarkable spike around 70-75.

From a practical standpoint, if an NFL front office were to consult our model's top 40 player recommendations during the later rounds of the draft with the goal of selecting players likely to be productive, they would find tangible value. According to the histogram on the right, any player remaining on the board from our top 40 has approximately a 65–70% likelihood of achieving a Pro Football Focus (PFF) grade of 57 or higher in their rookie season. Importantly, our model does not exclusively identify early-round prospects; a significant portion of the top-ranked players fall outside the first round. This supports the idea that these players are realistically available in the later rounds, thereby offering decision-makers multiple viable options. This represents immediate, actionable value, and we are confident that further refinements to the model and the inclusion of additional data can improve these results even more.

Limitations and Concerns

The primary challenge in developing our position-specific models was the limited availability of data. After applying necessary filtration and cleaning procedures, some test sets were reduced to as few as 10 players, representing only two of the eleven years in our dataset. While we are highly encouraged by certain outcomes, we acknowledge that our results do not establish causal relationships between the input features and performance outcomes. Furthermore, as previously noted, there remains a valid concern that the data derived from analyst reports may not perfectly reflect the actual attributes of the players. Nonetheless, given the constraints of time and data availability, we were able to produce results that are both promising and worthy of further investigation.

From the start of this project, we had always assumed that with data this complex, a well-refined neural network would give us the best results. In truth, we still believe this, but we are limited in our capabilities. We attempted to train a multi-layer perceptron (MLP) on our most successful dataset (offensive guard) to no avail. Our best AUROC was a 0.73, which isn't bad by any means, but doesn't compete with the 0.78 we got from a simpler logistic regression.

At the bottom of this shortcoming is a shortage of data. We do not have enough offensive guard data to properly train an MLP classifier, and on top of this, it may not be necessary for a single position. That said, with clear-cut differences between positions, it is impossible for a logistic regression model to perfectly fit our all-position dataset.

The underlying rationale is that for certain positions, specific combinations of feature categories are more predictive of success than others. When training a model across all positions simultaneously, a single uniform set of predictive features is unlikely to emerge. This necessitates the use of a more sophisticated model architecture—such as a multilayer perceptron (MLP)—capable of capturing nuanced, nonlinear interactions between features. One might suggest incorporating position as an

input feature to address this variability. However, doing so effectively returns us to training distinct models per position, which is not feasible given our limited data. While it is true that different positions emphasize different attributes, there is sufficient overlap in successful player characteristics across positions for an MLP to learn these patterns without requiring explicit positional information.

Unfortunately, we ran out of time to properly train and refine a neural network on our all-positions dataset. However, we will look to complete this in the future and continue pushing to better our results and create a valuable product.

Challenges and Lessons Learned

Process

How can we make it more efficient

NFL.com frequently changes their HTML element names, making data scraping tedious. To be more efficient next time, we could skip scraping altogether and find a way to collect all necessary analysis and statistics directly from NFL.com and PFF.

Running our code through the API was also slow due to Gemini bottlenecks. If an error appeared, we had to wait about 30 minutes to re-run the code and see if the fix worked. Since we were using the free version of the API, we were limited to one dataset run per day. This meant a single error could cost an entire day. To improve efficiency, we should add more print statements to verify outputs before using the API, helping us avoid delays caused by avoidable errors.

How can we incorporate more analysis

Given the amount of data we had, we did a relatively good job analyzing it. We produced a large number of graphs that revealed promising and interesting insights. In the NFL, some positions require similar traits, while others demand very different ones. It could have been useful to analyze traits across positions to see which attributes matter more when drafting for one position versus another. While we interpreted many of our graphs ourselves, we could have added more raw data analysis by using our database to calculate additional statistics not already present in the dataset.

Product

How can we improve the quality of the product in the future

In the future, we can make the project more accessible by better documenting how results are generated. Since we're familiar with the system, the way points are added or subtracted from a player's traits is intuitive to us, but it may not be as clear to someone new to the work. Providing a clearer output—particularly for the EDA analysis—would make the findings easier to follow. Adding more checkpoints and being more detailed in the data cleaning process would also improve readability and help the models run more efficiently on the collected data.

Challenges

Data Processing

A major challenge with using the data collected to determine player success is that the data collected in scouting reports, analysis, and statistics does not tell the players the entire story. Talent alone does not determine how a player performs. Team fit, coach fit, and surrounding talent are key factors in how a player performs on the field. If a player blends into a team better or has better teammates their stats on the page could be different than if they did not mesh well with their teammates and/or had teammates that did not execute to the same level as them. This is more true in some positions than others. For example, a quarterback's stats will be greatly affected by how well his teammates perform, but kickers on the other hand may not. While this is clear in stats, it is also true to scouting reports since analysts can not see a player's true talent one way or another if their talents do not fit into their team properly. Any given player can perform poorly on one team, but well on another. Given all of this and that this is a factor with some positions more than others, football data can be extremely noisy.

Modeling

Football data is not only extremely noisy, but also difficult to work with in a machine learning context—especially in the NFL. Scouting reports rely on language, while statistical analysis relies on numbers, and merging the two into a format that works well in a machine learning model is a major challenge. On top of that, written analysis and raw stats often don't capture the full picture of a player, and outliers in football can easily skew the data. Because of this, predictive models like ours are inherently prone to error and can struggle to accurately classify a player as a success, average, or bust.

Conclusions and Future Work

Future Studies

In the future, this model can be used as the basis for several follow-up studies. Its core goal is to predict whether incoming NFL players will become successes, average performers, or busts. One valuable experiment would be to run the model on a new draft class, compare its predictions to actual draft positions, and sort players into four groups: drafted high and a success, drafted high and a bust, drafted low and a success, and drafted low and a bust. While identifying average players is useful, the real value of the model lies in helping teams spot under-the-radar talent and avoid overhyped prospects—while also ensuring they don't pass up legitimate top-tier players or reach unnecessarily. Running this test would help assess how well the model performs this task.

Another potential direction is comparing the model's outputs with real-world team behavior—both in terms of overall performance and drafting success. By benchmarking our model against teams with strong or weak drafting histories, we can identify which teams might already be using effective strategies and which might benefit from adjustments. This could allow the model to serve as a guide for improving draft decision-making.

Finally, the model could be used to identify the most important traits for different positions. If certain traits consistently correlate with better outcomes for specific roles, teams could use this information to make more informed decisions—cross-referencing scouting reports, player stats, and college performance to better assess overall fit.

References

Dzwil, Matthew, et al. "Predicting Rookie Season Performance Based on National Football League (NFL) Scouting Combine Movement Analysis." *A Major Qualifying Project Proposal Submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE In Partial Fulfillment of the Requirements for the Degree of Bachelor of Science*, 27 Apr. 2023.

King , Connor. "Success or Bust? An Analysis of Draft Position and NFL Success." *Fisher Digital Publications* , 2013, fisherpub.sjf.edu/cgi/viewcontent.cgi?article=1064&context=sport_undergrad.

NFL. (2025). *2025 NFL Draft: Quarterback prospects tracker*. NFL.com.

<https://www.nfl.com/draft/tracker/prospects/qb/all-colleges/all-statuses/2025?page=1>

<https://www.pff.com/>