SIADS 696: Milestone II
College Football to NFL Draft Prediction

Andreea Serban (aserban@umich.edu), Chris McAllister (chrismca@umich.edu), Ryan Obear (obearyan@umich.edu)

Introduction

As University of Michigan students and enthusiastic football fans, following the University of Michigan's football team's national championship victory in 2023 we were inspired to explore what truly determines a player's path to the National Football League (NFL) starting from high school.

The journey to the NFL is the dream for many high school football players. However, the NCAA reports that only 7.5% of high school football players end up playing for a college [1]. It already takes a lot of time, effort, and hard work for these athletes to get to play in college. And choosing the right college for each player can make or break a player's career.

Our project aims to leverage data on high school players who have already committed to colleges, in order to cluster the data points and build a predictive model for whether a player will make it to the NFL.

The impact of this model comes in two parts: 1) It can help recruits understand the impact of choosing their school can impact their draft likelihood and 2) it can help colleges identify players with high potential that other schools are missing out on.

We incorporated two unsupervised models: 1) a simple K-Means model on all numerical features and 2) a model leveraging PCA first and then DBSCAN over the 2-dimensions created by PCA. We added the clusters from these algorithms to our dataset so they could be used as features going forward. The PCA plus DBSCAN algorithm had a slightly better silhouette score of 0.25.

With this dataset, we created three types of models: Logistic regression, SVM, and Random Forest. Our best model, the Random Forest Model, had a f1-score micro of 0.8602. Some of our main findings are that:

- Recruiter's rating is a highly predictive feature of draft likelihood.
- The closer the player was to the school they attended the more likely they were to get drafted, suggesting that good football colleges yield good football high schools (or vice versa).
- College teams that have a high rolling 2 year win percentage and point differential tend to recruit the students that are most likely to get drafted (or vice versa).

Related Work

This research builds upon our Milestone 1 project [2], where we explored to see what factors about a high school student lead to them getting drafted to the NFL. Additionally, there were two similar projects done by students:

1) One project was done by a student named Nicholas Wheeler at Claremont College entitled Do High School Football Recruit Ratings Accurately Predict NFL Success? [3] This project performed a linear

regression to see what attributes of a high school player's profile impacted their NFL success. We took a step further than this project by expanding on the feature set to include player's hometown coordinates and information about their college of choice and the quality of the school they commit to.

2) The other similar project was done by a student named Henry Gorelick at Fordham University titled: NFL Draft Analysis [4]. This student made a Random Forest model to predict a college student's likelihood to get drafted by the NFL. Though the scope is slightly different since it leveraged data from students that have already performed in college, the approach was the same.

Data Source, Preprocessing, and Feature Engineering

Data Source

Our data was sourced from two places: (1) an API located at https://collegefootballdata.com/ [5] and (2) data scraped directly from https://www.pro-football-reference.com/ [6]. For an overview of how to access the API, check out the first section of this notebook: 1_pull_data_M2.ipynb . If you try to pull the data yourself, be sure to swap in your own API key, which can be accessed here.

The majority of our data came from the college football API [5]. It included valuable information like the player's recruiting rating, their height / weight, what position they played, what school they committed to, and much more. Ultimately we pulled in 44k records from this API; one row for every highly rated high school recruit from 2010 - 2019. We'll deep dive which features were most important in the Supervised Learning portion of the report.

The only thing we pulled from pro football reference was our target column, a flag saying whether that player was ultimately drafted. To see the logic for pulling this data, see the second section of this notebook. This is the same data we used for our Milestone 1 project [2]. Both datasets were formatted as CSVs.

Data Manipulation and Preprocessing

Joining the recruiting data and the draft data was a tremendous challenge since the only identifier we had was the player's name, and players often have the same name. For thorough documentation on how this join works, please see slide 5 of our milestone 1 project [2].

Another data manipulation challenge that was unique to milestone 2 was creating what we call "team momentum features." These features looked back over a two year window leading up to the player's commitment to get a sense of how successful the program was at the time of commitment. This included features like win percentage, playoff victories, and point differential - all of which ended up being valuable features for prediction.

Fortunately, not many of the records from the college football API were null. In fact, no columns had missing data in more than 2% of records. See below for a summary of how many records were missing, and the imputation logic we leveraged to fill them:

- **Recruiting Ranking (.16% missing)**: Missing rankings were imputed using the 'rating' column
- **Height / Weight (0.1% missing):** Imputed with the average height / weight of a player in their position.

- **Hometown Latitude / Longitude (0.7% missing):** Imputed with the mean latitude / longitude of their home state.
- **Name of Hometown Country (0.2% missing):** Missing values are filled with USA.
- **Missing "Team Momentum Features" (1.5% missing):** These were often missing if a school was recently promoted from FCS to FBS college football (think of it like going from AAA to MLB in baseball). In these instances, we replaced their momentum features with the mean value of the conference that they now compete in.

Feature Engineering

Given the list of variables we got from the dataset, we calculated some additional features that we believed would be insightful to our clusters and models.

For numerical features, in addition to the "program momentum features" described in the Data Manipulation section above, we calculated the distance metric (distance_miles) which took the distance between the hometown coordinates and the college the player committed to.

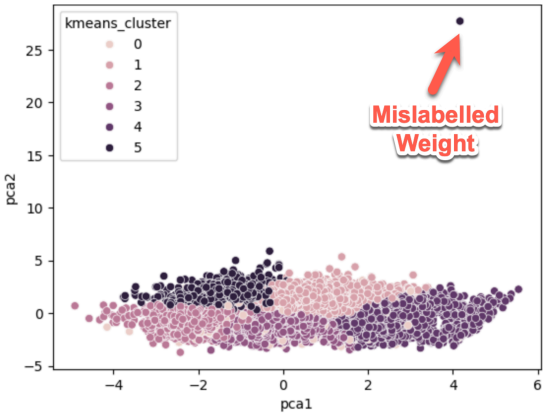
For some categorical features we created groupings of features based off more granular categorical fields:

- **Side_of_ball:** Whether they play offense, defense, or special teams.
- **Position_group:** Converted things like Guard, Tackle, and Center to "offensive_line"
- **Conference_group:** Flag saying whether the school they committed to is in one of the 5 major conferences.

Anomaly Detection

Any anomalies caused by a data input issue can result in poor performance from clusters and classification models. So we leveraged K-means clustering on the numerical features to help us find any anomalies within the dataset. After finding the optimal amount of clusters (six clusters), we transformed the features with Principal Component Analysis (PCA) to visualize the cluster.

Figure 1: K-means clustering with PCA

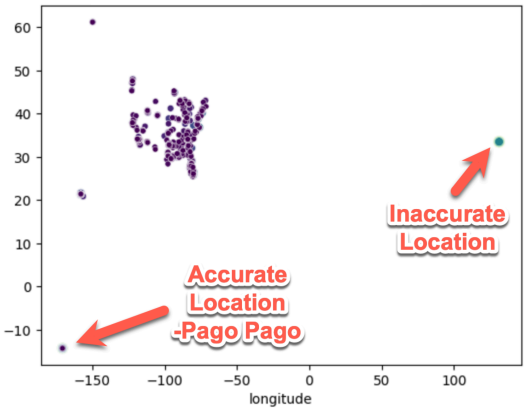


As we can see in Figure 1, there was a noticeable anomaly datapoint that turned out to be an inaccurate weight (2,000 lbs) for a player. After looking up this player, we found his true weight to be around 200 lbs - not 2,000 lbs. So, we can only assume that there was some sort of data entry issue.

After visualizing that clear anomaly, we expanded our search to see any addition anomalies in the data by measuring the distance between each point and their cluster's centroid, and flagging any data points that were more than three standard deviations away from the centroid compared to the other points in the cluster. After doing so, we found 574 players that were flagged as anomalies. Out of these flagged anomalies, we discovered that:

- 565 anomalies were associated with players who had hometowns on an island called Pago Pago. This island is located in the distant South Pacific in the American Samoa. As shown in Figure 2, the coordinates are drastically different from the rest of the dataset.
    - We were unaware that there were players from this remote island. After doing some research, we found that this is a strong football presence with this very distant island [7]
- 7 players had inaccurate hometown coordinates and located somewhere in China - which was incorrect once we looked up each of these players true location
- 2 players had inaccurate weights. One player was the one referenced before with a recorded weight of 2,000 lbs, and another player with a recorded weight of 27lbs.

Figure 2: Football Player's Hometown Coordinates



Even though we only found a total of nine anomalies, we were able to get rid of some noise with the datasets and most gained a powerful insight about our dataset.

Final Dataset

After these data manipulation and feature engineering steps, we've finalized the dataset that will be used within the classification supervised learning model.
To see the full list of features and their descriptions, please reference Appendix B.

Unsupervised Learning

After we cleaned up the dataset, we identified that our primary use case for these unsupervised models is to use these clusters as input within our classification model. Since we would like to use this within our classification model, there were two things we had to keep in mind:

1. Prevent any data leakage.
2. Select clustering approaches that can provide predictive value.

To prevent data leakage, we were mindful in separating out our dataset into training and testing datasets. We did this by applying a random 75/25 train test split. With the intention of using the train dataset to fit our clusters and then predict the clusters with our testing dataset. Note that not all dimensionality reduction techniques allow for this flow; t-SNE and MDS are only meant for visualization - not for "learning a transform and then reusing it on different data (ie a Train/Test)." [8]

With these goals in mind, we decided to look into these modeling approaches to create clusters with predictive value :

1. **K-Means:** simplest clustering method
2. **DBSCAN with PCA:** best at identifying clusters of any shape and feature of assigning noisy points to their own cluster.

Evaluation Metric

We considered using the Calinski-Harabasz score (CHS), but ultimately we decided on the Silhouette score. We made this decision primarily because silhouette score gives us an objective way to measure the separation and quality of our clusters to find well defined clusters, but also because CHS can struggle with non spherical clusters. Silhouette score gives us an easy to interpret value of -1 to 1 where higher values indicate that points are well clusters and distinct, and negative values indicate points are likely assigned to the wrong cluster.

We've also added two more domain-specific metrics, to help us understand its effectiveness to separate out players that are most likely to get drafted vs. ones that are not. These two additional metrics are:

1. **Capturing Players with High NFL Likelihood -** The highest draft percentage among the clusters
2. **Capturing Players with Low NFL Likelihood -** The lowest draft percentage among the cluster

Now that we've established our use case, created our training dataset, and aligned on our evaluation metrics, we went ahead and created our clusters.

Simple k-means

With the k-means approach, we just leveraged numerical values and apply the standard scalar so that each of the values are equally comparable across the features. We determined the optimal number of clusters by using a combination of a silhouette analysis and elbow method. We found that four clusters were most optimal with a silhouette score of 0.24.

After creating the clusters, we analyzed the clusters by transforming the features with PCA and plotted it in a scatterplot. As shown in Figure 3, there seems to be some overlap or maybe adding another dimension could make these clusters look a bit more separated.

To help us understand these clusters more, we took the already standard scaled features, calculated the mean within each of the clusters and plotted it on a heatmap with each cluster next to one another to produce Figure 4. We also calculated the draft percentage of each cluster. With these two visuals, we were able to develop some personas for each of the clusters. These were the personas we created:

Cluster 0: "The Underdogs" – Smaller, lower-rated players who make up the largest group but have the lowest draft rate.

Cluster 1: "The Giants" -Taller Players" (Higher weight and taller height) + lower ratings/stars

Figure 3: k-means clustering with PCA

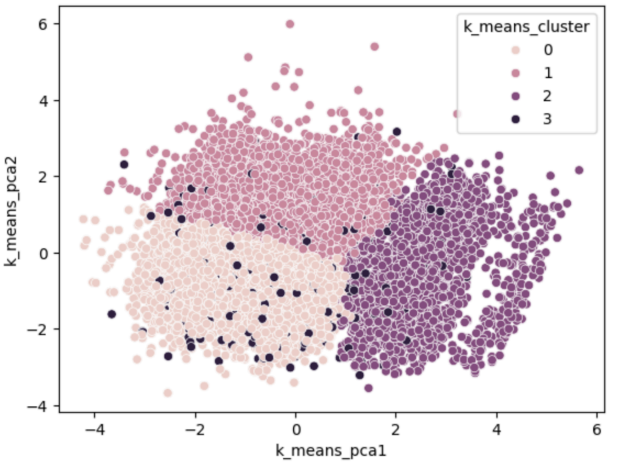
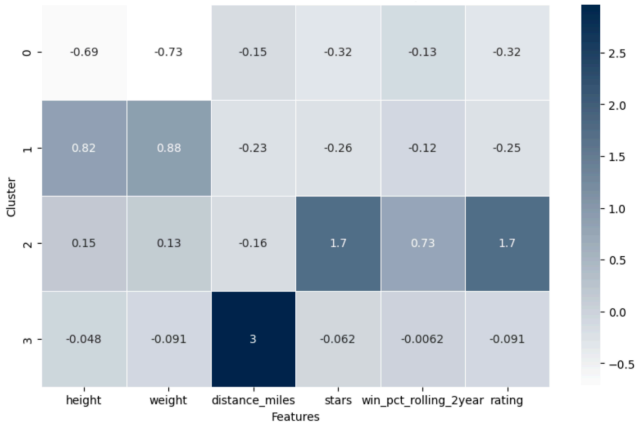


Figure 4: Normalized Mean Values per K-means Cluster



Cluster 2: “The Rising Stars” - Average Height/Weight + Higher college team win percentage + Higher ratings/stars
Cluster 3: “The Traveling Warriors” - Travel the furthest + Average on height, weight, stars, win percentage + rating

Overall, this approach was the best at bubbling up the players that got drafted without using that as a feature.

DBSCAN with PCA

The most critical parameters for DBSCAN are epsilon, and min_samples. Epsilon, named eps, is the maximum distance between samples for one to be considered in the neighborhood of the other. If Epsilon is too high, dissimilar points will be in a single cluster, however if it is too low, similar points will be treated as noise. Min samples, is the number of samples in a neighborhood for a point to be considered as a core point. If min_samples is too high, clusters may be treated as noise, however if it is too low, good clusters may be broken into smaller clusters that don’t capture the underlying structure of the data.

Figure 5: PCA Draft percentage

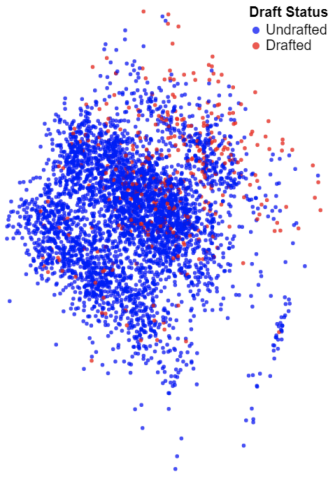
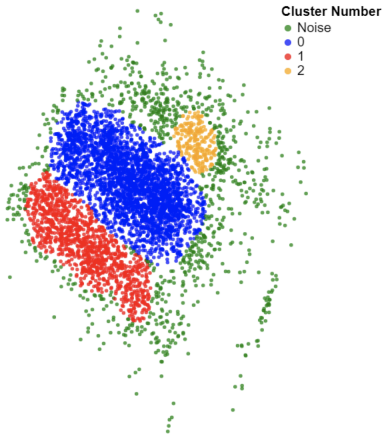


Figure 6: DBSCAN with PCA Clusters



To tune these hyperparameters, we performed a gridsearch over many different epsilon and min_sample values, optimizing for silhouette score. Using this search, we found that an epsilon of 0.45 and min_samples of 250 gave us a score of 0.253.

Results

As stated in the evaluation section, for our unsupervised learning approaches, we evaluated the results using three success metrics:

1. **Silhouette Score -** This was the most important metric, as it measures the closeness and distinctness of the identified clusters.
2. **Capturing Players with High NFL Likelihood -** A secondary metric aligned with the objective of identifying players likely to succeed in the NFL.
3. **Capturing Players with Low NFL Likelihood -** Another secondary metric to identify players with low probabilities of NFL success.

By prioritizing the silhouette score and incorporating the two secondary metrics, we were able to select the unsupervised learning model that best met the overall objectives of the project. The silhouette score provided the primary indicator of clustering quality, while the additional metrics ensured the model's outputs aligned with the goal of distinguishing players based on their NFL success potential.

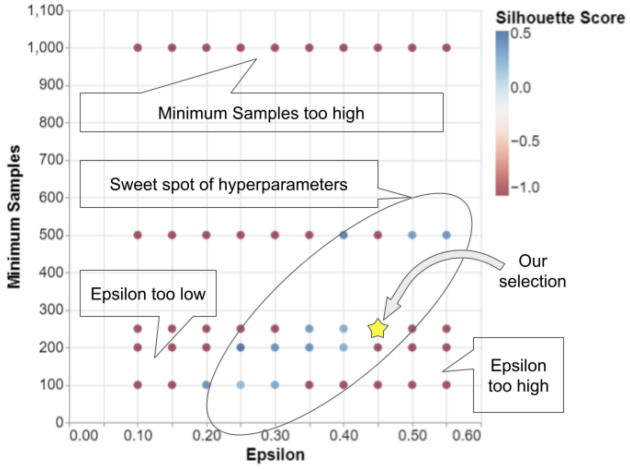
	A	B	C	D
	Unsupervised Learning Model Performance			
1	**Method**	**Silhouette Score**	**Highest Draft Percentage in Clusters**	**Lowest Draft Percentage in Clusters**
2	Simple K-Means	0.24	25.2%	5.8%
3	DBScan No Noise PCA	0.25	21.0%	3.6%

These modeling approaches were very similar but with just 0.01 in the silhouette score, the DBSCAN with PCA performed the best.

Best Model Sensitivity Analysis

Our highest performing model using Silhouette score was DBSCAN with PCA, slightly ahead of our simple K-means model. For our dimensionality reduction, we used PCA without any hyperparameter tuning, simply setting our desired number of components at two. With the DBSCAN portion, there were many parameters that we could have optimized, however there were only 2 that really had an effect on the clustering: eps (Epsilon), and min_samples.

Figure 7: DBSCAN Model Sensitivity



Epsilon is the max distance between samples for them to be neighbors, and min_samples is the number of samples in a neighborhood to have a core point and therefore a cluster. We found that DBSCAN was highly sensitive to the selection of eps and min_samples. With just slight variation, the resulting clusters could have silhouette scores as high as 0.25 or as low as -0.5. Additionally, with a poor selection of parameters, all points will belong to the noise cluster, which breaks Silhouette Score as there are no other clusters to compare cluster assignment of points to (Scored as -1 in Figure 7). This meant it was critical to optimize our hyperparameters to achieve effective clusters.

Supervised Learning

Utilizing our original dataset, and our newly found clustering features from both K-means and DBSCAN, the next step was to perform Supervised Learning to learn from our training set to predict if an athlete will be drafted from our testing set. We tried three different Supervised Learning models:

- Logistic Regression
- Support Vector Machines (SVM)
- Random Forest

For each of these models, we had to tackle dataset imbalance (only 8% of our target was 1, the rest were 0's). First we tried down-sampling and training the models on a 50-50 split of 1's and 0's; however, for each of our models we noticed a drop in performance. We believe down-sampling diminished performance because it reduced the amount of training data available. So, we pivoted to handling the imbalance by setting the class weight parameter to "balanced," which forces the model to pay more attention to the minority class when training the model. This technique is also called 'upweighting.'

Logistic Regression

For our first model, we used Logistic Regression, as it is a simpler model that we thought would be great for a baseline to see if we could capture more information from our dataset using more complicated or non-linear models. We preprocessed our dataset with sklearn's StandardScaler for numerical features, and their OneHotEncoder for categorical features. The StandardScaler improves performance by normalizing features to have 0 mean and a standard deviation of 1 which eliminates the model from over prioritizing larger numerical features over smaller ones. The OneHotEncoder allows us to perform logistic regression with categorical features, as it converts them to numerical binary values for modeling.

Support Vector Machines (SVM)

We decided to look into this approach due to its ability to catch any higher dimensional trends. Preprocessed similarly to the Logistic Regression, we preprocessed our dataset with sklearn's StandardScaler for numerical features, and their OneHotEncoder for categorical features. For hyperparameter tuning, we had the model iterate through four main kernel types: linear, polynomial, radial basis function (RBF), and sigmoid. We leveraged the GridSearchCV 5 fold and iterated through various values for the regularization parameter (C), gamma, and degree hyperparameters.

Overall, the model performed reasonably well against the test dataset. However, when we ran the cross-validation performance score, it was lower than the test or training scores. This made us less confident about the model's generalization ability.

To address this, if we had more time, we would have considered setting the regularization parameter (C) to a higher value. A higher C value would likely improve the model's ability to generalize by preventing overfitting to the training data.

Random Forest

Our best performing model ended up being a Random Forest. We began by training a base version of a model that acted as a baseline as well as a guide to understand what features were less important. This feature importance analysis led us to consolidate some features and entirely remove others.

Next, like with the two models above, we one-hot-encoded our categorical features and normalized the numerical features using sklearn's Standardscaler. In a different version of the model we didn't use any scaling, but this led to a significant decrease in model performance. This was a big surprise since tree-based models should be agnostic to normalization.

We leveraged randomized search with 10 iterations and 3-fold cross validation, and set f1-micro as our scoring metric to optimize for. See Figure 8 below for a breakdown of our search grid. It's worth noting that we initially

encountered some overfitting, so we changed the search grid to avoid this. Specifically, we reduced the max_depth values and increased the values for min_samples_leaf.

Figure 8: Random Forest hyperparameter tuning search grid

```
param_dist = {
    'n_estimators': [50, 75, 100, 150],  # How many trees
    'max_depth': [10, 15, 20, 30],  # How many splits in the trees are allowed
    'min_samples_split': [10, 15, 20],  # More options for minimum samples split
    'min_samples_leaf': [8, 10],  # How many samples are allowed to comprise a leaf
    'bootstrap': [True, False],  # Whether bootstrap samples are used
    'max_features': ['auto', 'sqrt', 'log2', None],  # Different ways to limit the number of features
    'criterion': ['gini', 'entropy'],  # Different criteria for splitting nodes
    'class_weight': ['balanced']
}
```

For a more detailed visual summary of all the decisions we made during model development, and the score resulting in each version of random forest, please refer to Appendix D.

Supervised Model Evaluation

In order to evaluate our supervised learning models, we had to decide on an evaluation metric that best fit our particular use case. There are many different options such as accuracy, precision, or recall; however, our team decided to use f1-score. F1-score is calculated by taking the harmonic mean of precision and recall. This gives us the advantages of both, namely that we care both about minimizing false positives and false negatives. Additionally, we set the average hyperparameter to micro, which allowed us to better optimize our models for our minority class which is the players who were drafted. Other metrics such as accuracy may show strong performance simply selecting the majority class every time, which would always misclassify our minority class. Taking all this into account, f1-score was the best option for our particular use case.

	A	B	C	D
	Team #29: College Football to NFL Draft Model Performance			
1	**Metric**	**Random Forest**	**SVM**	**Logistic Regression**
2	F1-Score (Test)	0.8602	0.842	0.7479
3	F1-Score (Train)	0.917	0.846	0.7415
4	Mean of F1-score across 5 fold CV	0.865	0.725	0.725
5	St. Dev. of F1 score across 5 fold CV	0.003	0.027	0.013
6	# of Features	20	11	19
7	# of Categorical Features	6	6	8

Sensitivity Analysis

To better understand the impact of our Random Forest's model parameters, we used Randomized Search with Cross Validation to train 100 models on permutations possible in the search grid referenced above in Figure 9.

For most combinations the results were fairly stable; in fact, on 80% of models the f1-score ranged from .80 to .90 (see Figure 9 to the right). However, in some instances, the model returned f1-scores as low as 0.66.

After digging in we realized that most of the lower model scores were a product of max tree depth parameters being too low. Specifically, the majority of the bottom 20% of models occurred when the max tree depth parameter was set to 10. This resulted in the trees not being able to grow deep enough to learn our data, which led to an underfitting model.

We highlight this phenomenon below in Figure 10. Note how models with a max_depth of 10 always underperform other models, even while controlling for other hyperparameters:

Figure 9: Distribution of f1-scores on 100 trained models

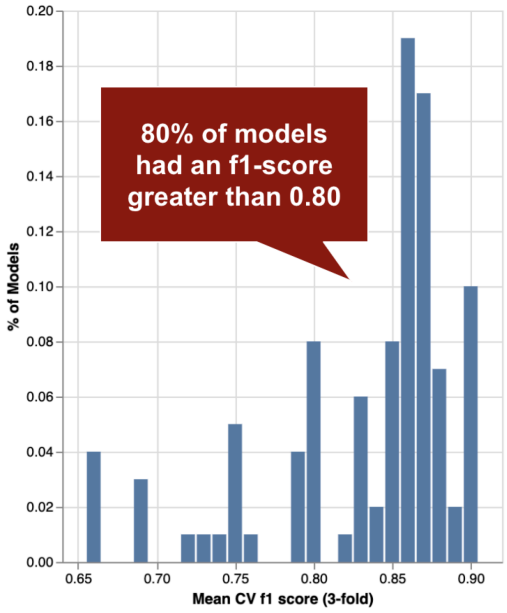
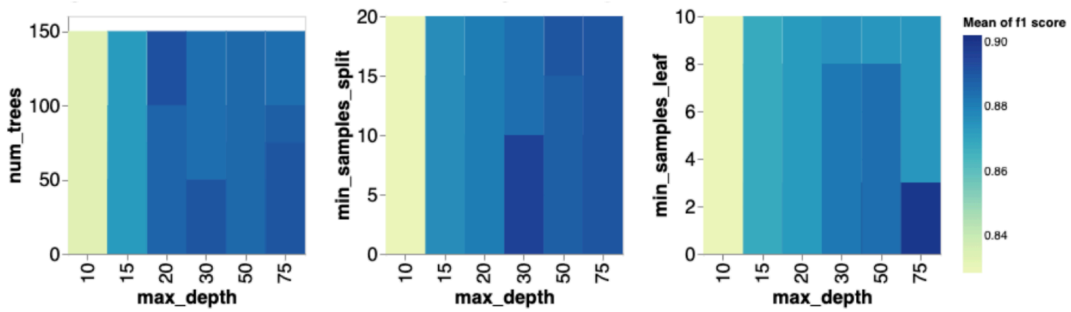


Figure 10: Average f1-scores of Models - Broken Down by Max Depth Value vs. All Other Parameters *

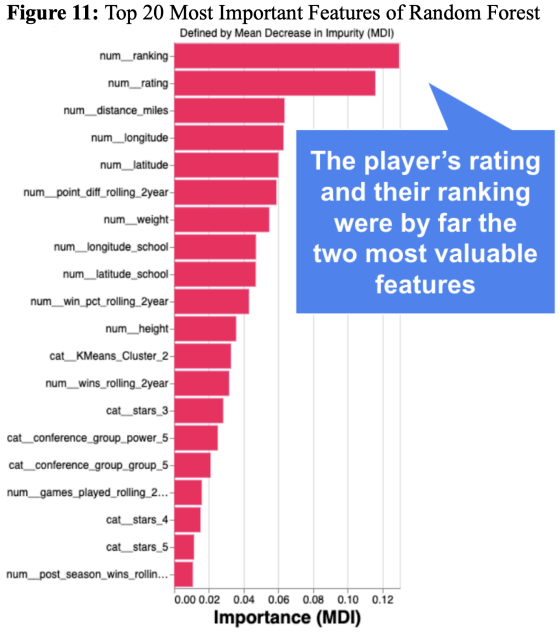


** Only includes numerical features - excludes categorical ones like criterion and bootstrap*

Feature Importance and Ablation Analysis

By far the two most important features were the player's rating and their rank among recruits for their graduation year. Surprisingly, features like the player's height, weight, and position didn't even make the top 10 (see Figure 11). Features like the location of their hometown and the quality of the school they committed to were also important. Interestingly, the closer the player was to the school they attended the more likely they were to get drafted, suggesting that good football colleges yield good football high schools (or vice versa).

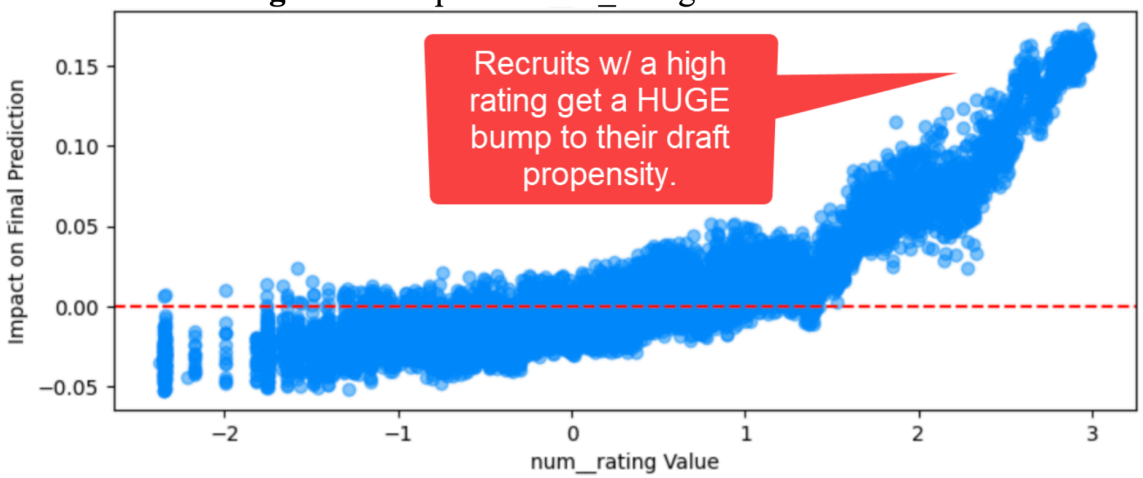
Figure 11: Top 20 Most Important Features of Random Forest



The least important features were the ones that didn't appear very often. For example, players who committed to an independent school (ie a school that's not in any conference at all) didn't get much predictive power from that feature. Additionally, players who play odd positions such as kickers and long-snappers also didn't yield much predictive power.

We can leverage the SHAP library to conduct an ablation analysis. Specifically, we can remove our most important feature to see how individual predictions are influenced both with and without it in the model. See Figure 12, which shows how each individual player's prediction was changed by adding in the rating feature - note how some players with a high rating saw their draft likelihood increase by 16% with the addition of this feature (more on this later in the ethical considerations section).

Figure 12: Impact of num_rating on Final Prediction



Failure Analysis

Failure #1: Our highest propensity player who ended up going undrafted.
We had one player (Trent Thompson, index = 3919) that showed all the right signs of being drafted - he was the #1 player in the country, he committed to one of the best schools (University of Georgia), and he was even from the southeast which is where a lot of the most talented players come from. These were all features captured in our model, so the model predicted he had a high likelihood of being drafted.

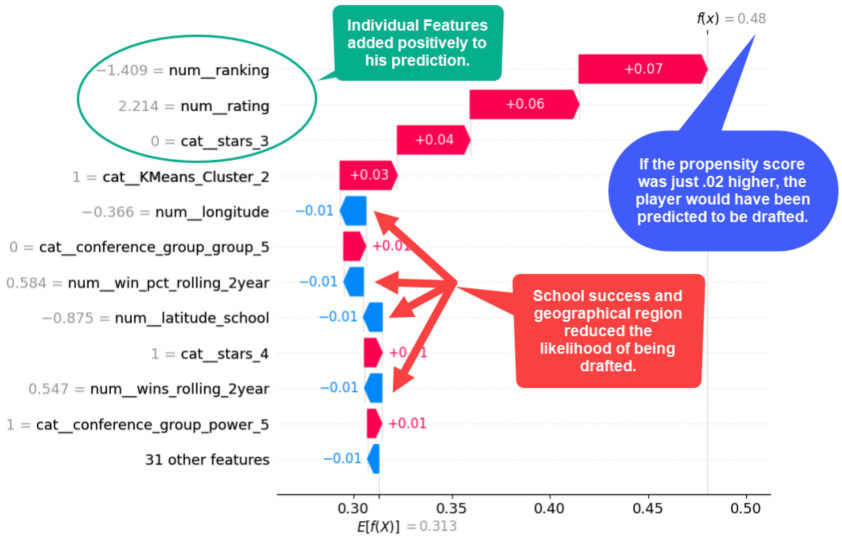
He had a tremendous freshman and sophomore season, but early in his junior year he suffered a torn labrum in his shoulder and two sprained MCL ligament injuries [9]. Despite these huge setbacks he "unexpectedly" declared for the NFL draft and went undrafted.

This is an unusual scenario that would be difficult to account for, and we think our model was correct in predicting him to be drafted given the information available. Injuries like this are hard to predict, but perhaps we could include an injury propensity model as a feature. See Appendix C for an ablation analysis on this player.

Failure #2: A highly rated player who got drafted, but our model chose to downgrade.
We had one player (Jalen Reagor, index = 979) who was an above average recruit who our model incorrectly predicted would not be drafted. While our model did increase the likelihood of him getting drafted due to his

individual features (see green annotation in Figure 13). It downgraded his likelihood because he committed to TCU, a lower tiered school with below average success at the time (see red arrow annotations).

Figure 13: Failure #2 Local Bar Plot

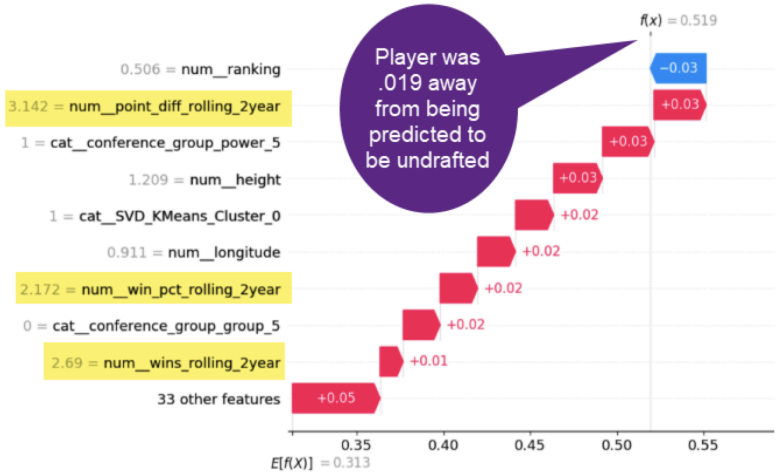


However, our model did incorrectly discount the prediction due to the recruit's hometown coordinates. The recruit is from a town nearby Dallas, TX, but its longitude coincides with states that have a relatively low draft rate like Kansas, Nebraska, and the Dakotas. In the future, I would try re-structuring the geographical features so that longitude and latitude are not separate features.

Failure #3: A low-rated player who went undrafted, but our model chose to predict highly.

We had one player (Cameron Gardner, index = 979) who was a pretty average recruit. He was a 3-star prospect, and was just a little above the average rating. However, he received a huge bump in draft propensity due to the quality of college team he committed to (see highlighted portion in Figure 14), and even then, he just barely was predicted to be drafted.

Figure 14: Failure #3 Local Bar Plot



However, the features about the quality of the team were actually part of a data quality issue. There were two errors in our code that caused this. The first was that any player who committed to this particular school (Wake Forest) had missing data for the "Team Momentum" features. This error was due to us beginning our indexing at 1 instead of 0, which caused this particular school to populate NULL values.

The problem was exacerbated with another error in our imputation logic. Our intention was to impute missing team momentum features with the mean of the conference they compete in; however, we unintentionally imputed with the max value rather than the mean. The effect was that any player who committed to Wake Forest,

got a boost to their propensity as if they committed to Clemson. This is something we would certainly fix in another iteration.

Discussion

Supervised Learning

There were many surprises that we uncovered throughout the creation of our supervised model. Perhaps the most interesting is that player stars, ranking, and rating (all related attributes) played a huge role in player outcomes. Generally the scouts seem to have great predictive power, however our other attributes still played a significant role in predicting players to be drafted. Another surprising find was that our SVM model performed significantly worse on cross validated results, leading us to believe additional regularization could improve its performance.

Our biggest challenge was the imbalance nature of our dataset, and the issues that arise predicting on a minority class that makes up only about eight percent of the entire dataset. Techniques like downsampling seem like viable options but ended up hurting model performance compared to using a balanced class weight parameter in our model. Another challenge is that hyperparameter tuning took a long time to search especially with models like SVM where additional computing resources would have helped speed up our optimization. We were also surprised to see a reduction in model performance for the Random Forest when we didn't normalize our numerical data; which is odd since tree-based models typically don't need to be scaled.

With more time we would have liked to bring in more data from previous years, as well as more data on player stats in high school. This additional data could have uncovered more hidden features, and even showed trends over time on how our variables' predictive power for being drafted changed through different eras in the NFL.

Unsupervised Learning

During the unsupervised learning portion, we gained some valuable insights of our dataset like how there is a group of players that are located in the American Samoa Island. Each of our clustering approaches had a cluster that would group these players together due to their distance to their college they committed to.

The biggest challenge was learning the hard way that t-SNE cannot be leveraged to classify an unknown datapoint and predict. We put in the time to create these clusters with t-SNE that had a silhouette score that doubled our other approaches and produced these beautiful figures, but since t-SNE could not have been used within our classification model we had to drop these visuals and focus on our other clustering approaches.

In addition to grabbing more data points and more data, if we had more time/resources, we would be likely to experiment with the features more in the future. Incorporating more categorical variables within the K-Means clustering by first taking the PCA would have been interesting to experiment with.

Ethical Considerations

As we wrap up this report, we've highlighted some of the ethical considerations that could arise:

1. Recruiting rating bias
2. Unprotected class impact

Recruiting rating bias

Recruiting rating has been shown to be a top feature in identifying whether a high school football player will play in the NFL. However, it's important to keep in mind the biases that come with this feature. Since these ratings are based on recruiters, it is very subjective and can have some underlying biases that the recruiters have. Additionally, now that we know this feature strongly impacts their draft likelihood, recruiters could be incentivized to receive monetary payments to falsely inflate a recruit's rating.

Unprotected class impact

In our anomaly detection section, we observed a group of players from Pago Pago in the American Samoa community, identified solely by their hometown coordinates. This illustrates how location can inadvertently identify a non-targeted group. Utilizing distance and coordinates in our models might unintentionally affect predictions about their chances of reaching the NFL. Our unsupervised model, such as the K-means cluster, may also inadvertently group them together based on these factors.

Statement of Work

Categories	Andreea Serban	Chris McAllister	Ryan Obear
Data Collection	API setup, Web scraping , EDA, Anomaly Detection	Combining the datasets , data manipulation, preprocessing, EDA	Environment Setup (Requirements.txt)
Unsupervised Learning	Simple K-means, Analyze all clusters	K-means with SVD , Add Clusters to final dataset	t-SNE, DBSCAN with PCA
Supervised Learning	Support Vector Machine (SVM)	Random Forest , Failure Analysis, Sensitivity Analysis	Logistic Regression, Sensitivity Analysis
Other	Writing Report, Git Repo	Writing Report, Git Repo	Writing Report, Git Repo

Appendix A: Bibliography

[1] "Estimated Probability of Competing in College Athletics." NCAA, 2 Mar. 2015, www.ncaa.org/sports/2015/3/2/estimated-probability-of-competing-in-college-athletics.aspx.

[2] Aserban13, chrismca. "Milestone I Git." GitHub, github.com/aserban13/SIADS593-MilestoneI.

[3] Nicholas Wheeler. "Do High School Football Recruit Ratings Accurately Predict NFL Success?" Scholarship @ Claremont, scholarship.claremont.edu/cgi/viewcontent.cgi?article=3035&context=cmc_theses.

[4] Hgorelick. "NFL Draft Analysis." GitHub, github.com/hgorelick/NFLDraftAnalysis/tree/master.

[5] "College Football Datasource." College Football Data.com, api.collegefootballdata.com/api/docs/?url=/api-docs.json.

[6] "Pro Football Draft Dataset." Pro Football Reference, 2022, www.pro-football-reference.com/years/2022/draft.htm.

[7] "American Samoa: Football Island." CBS News, 17 Sept. 2010, www.cbsnews.com/news/american-samoa-football-island-17-09-2010/.

[8] "T-SNE Transform Does Not Exist." Stack Overflow, 6 Dec. 2019, stackoverflow.com/questions/59214232/python-tsne-transform-does-not-exist.

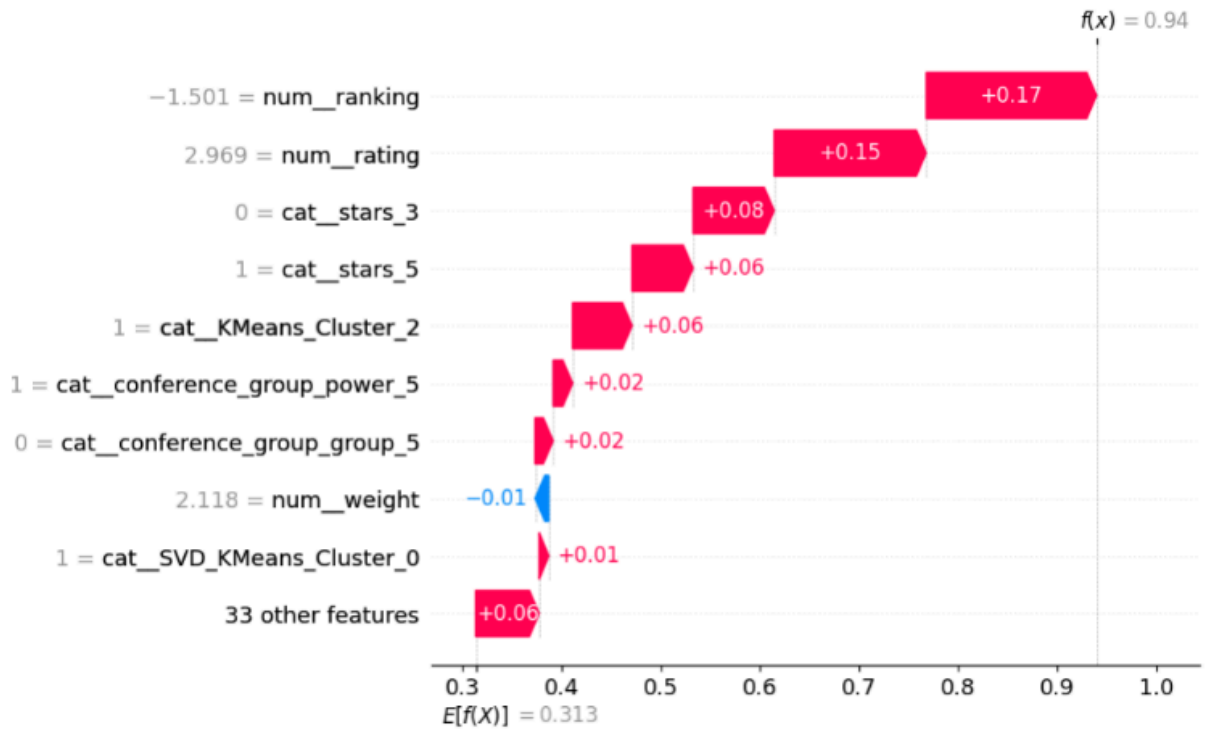
[9]]Whitfield, J. "Former Georgia DL Trenton Thompson still pursuing a dream and a football career." USA Today, 7 Jan. 2020, ugawire.usatoday.com/2020/01/07/former-georgia-dl-trenton-thompson-still-pursuing-a-dream-and-a-football-career/.

[10] Aserban13, obearyan. "Milestone II Git." GitHub, github.com/aserban13/SIADS696-MilestoneII.

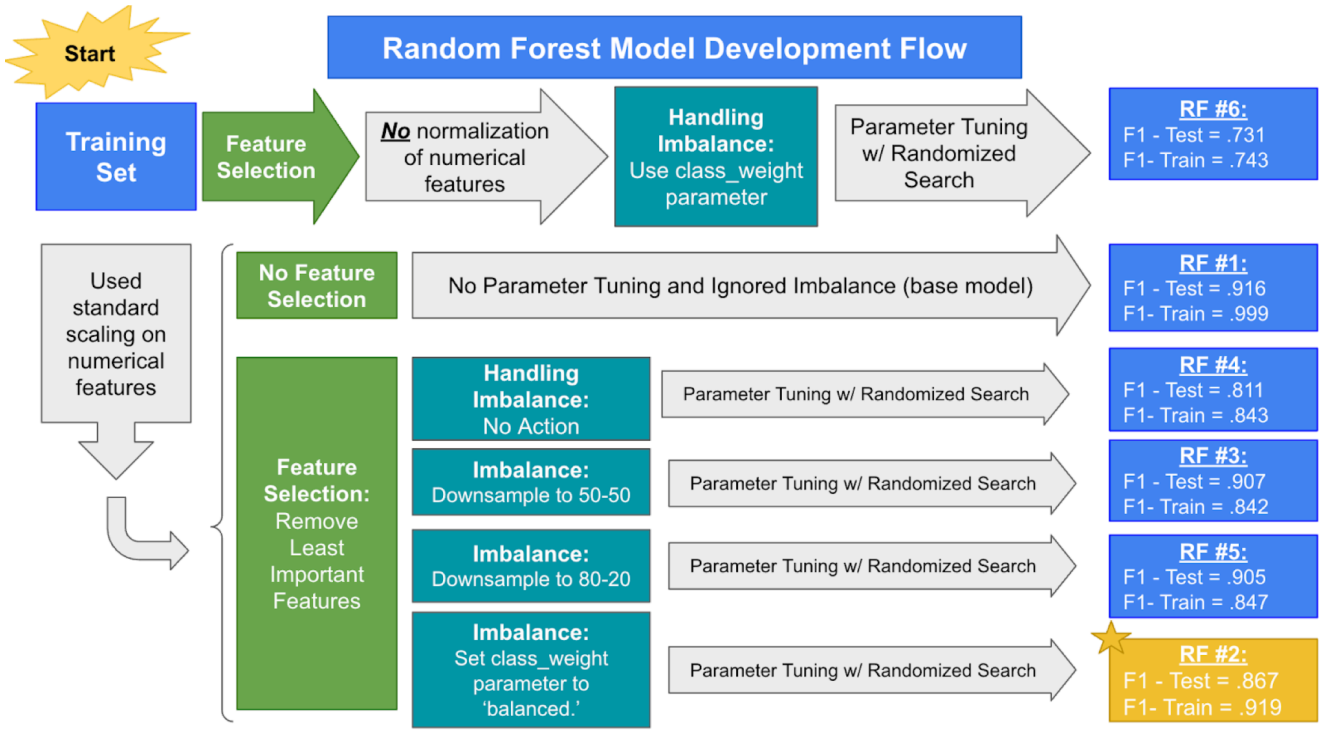
Appendix B: Dataset Descriptions

Column Name	Description	Data Type	Feature Flag?	Source	Calculated from other colu...	Example
year	High School year that they graduated	int	No	CFBD - RecruitingApi	No	2015, 2016
name	Player's Name	string	No	CFBD - RecruitingApi	No	Aidan, Antonio, ..
is_drafted	Indicate whether the player got drafted in the NFL	bool	No - Target	Draft Data	No	1 or 0
wins_rolling_2year	Committed college team's win percentage of the over the previous two years before the player committed	float	Yes	CFBD - GamesApi	Yes	20, 40,...
win_pct_rolling_2year	To measure trend between the two prior years for the committed college, take the win percentage difference between the two years ago and a year ago	float	Yes	CFBD - GamesApi	Yes	0-1
post_season_wins_rolling_2year	Number of games won by the committed college team over the previous two years before the player committed	float	Yes	CFBD - GamesApi	Yes	0.0, 1, 3.0
point_diff_rolling_2year	To measure trend between the two prior years for the committed college, take the point difference between the two years ago and a year ago	float	Yes	CFBD - GamesApi	Yes	-112, 100
games played_rolling_2year	Number of games that the committed college team played two years before the player committed	float	Yes	CFBD - GamesApi	Yes	11,.0 14,.0 8.0
weight	Weight of the high school football play (lbs)	float	Yes	CFBD - RecruitingApi	No	220
state_province	State the high school is in	string	Yes	CFBD - RecruitingApi	No	GA, MI
rating	High School Rating based on the stars given by the recruiter	float	Yes	CFBD - RecruitingApi	No	0.7-1.0
ranking	High School Ranking given based on the rating (inverse relationship)	int	Yes	CFBD - RecruitingApi	No	1-5,000
position	Football position that the player plays	string	Yes	CFBD - RecruitingApi	No	WR, OT, CB, S
longitude	High School longitude	float	Yes	CFBD - RecruitingApi	No	-81
latitude	High School latitude	float	Yes	CFBD - RecruitingApi	No	34.9
height	Height of the high school football play (inches)	float	Yes	CFBD - RecruitingApi	No	61.0-83.0
commited_to	University/College that the high schooler committed to	string	Yes	CFBD - RecruitingApi	No	Georgia
side_of_ball	Type of football player (based on position)	string	Yes	CFBD - RecruitingApi	Yes	defense , offense
position_group	Football player's position (based on position)	string	Yes	CFBD - RecruitingApi	Yes	d_line , pass_catcher
stars	Number of stars a recruiter gives to the high school football player	int	Yes	CFBD - TeamsApi	No	1-5
longitude_school	Longitude of the college player committed to	float	Yes	CFBD - TeamsApi	No	-81
latitude_school	Latitude of the college player committed to	float	Yes	CFBD - TeamsApi	No	34.9
conference	Football conference of the college player committed to	string	Yes	CFBD - TeamsApi	No	Big Ten, SEC
distance_miles	Distance between hometown coordinates and the college they committed to coordinates (miles)	float	Yes	CFBD - TeamsApi	Yes	0, 400
SVD_KMeans_Cluster	K-means with SVD Cluster	string	Yes	Mix	Yes	0-3
KMeans_Cluster	Kmeans Cluster	string	Yes	Mix	Yes	0-4
DBSCAN_Cluster_PCA	DBSCAN with PCA Cluster	string	Yes	Mix	Yes	-1 to 22

Appendix C: Local bar plot - Trent Thompson



Appendix D- Visual summary of modeling techniques and decisions for Random Forest


Start
Random Forest Model Development Flow
Training Set
Feature Selection
No normalization of numerical features
Handling Imbalance: Use class_weight parameter
Parameter Tuning w/ Randomized Search
RF #6:
F1 - Test = .731
F1- Train = .743
Used standard scaling on numerical features
No Feature Selection
No Parameter Tuning and Ignored Imbalance (base model)
RF #1:
F1 - Test = .916
F1- Train = .999
Feature Selection: Remove Least Important Features
Handling Imbalance: No Action
Parameter Tuning w/ Randomized Search
RF #4:
F1 - Test = .811
F1- Train = .843
Imbalance: Downsample to 50-50
Parameter Tuning w/ Randomized Search
RF #3:
F1 - Test = .907
F1- Train = .842
Imbalance: Downsample to 80-20
Parameter Tuning w/ Randomized Search
RF #5:
F1 - Test = .905
F1- Train = .847
Imbalance: Set class_weight parameter to 'balanced.'
Parameter Tuning w/ Randomized Search
RF #2:
F1 - Test = .867
F1- Train = .919