Greg Vargas

Final Report:

Analyzing NFL Team Statistics to Predict NFL Draft Order

Problem Statement

Every April, 255 of the top football players in the country sit anxiously awaiting a call, informing them that they have been drafted by one of the 32 teams in the National Football league. They have been training their whole life for this moment, and rightfully so, as they look to score the two biggest wins in their sport- the Superbowl and a massive contract. In 2020 the 'rookie compensation pool' was $1.4 billion across all teams, with the first and thirty second picks getting $36 million and $10.8 million, respectively. With that much money and success on the line, it is possible to analyze historical data and determine the player position they should draft? Can predicting the draft order be determined by the team needs, without looking at what players are eligible for the draft this year?

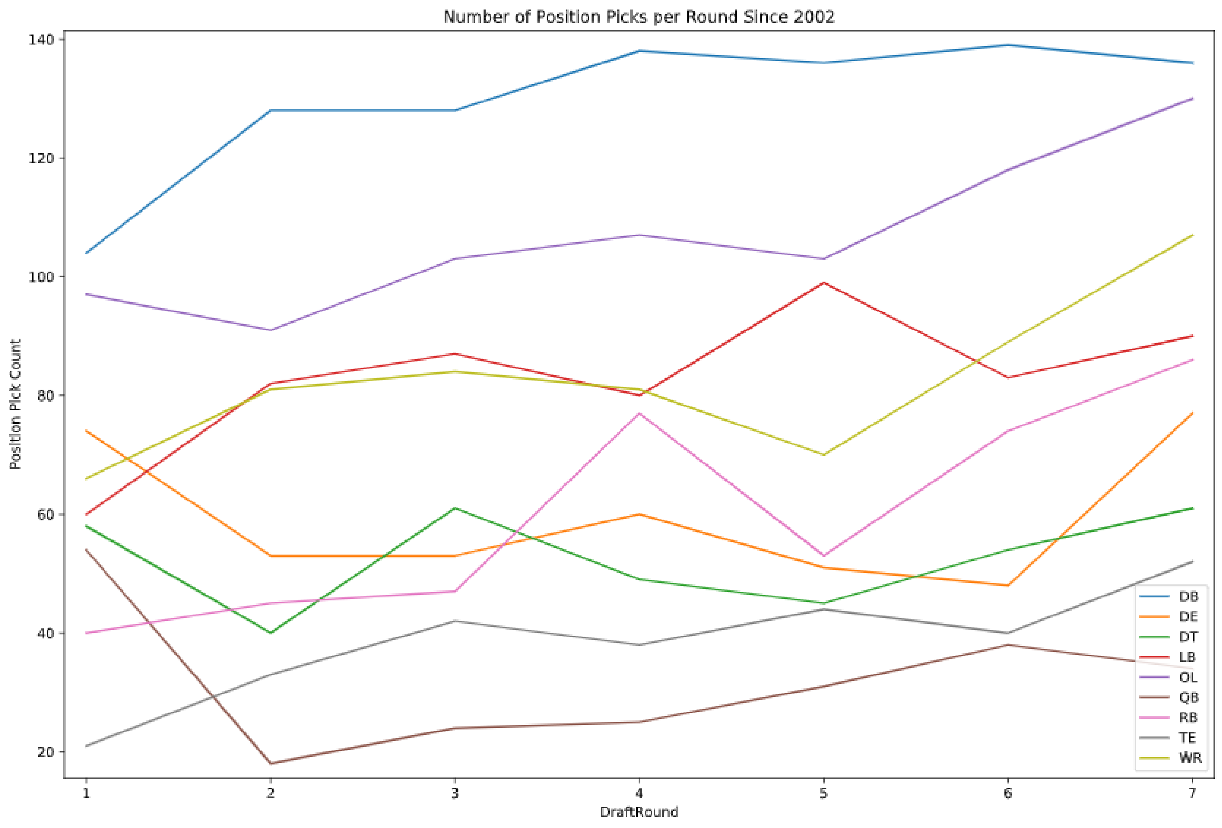
Data Wrangling

To create my dataset, I web scraped data from Pro-Football-Reference.com. I scraped data for each team and year from 2002-2019, which included yearly/weekly game summaries, starting player information, and draft order. I chose 2002-2019 because 2002 is the first year with all 32 current NFL teams (Houston Texans being added as an extension team), and 2019 being the last full season of data on Pro-Football-Reference. This information was then added to a sqlite database, to allow for more flexibility by running analysis on different query results. Some positions, such as cornerback(CB), safety(S), free safety(FS), had to be grouped into one position (Defensive Back(DB)) because the players are drafted for a generalized position.

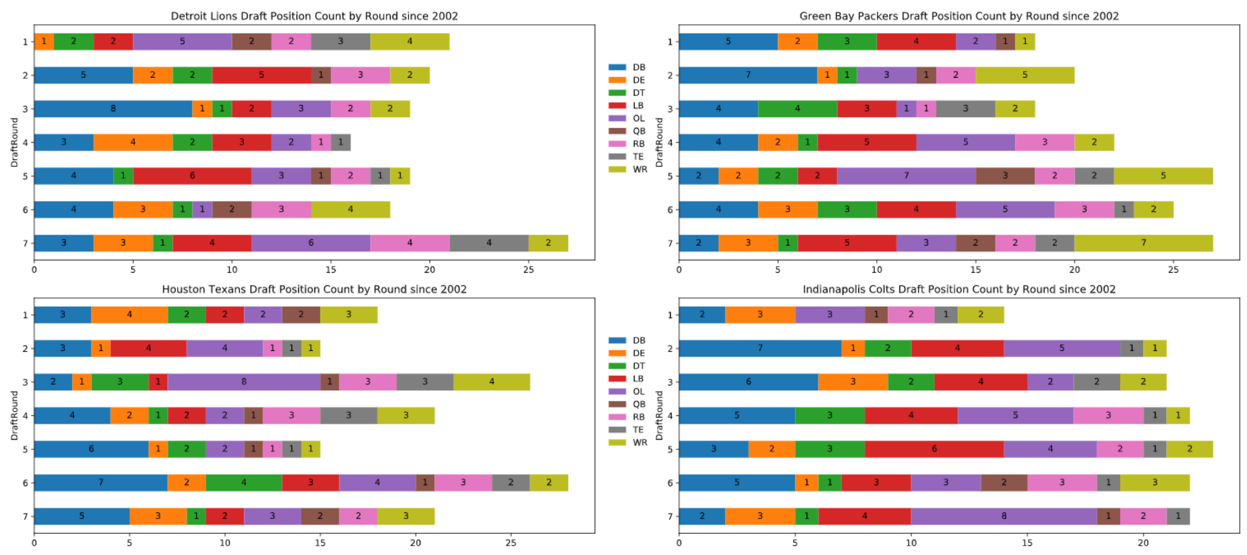
Exploratory Data Analysis

The 4 tables created by the web scraper were then turned into 4 dataframes: 1) yearly summary 2) weekly summary 3) Starting Player Stats 4) Draft order. Because I scraped both the weekly and yearly summary statistics, I did all transformations on both dataframes to compare how the different granularity of data performed in each model. I then found a dataset with player rankings, and joined that table on to each dataframe, creating 4 total dataframes to run through the model. I also tried to create features that I believed would add to the predictive quality of my model. To do this, I had to pretend I was an owner/coach of an NFL team, and see how I would determine who I would be drafting next year based on my current players. Features created included the number of years the starter had been with the team, and the average positional ranking of each player. With teams being able to trade draft picks up to the day of, I removed the 'draft round' count and converted it into the team pick order. By doing this, each draft pick would be an ordinal feature, as opposed to a nominal feature, with multiple draft picks in the same round.

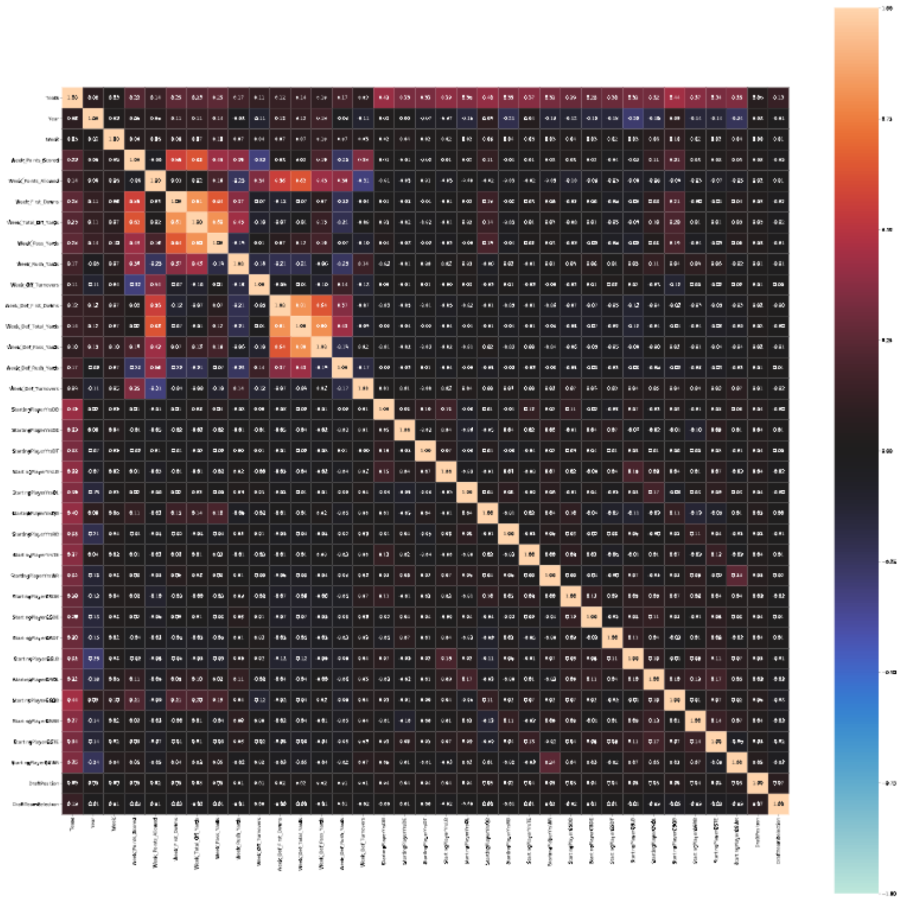
Analysis of the total draft pick counts by position showed some interesting trends. There are 130 NCAA Division 1 FBS football programs, each with their own elite players vying for a spot in the NFL. With the quarterback position being one of the most important and visual positions, it surprised me to see just how few quarterbacks have been drafted since 2002. As seen in the figure below, there was a 66% drop just from the first round to the second.



While analyzing the different trends, I created a bar chart of each team's draft breakdown since 2002. When looking at the draft selections by team, you can clearly see the impact of Hall of Fame caliber players for many years. For instance, the Indianapolis Colts had Peyton Manning as a quarterback from 1998 – 2010. As seen in the below snippet of a plot, the Colts only drafted one quarterback in the first 5 rounds in 17 years. This one pick was Andrew Luck, who was selected to replace Manning after his neck injury.



After creating the final datasets, I used a data science package called dython, to make a figure that shows correlation heatmaps with both numerical and categorical features. As seen in the figure below, there are a few features with high correlation, but none that show any with the two target variables. This was an area of concern, but the hope was that a model such as Random Forest would help find features that could predict the correct classes.



Model Preproccessing

Because I was going to be testing multiple models on multiple dataframes, I put everything in a pipeline within a loop. Because of the nature of handling this, I used ColumnTransformer to handle both categorical and numerical features before running the model. For categorical features, such as team name, coach, and position, the feature was one hot encoded. Although there was no missing values in the dataframe, when mapping out a groupby into a single index dataframe, missing values popped up throughout. These values were imputed with the median values as part of the pipeline. After imputing the missing values, all numeric features were transformed using a minmax scaler. Features such as wins were single digit numbers, while yards were 4 digits, and needed to be in the same range to give equal weight to the features.

Model Selection

By reconfiguring the target variable, between team draft pick and position pick, this problem was able to be run as a regression and classification model. The original 4 datasets were run in a regression model, with poor results. The categorical features in the datasets were being encoded into too many additional features because of their high cardinality, and the curse of dimensionality was coming into play. To remove some of the encoded features, the categorical features were bucketized to include the values with many instances and an 'other' bucket. The results of these new datasets did not provide any additional accuracy of the model. These regression models included Ridge and Lasso regularized models, to penalize additional features, but still did not improve the model. The categorical models were then run with the original datasets and included Random Forest and SVC models. PCA was also applied to the model, to remove any collinearity between the numerical features. With still low levels of classification accuracy, the Weeks dataset was chosen to compete against a dummy classifier. This was done to see how much better the model was doing against chance. Random Forest and SVC classifiers were then run with multiple hyperparameter models against a dummy classifier using a stratified, most frequent and uniform strategy. The Random forest classifier had the top 5 accuracy scores within the 54 model configurations attempted with GridSearch Cross validation. This also had a an 84% increase in accuracy over the best dummy classifier, with 37% accuracy vs 20%. The mean test score of my 5-fold cross validation yielded a mean test score of 36.7%, and a model accuracy of 37%. This means that the model generalized to new data well and had a similar score to the cross-validation scoring.

Takeaways/Further Research

Although the model did better than a chance classification, it still was correct less than half of the time. Further feature refinement can be done to increase the accuracy of classification, but it is unclear how much this would increase the accuracy. The data that was scraped may also not have the predictive power needed to complete such as prediction, so further research into more quality data may be needed to increase the predictive power needed to use this in a commercial setting. Further research into other classification would be ideal, as things like Random Forest can handle the high cardinality categorical features, without the feature predictive power being diluted by encoding.

References

https://www.pro-football-reference.com/

https://github.com/leesharpe/nfldata

https://www.forbes.com/sites/kurtbadenhausen/2020/04/24/2020-nfl-draft-first-round-rookie-salary-projections-what-burrow-tua-and-chase-young-will-make/?sh=703b5eff5be3