"Modeling NFL Season Success: Developing and Evaluating a Novel Approach to Forecasting Team Wins"

The nature of the National Football League (NFL), with its complex strategies, passionate fan bases, and its unpredictableness, ironically makes it a fascinating subject for predictive modeling. Inside all of this uncertainty lies a question: Can the outcome of a team's entire season be accurately forecasted before the first kickoff?

My personal motivation for developing this model stems from both a curiosity sparked by a witnessed account of the pitfalls of unfortunate optimism, and a keen interest in recreational sports betting.

With hopes of making an easy profit, a friend of mine purchased New York Jets season tickets immediately after superstar quarterback Aaron Rodgers was favored to join the team. This decision immediately got flipped on its head after Aaron Rodgers suffered a season ending injury on the third play of his season, further portraying the harsh unpredictableness of the NFL.

In April, 2023, I placed a $100 bet on my favorite team, the Pittsburgh Steelers, to win at least 10 games during the 2023 season. The Steelers were already coming off of a 10 win season, their schedule was more favorable than the previous year's, and the now second year quarterback would seemingly improve from his already decent rookie year. (He did not). At +165, the odds, combined with my admitted biasness, this $100 was guaranteed to produce free money.

If there were a way to model the amount of games an NFL team will win, we wouldn't have to purchase our season tickets based on intuition, we wouldn't have to throw our child's college fund on a bet based on biasness.The significance of modeling this phenomenon extends beyond making a quick buck. People make a living off of investing in season tickets, as well as

‘professional’ sports betting. The viability of these investments is dependent on the team's anticipated performance. One that we can hopefully, accurately predict.

Historical attempts to model NFL team performance offer a foundation upon which to build. Austin Streitmatter's model, using linear regression on basic statistical data, achieved a 70.5% success rate in predicting weekly game outcomes. However, this approach did not extend to forecasting a team's total wins for the season, only weekly winners.

Additionally, the Nfelo web application stands out as a noteworthy model in this field. Nfelo employs a vast amount of metrics to model NFL team performance, serving as both a source of inspiration and a benchmark for comparison.

Predicting the number of games an NFL team will win during the regular season is a task that I look forward to attempting to model. It combines three of my favorite hobbies: mathematics, watching football, and gambling (not in a sad, addictive way). And in the upcoming paragraphs, we will explore all of the preliminaries it takes to create this model.

Defining Phenomena and Models

A phenomenon is something of interest that occurs in the natural world. This can be anything. It can be the speed of an Olympic swimmer, or even the reason why my mother doesn’t love me anymore. The phenomenon I am interested in is the ability to predict the number of games an NFL team will win. This phenomenon is inherently complex due to the unpredictable nature of sports, particularly in a league as competitive and physical as the NFL. Factors like player injuries, team dynamics, weather conditions, and even subtle shifts in coaching strategies significantly influence the outcome of games, and by extension, the season. Therefore, predicting

an NFL team's performance is not just about analyzing static data but also about understanding the fluid and often chaotic nature of the sport.

In order to make predictions of our phenomenon, we need to use something called a model. A model is a representation of reality designed to understand, explain, or predict aspects of the real world. The goal of our model is to create a simplified yet effective representation of the complexities of NFL games in a way that provides meaningful predictions. Our model will take a number of settings, and it will take an algorithm, and it will spit back our y, which is how many games our team will win. Causal drivers are the real world settings that are inputted into our model. The causal drivers of our phenomenon are

- z1 = How healthy is our team during each of our games
- z2 = How good is our team when fully healthy
- z3 = How good is our scheme against each specific team
- z4 = What is the weather for every game we play
- z5 = How clutch is our quarterback
- z6 = How clutch is our kicker
- z7 = How healthy is our opponents team for when they play us
- z8 = How good is our opponents team when fully healthy
- z9 = How good is our opponents scheme against our team
- z10 = How clutch is our opponents quarterback
- z11 = How clutch is our opponents kicker

Right now, the settings that my model would take in are ambiguous. There's no way to actually measure them. We need to convert these settings into quantifiable metrics, and we will then be able to use a mathematical model. A mathematical model is a description of a concrete system

using mathematical language and concepts, and it allows us to quantifiably measure our metrics. Our metrics will be inputted into our model, and we will then be able to make a prediction.

To determine what my inputs are, I decided to look at some of the most practical, and accessible data. I want to know how good my team was last year, how much better (or worse) my team has gotten from last year, how good my opponents were last season, and how much better (or worse) my opponents have gotten. In order to know how good my team was last year, I can just see how many games they've won. It's a super easy and accessible stat to find. And in order to see how good my opponents were, I can do the same for them. In order to see the skill difference for this year's team compared to last years, I will need to compare this year's roster to last years. Pro Football Focus (PFF) produces a score for every (starting) player in the NFL. If I take the average all of the scores that PFF gave to my players before the start of last season, and compare them to the average all of the scores that PFF will give to my players before this season starts, and then I subtract this years average from last years average I will get a score that is in R that I could then put into my model. For example, if a team had an average score of 68.5 before the start of the 2021 season, and then they had a score of 71 before the start of the 2022 season, then they would have a score of 71 - 68.5 which is 2.5. We can follow the same process for comparing my opponents roster to their last year's roster.

We now have our first steps of how our prediction target will be measured. We will be inputting our x's (independent variables, or features) into some undetermined algorithm, and we will get back how many games our team will win. Independent variables, also known as features, are the input variables that are presumed to influence or predict the outcome, and they are used to create models that estimate or classify the dependent variable. In our case, our features will be:

- x1 = how many games our team won last year

- x2 = the score we get when we take the average PFF scores of all of our starters last season, and subtract it from the average PFF score of all of our starters this upcoming season
- x3 = how many games each of our opponents won last year
- x4 = the score we get when we take the average PFF scores of each of our opponent's starters last season, and subtract it from the average PFF score of all of our opponent's starters this upcoming season.

These are good features to choose, because all of these features are easily accessible, and you can see the practicality of them. It's logical to assume if our team was good last year, and our roster got even better, we are still going to have a good team, and we would therefore win more games. Now that we have our features, we can set up the following function: $y = f(x1, x2, x3, x4) + L$, where L is Error Due to Ignorance.

There are three main types of errors: Error Due to Ignorance, Estimation Error, and Misspecification Error. Error Due to Ignorance is the error inherent in the data or the process being modeled, often due to randomness or natural variability in the system. It's considered "irreducible" because no matter how good the model is, this error cannot be eliminated. Estimation Error is the error that arises from the process of estimating the model's parameters. In our case, it's the difference between the model's prediction and the true outcome due to the limitations in the algorithm's ability to learn from a finite training dataset. Misspecification Error occurs when the chosen model is not the correct one for the data or the problem at hand, meaning the form of the model is incorrect. This could be due to incorrect assumptions, omitting

important variables, including irrelevant variables, or choosing the wrong type of model altogether.

I think Error Due to Ignorance will be my highest source of error. There are a lot of features of an NFL season that my model is not predicting on. Some of these features are somewhat impossible to measure, like injuries, and mid season trades. And because of that, it can oversimplify the complexities of an NFL season, and can lead to a significant bias if these features have a strong outcome on the season. Since my model isn't so complex and only uses four features, I think estimation error will be the least significant error. Especially because my model does not contain any fluctuating or noise sensitive features.

So now that we have our features, and we are able to express our features as a function, how are we going to be able to put them to work? How much should each of our features affect our model? To answer these questions, we have to use Supervised learning. Supervised learning is where an algorithm is trained on already known datasets, known as training data. Training data are the datasets used to train a machine learning model, consisting of known input-output pairs that enable the model to learn and identify patterns, relationships, or features necessary for making predictions or decisions.We can use supervised learning, along with training data to help me model my phenomenon. I can gather old training data of our features, and actually see the real outcome of what happened. For example, we already know the outcome of the Pittsburgh Steelers 2022 season, but we can use our features that are mainly based off the 2021 season, to "make a prediction" on what will happen in the 2022 season, then we can compare our model's prediction to what actually happened. If we did this for every single team for the past 5, 10, or even 20 seasons, our machine will be able to tell us what function to use, to actually input our features into, and how much each of our features will weigh.

Now that we have our training data, D, and a function, f, we have to find an algorithm, A, to finish the supervised learning process. An algorithm is a rule, or a set of instructions that will be applied to the features in my model. A candidate set, H, is the space where all possible algorithms lie. We take an algorithm from a candidate set. There are many algorithms in our candidate to choose from. This is known as the Model Section Problem, where during a modeling process we have a surplus of algorithms to choose from. Some of our algorithms we can choose from include OLS, SVM, and KNN. We need a way to know which A is best to use.

To determine which algorithm we should use, let's look at our feature x2. There are three teams I would like to talk about who were better in their 2023 season than in their 2022 season. The teams who were better are the Pittsburgh Steelers, Los Angeles Rams, and the Houston Texans.

The Pittsburgh Steelers won one more game in 2023 than they did in 2022, it's because they had a small upgrade at running back (which in my opinion is the least important position on offense), and their now sophomore quarterback played a tiny bit safer (he only turned the ball over four times on the season). The Los Angeles Rams won five more games in 2023 than they did in 2022. This is because their star quarterback, Matthew Stafford returned in 2023 form a season ending injury he sustained in 2022, and because of the addition of superstar rookie wide receiver Puka Nacua. Nacua set the all time rookie receiving yards record with over 1400, and was the runner up for the Offensive Rookie of the Year award. Because their team got much better, they were able to record much more wins. The Houston Texans had the biggest turnaround than any team in the NFL. They went 3-13-1 in the 2022 season, securing the second overall pick in the draft. In 2023, they went 10-7, winning the AFC South division, and even winning a playoff game. This is because with their second overall pick they drafted C.J. Stroud, a

quarterback who started from day 1, and had arguably the best season ever recorded by a rookie quarterback, nabbing him the Offensive Rookie of the Year award. And with the 3rd overall pick (which they traded up in the draft for) they selected Will Anderson, an instant difference maker who was the Defensive Rookie of the Year. Along with new head coach Demico Ryans, and a couple of other improvements in the wide receiver corps and the offensive line, this team was a significantly better unit from their 2022 version. You can see that because they had an instant rebuild, and a much, much, much better roster. They were able to record significantly more wins.

From this information, we can visualize a direct correlation between how much better your roster has gotten, and how many more games your team won, and we can see a straight line going through all of the data points. Because of the linearity of this correlation, it makes sense to use OLS as our A, and linear regression as our model.

We have our features, our function, and our algorithm. So what would be our null model? What is our model trying to do better than? The null model that I would be trying to beat, would be taking the amount of wins that my team had over the past 5 seasons, and averaging them out. Whatever our average is, that is the amount of games our null model is predicting our team will win. For example, in 2019, 2020, 2021, 2022, and 2023, the Pittsburgh Steelers won 8, 12, 9, 9, and 10 wins. So my null model would “predict” that for the 2024 season, the Pittsburgh Steelers will win 9.6 games. I would consider my model a success, if it’s prediction is closer to the actual result for the 2024 season than my null models. I guess we’ll have to revisit this in January.

Prediction error metrics are quantitative measures used to assess the accuracy of a predictive model by comparing its predicted values against the actual, observed values. They are an integral part of objectively assessing the performance and accuracy of a model. So for my model, I can compare the results of what my model would produce for teams during the 2023

season, vs the actual results of those teams during the 2023 season. After I compare and analyze these results, it would allow me to tweak and fine-tune the model, to ensure that it would produce the best outcome.

There are a lot of prediction error metrics to choose from. Some of them include: mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). For my model, I would choose MAE. Since my model is producing the number of games a team will win, I would want my prediction error metrics to show me exactly how many games my model was actually off by. MAE would do just that for me. It would take the absolute value of the difference between my model's prediction and the actual result, and it would show it to me. A downside of using MAE is that when there are a lot of features in the model, the MAE can get thrown off by its lack of sensitivity to the outliers, it would treat the small, outlier error with the same weight as a big, important error. Also, since MAE is one of the most simple prediction error metrics, it doesn't capture the noise, and the slight movement in the model, and could therefore be a faulty prediction error metric. But since my model only takes in four simple features, I'm not so worried about the potential downside of MAE.

In order to determine whether or not my model was useful, I will need to set a threshold of usefulness. My Threshold of usefulness will be set to within a game and a half of the Draftkings Sportsbook over/under wins line, at the time that my model made its prediction. Before each NFL season, Draftkings Sportsbook creates an over/under line of how many games a team will win. For example, at the time I am writing this paper, the over/under line on the Pittsburgh Steelers regular season wins is at 8.5. So if I think that the Steelers will win 8 games or fewer, I would bet under 8.5, and if I think they will win over 8 games, I would bet over 8.5. (I have never been more confident on a bet in my entire life and to whoever is reading this paper if

you want to get rich throw your mortgage on Pittsburgh Steelers over 8.5 regular season wins.) This line is set by a team of odds making professionals that take everything into account, and have created much better models than I can even think of. So if my puny model can come within a game and a half of their line, I will consider it a very useful model. For that reason, being within a game and a half of the Draftkings Sportsbook over/under lines is my threshold of usefulness.

There is a lot of risk for using a model to predict the number of games a team will win, because the users of my model will be extrapolating. Extrapolation is the process of estimating values outside the range of known data points. It's extending the existing data trend to predict future values. As opposed to Interpolation, which is the method of estimating unknown values that fall within the range of a set of known data points. It's like filling in the blanks between known values.

In the case of my model, users will be extrapolating because they will be estimating wins for a season that has not yet happened. Since it’s in a future season, they are going out of the range of the historical data. Due to extrapolation, we have to make sure that our models are always being tweaked, and made as best as they can.

In order to improve our model to the best we can, we need to use Effective Model Validation. Effective model validation usually involves dividing your dataset into at least two parts: one for training the model (in-sample) and another for testing it (out-of-sample). Sometimes, a third split for validation is used, especially for tuning model parameters. Techniques like cross-validation, where the data is divided into several subsets that are used in rotation for training and testing, can provide a more robust evaluation by ensuring the model is tested on different out-of-sample data.

I mentioned before that this is a relatively simple model, and our data is linear. This is good for our error metric MAE, but it can introduce us to a dangerous world of underfitting. Underfitting occurs when a model is too simple to capture the underlying patterns and complexities of the data it's trying to predict or classify. This often happens when the model doesn't have enough features, or the features used are not powerful enough to account for the variations in the data. As a result, the model performs poorly on both the training data and new, unseen data, indicating that it hasn't learned the essential structures of the dataset. Since our model only contains four features, they are both simple to calculate, and linear. It is possible that our model is underfitting, due to the fact that it can't capture all of the complicated noise from our simple features.

However, I don't think that my model would overfit. Overfitting occurs when a model is too complex and learns not only the underlying patterns in the training data but also the noise and random fluctuations. This results in a model that performs exceptionally well on the training data but poorly on new, unseen data because it has essentially memorized the training dataset rather than learning to generalize from it. Overfitting is often characterized by a large discrepancy in performance between the training and validation/test datasets. My model is not complex enough to model the noise and fluctuation on top of the real data

In my opinion, it is better to have a model that is underfitting, than one that is overfitting. If I underfit, I know right away that something is messed up, and I know that I'm going to have to make some major tweaks to my model. But if I overfit, I'm just going to think that I created a killer model, but once I use it on the testing data, everything is going to be way off and I would have to recreate a model.

We've finally finished discussing all of the boring stuff about our model. We can get into the practicality of it. My model can be used to predict by anyone who likes to sports bet, or anyone who buys and flips season tickets. All the person has to do is choose an NFL team they are interested in, go onto my hypothetical website that I have created for this model, and my software will gather up information on all of the features of the chosen team. It will then spit back the y, the amount of games the chosen team will win.

I can imagine sports bettors feening for this model. At some point during the NFL offseason, all of the sportsbooks release their lines on how many games an NFL team will win. If a sports bettor wanted to place a profitable bet, or an investment as I like to call it, they can just use my model. Now obviously my model won't be 100% accurate, as I've mentioned before, there are certain things that are just impossible to measure. But a sports bettor can expect a profit if he were to bet on every single team using my model.

Season ticket flippers can also use my model. There's a large number of people who buy season tickets for a team who they think will improve from last season. They would buy these tickets before the season starts, and then seek each ticket individually before every game, in hopes of making a profit. Fun fact the principal of my highschool, Rabbi Storch, would buy and sell season tickets for dozens of teams each year throughout the four major sports as a side hustle. I mentioned before how the Houston Texans had an amazing turnaround in just one offseason. If Rabbi Storch wanted to make some money, he would have looked into the Houston Texans, and he would have plugged them into my model. I can imagine the drool coming out of his mouth. He would be able to make a lot of money off flipping these tickets, all thanks to my model.

I titled this paper "Modeling NFL Season Success: Developing and Evaluating a Novel Approach to Forecasting Team Wins". I came up with this title before I even wrote the first word of my introduction. A full 11 pages later and I still do believe that the title of my essay is correct, and that it does accurately describe the modeling performance.

No model will be perfect, if it was, then it wouldn't be a model. I don't believe my model will be even close to perfect. However, due to the algorithm and the important, practical features, I do believe that it will definitely outperform the null model, and it will meet my threshold of usefulness. Hopefully one day I can actually put this model to use, and make my healthy hobby of recreational sports betting a profitable one.