Jonathan Langley, Yong Liu, Sujeet Yeramareddy
Professor Aaron Fraenkel & Professor Jelena Bradic
DSC 180B
9 March 2022

Applying Post-Prediction Inference to Sports Analysis

1. Introduction

For our project, my group decided to use our domain methodology research of post-prediction inference (postpi) and apply it to sports analysis based on the outcomes of past NFL games. We designed a model that can predict the outcome of a football game, such as which team will win and what the margin of their victory will be, and then corrected the statistical inference for selected key features. The main goals of our investigation is discerning which features most strongly determine the victor of a football game, and subsequently which features provide the most significant information to predict the margin of that victory. For example, does the home field advantage increase a team's chance of winning by 7 points? Is the comparative offense to defense rating the most critical factor in securing a win? Does temperature on gameday play a statistically significant part in influencing margin of victory? These are just some of the questions we have brought up to seek an answer during the course of our research, and by conducting this project we are attempting to revolutionize the way NFL analytics are conducted via a more accurate statistical method of inference, postpi.

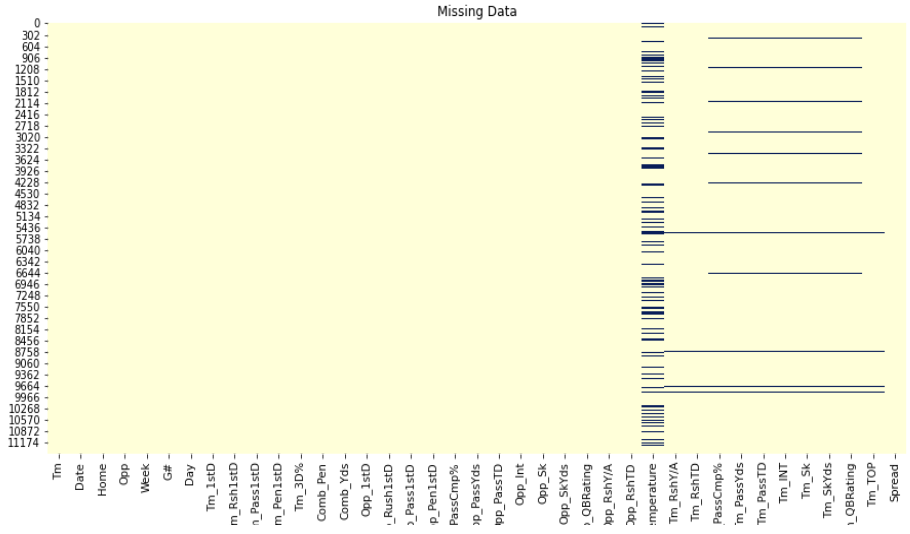
2. Data Collection

To begin a data science project, one requires data. Surprisingly so, the process of data collection was quite a bit more grueling than we had hoped. NFL datasets themselves and websites dedicated to NFL data are abundant yet almost none of them were helpful. We began our data search by hoping we would easily find one or two websites with all of the information we wanted neatly wrapped in uniform csv files just waiting to be merged together, however, much to our disappointment this was not the case at all. One website might have all of the offensive ratings for NFL teams from 2000-2021 yet lack any of the defensive ratings (which are just as important), and another website might have the complimentary defensive ratings but only from 1993-2010, and also set up in a completely unique format so as to make the prospect of data merging nightmarish at best. We had hoped that maybe such a scenario would only be true for a couple of the features we wanted, but as it were to get the minimum features we needed with sufficient data points to allow for any reasonable machine learning model to be accurate we were looking at having to merge 10+ websites' csv files all formatted uniquely and encompassing different year ranges. Our stroke of luck came when Sujeet discovered the website https://stathead.com/football/, which contains all of the features we needed, for all of the years that we needed, and all formatted in a very similar way. The catch was that we had to pay $8 a month each to get access to the csv files for the target features. Additionally, we only had access to 100 rows of data, in csv text form, at a time for each individual game statistic (you had to click next at the bottom of the page to go on to rows 200-299, etc)

and considering that there were 5,632 NFL games played from 2000 to 2021, and further considering we had 10 target features, this took a lot of tedious manual labor. The data only represented 5,600 games, however the number of observations was double this due to the fact that a game involved two teams and there are two entries for each individual game. This is an important aspect of our data because otherwise any model that we create down the line would bias our Spread predictions, because the Spread column would either only contain all positive values or negative values. To avoid this, we included all the observations in our dataset, which also further increased the time it required to collect all the data.

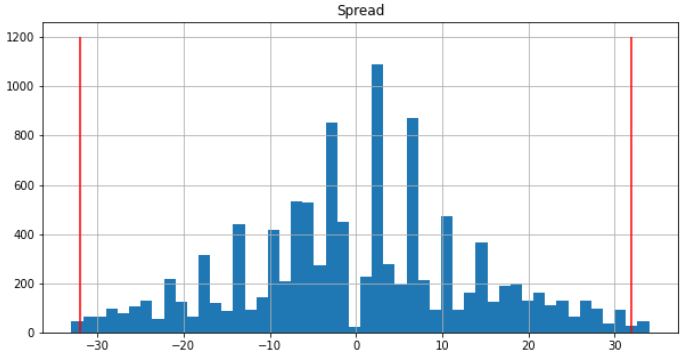
3. Data Cleaning & Exploratory Data Analysis

After we divided up the features we determined could explain the variance in NFL game spread, we pushed our combined data to the Github repository, and we set out on our next step of a data science project: Exploratory Data Analysis. Merging the datasets together wasn't as challenging as it could have been simply because all of our data came from the same place, and all of the features shared multiple columns. The combined dataset consisted of 64 total columns ranging from the year the game took place in, to the number of passes completed in the game, to number of sacks allowed in the game, etc. Many of the columns were redundant, like having the number of passes attempted, the number of passes completed, and then the percent of passes completed (making the previous two unnecessary). In several instances columns were included for the winning teams but not for the opponents (an analysis on the importance of rushing yards must include how many yards both the winner *and* loser ran for). Additionally, several columns were present that were objectively not helpful and could be removed to help declutter the dataframe, like the "year" column while also having the "date" column, which included year. Some other columns also needed to be transformed because they weren't in a format that was machine learning ready. For example, our dataset contained one column called "Result" which contained the result of each game in the form of a string like "W 12-9". This represents the result of each observation in the following format: "TmOutcome TmScore - OppScore". "Tm" in this dataset represents the NFL team that has it's game statistics listed first, and "Opp" represents the NFL team that has it's game statistics listed after. This is why our dataset has twice the number of rows as the number of games because the same game can be listed in an opposite order which would contain the result column as the following: "L 9-12". Another column that needed transformation is the "Home" column

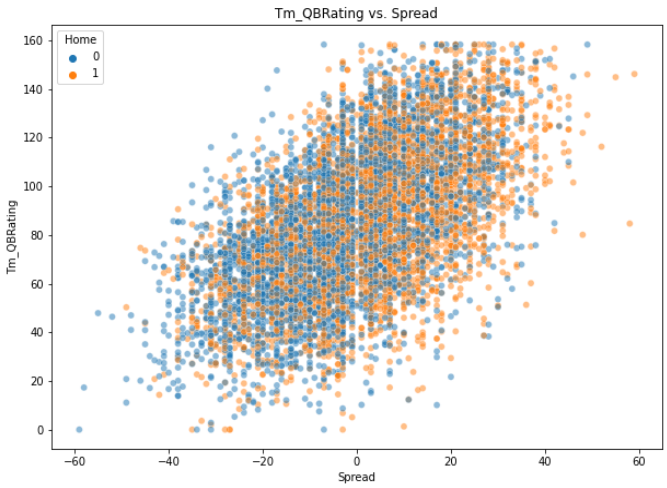


because it contained an "@" symbol in the data when the "Tm" was at the "Opp" stadium, meaning it was a home game for the "Opp". To fix this, we changed the column to a binary column representing when "Tm" was at home for the given game.

With a refined dataframe in hand we moved on to dealing with null values. A heatmap of null values, figure above, allowed us to visually identify 9 columns that had between 100 and 301 null values, with temperature having almost 2500 missing values. As 300 data points out of 11,000 is quite low, we opted to impute the missing values by randomly sampling from the respective columns. For the temperature column, we imputed the remaining null values again based on randomly sampling from the rest of the data points in the temperature column. Applying our methods for dealing with null values removed we had a final count of 11,148 data points.



Once addressing the missing values in our dataset, the last thing we did to preprocess our data was remove any outliers in our Spread column. To do this, we used a common rule of thumb which is the 1.5*IQR rule. This helped remove some bias in our model as it is very uncommon for the spread of a game to be something more than 35 points. This only removed approximately 200 data points from our dataset, therefore it won't have a significant impact on our analysis.



Next, we generated a blend of scatterplots based on the remaining features in order to do an initial visual evaluation of how the covariates relate to game spread. Most variables don't have a linear relationship when compared individually with the spread, which is why we aimed to develop a neural network that can capture nonlinearities and interactions in the data that other machine learning models would not be able to. However, the QB Rating is one variable where a reasonable positive linear relationship can be seen individually with Spread, as shown in the scatterplots to the left. We can see that the better a QB of a specific team performs, the spread tends to favor that team.

Observing this, we performed a simple linear regression to predict the Spread based on how each teams' QB performed, while including the home/away variable to normalize the comparison.

4. Baseline Ordinary Least Squares Model

To generally view the linear relationship between different stats of match and the score spread, we build a simple linear regression model. The summary of the baseline model is shown as below.

	Coefficient	Std. Err	t-stat	P > | t |	0.025	0.975
Constant	0.1734	0.081	2.134	0.033	0.014	0.333
Home	0.900	0.082	11.380	0.000	0.740	1.060
Tm_QBrating	6.426	0.090	71.684	0.000	6.251	6.602
Opp_QBrating	-7.085	0.084	-84.443	0.000	-7.249	-6.920
Tm_RshTD	3.771	0.082	45.710	0.000	3.609	3.933
Tm_Temperature	0.081	0.082	0.999	0.318	-0.078	0.241
Tm_Pass1stD	0.226	0.091	2.489	0.013	0.048	0.404

The ordinary least square linear regression model estimates showed us that the team quarterback rating has a strong positive linear relationship with the score difference and the home feature seems not that important. This simple model suggests that with a standard deviation increase in QB performance, it can translate to approximately 7 game points (equivalent of a touchdown and extra point), while explaining approximately 61% of the variance in spread. This model is by no means a strong predictor of spread, however it helped us gain a stronger understanding about the relationships in our data going forward. We also notice that the OLS linear regression model did not capture a strong linear relationship with Temperature and Pass1stD with Spread. We think that there are more of our features that we can not perfectly capture their relation to the score differences simply by linear regression estimates. Therefore, we decided to develop an N-N model to have more accurate predictions since it can capture the non-linear relationships and interactions in the data.

5. Multi-Layer Perceptron Neural Network Model

Since our prediction is a regression problem and our data points are all numerical values, the Multi-layer Perceptron regressor was a good choice and the sci-kit learn package meets the requirements we need to develop this. We definitely experienced a tough time at first when trying to find the best hyperparameter for our Neural network model, which included the activation function, hidden layer size, number of neurons, and number of epochs.

5.1 Feature Selection

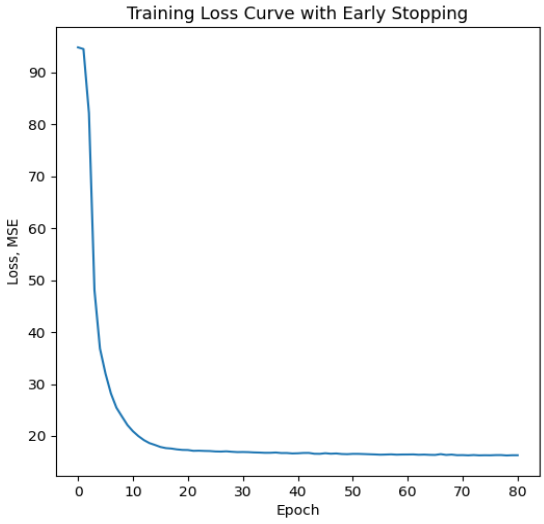
At first, we had 34 features as input for a MLP regressor model using relu as activation function , but unfortunately the performance was bad. Even though the training error and test error was low, the real prediction for 2022 Super Bowl Prediction, as well as our validation set predictions, were not even close. We collected the features of the 2022 Super Bowl match and the prediction value varied a lot with even negative spreads which is completely opposite to the real result: Los Angeles Rams defeated the Cincinnati Bengals in Super Bowl 2022 with the score 23-20. Our model at this phase was not robust at all, we kept getting results like +80. +120, -34, -6 which vary from the actual +3 a lot. Before tuning hyperparameters, we did feature selection first to improve the performance. We subtracted "Tm" stats from "Opp" stats and created a new set of features. This brought us down to 16 features. With this reduction in features the prediction was a little more reasonable as the real predictions ranged from -5 to +60. Additionally, the testing error (MAE) also reduced from 6 to 4, but it was still not robust enough, so we needed to tune different hyperparameters.

5.2 Hyperparameter Tuning

Below is a table that briefly summarizes some of the hyperparameter combinations that we tried.

Experiment	Activation function	Hidden-layer size	Neurons size combination	Regularization parameter	Epochs	Train loss (MSE)	Test loss (MAE)	2022 Super Bowl Prediction (real-spread is 3)
1	relu	5	20,20,20,20,20	0.0001	200	16.425	4.443	56.24
2	relu	5	34,34,34,34,34	0.0001	200	15.454	4.441	-2.3
3	sigmoid	4	32,64,64,128	0.001	200	16.434	4.338	2.8
4	sigmoid	5	16,32,64,128,256	0.001	200	16.469	4.387	6.3
5	sigmoid	7	32,32,64,64,128, 128,256	0.001	200	16.617	4.456	10

As you may have noticed, the sigmoid activation function is better for our NFL predictions and since we have around 10,000 data points, the hidden-layer size shouldn't be large, to avoid overfitting. Experiment 5 actually resulted in higher errors compared to others with smaller hider layer size. We also surprisingly found that 2^n neurons in each hidden layer performs much better than other random numbers. After trying out tons of hyperparameter, we finalized our MLP-Regressor model with 4 hidden layers and a neuron size combination of (32, 64, 64, 128); The maximum epochs (how



many times each data point will be use) of the model setting is 200 since the loss curves at right showed us the training loss stopped decreasing around 80 epochs because the validation loss started to increase at that iteration. The validation data was set to 20% of the training data.

5.3 Test Robustness

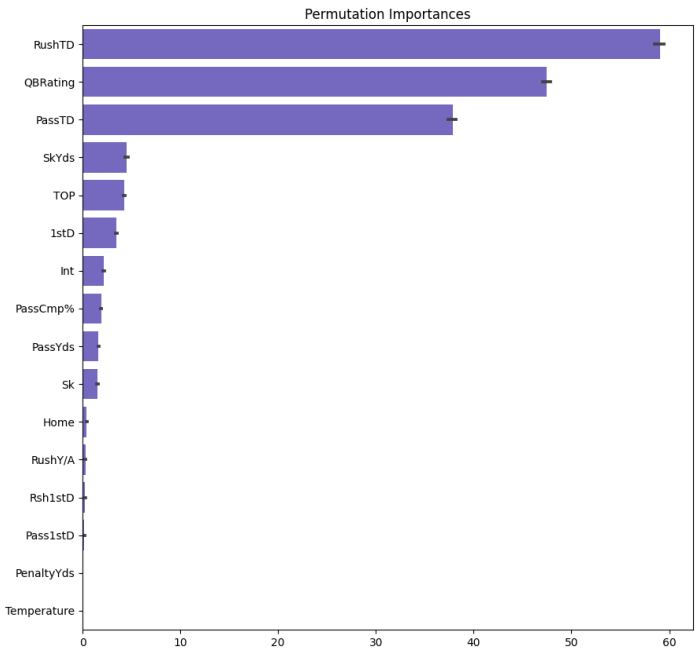
To test the robustness of our MLP model, we tried different initial values of weight and bias by changing the parameter of "random_state" in scikit-learn MLP regressor package.

Initialization	2	12	46	78	123	290	453	544	999	9999
Training loss (MSE)	16.643	16.475	16.770	16.389	16.275	16.181	16.292	16.483	16.427	16.626
Test loss (MAE)	4.421	4.375	4.398	4.354	4.388	4.373	4.396	4.369	4.403	4.389
Super Bowl LVI Prediction	2.358	4.886	9.394	1.952	6.945	4.464	3.007	3.218	3.170	4.975

After trying out different initial values of weight and bias we can conclude that our model is robust since the training error, test error, and prediction for the super bowl this year did not have large variance. In the real world, our averaged prediction among these 10 different initializations is 4.437 which is very close to the real value 3. It is also surprising that the prediction on real games is extremely accurate when we choose initialization from a (400-1000).

5.4 Permutation Importance

With this final model, we started to inspect the importance of all features by performing a permutation importance analysis on this Neural-Net Model. This analysis measures the decrease in model performance when shuffling an individual column. The greater loss in model performance means the feature is more important. This randomly shuffled procedure breaks the relationship between the feature and the predicted value, therefore the drop in performance is indicative of how much the model depends on the feature. According to our permutation graph, RushTD , QBRating, and PassTD are the three most important features. It also agrees with what OLS



linear regression estimates' results that Temperature, Pass1stD, etc has little explanation of Spread. We should always keep in mind that permutation importances does not reflect the intrinsic predictive value of a feature by itself but how important this feature is for this particular MLP regressor model. Although we have shown that this model performs extremely well, we still want to do further research like correcting the statistical inference on the relationship between the selected features and the game spread.

6. Post Prediction Inference Correction

The permutation graph shows us the relative importance for the prediction model of each covariate in predicting the game spread, and we must apply postpi to every single feature in order to see what degree (if any) postpi will assist in correcting statistical inference. Will it provide large corrections for our two most important features (QBRating and RushTD)? Perhaps only the weaker features like 1stD or Sk will benefit from postpi? There's even the chance that no features will derive a benefit from postpi, due to any number of reasons. These questions guided our analysis in our quest to revolutionize sports analysis.

6.1 Post Prediction Inference Definition

With a successful and sufficient prediction model now completed, it was time to begin the main focus of our methodology of postpi, but first some background as to what postpi actually is. Post-prediction inference is a method for providing corrected statistical inference by correcting model bias and improving variance estimations. Statistical inference (and the reliability of the inference) refers to the theory, methods, and practice of forming judgments about the parameters of a population and the reliability of statistical relationships, typically on the basis of random sampling. An example to understand this in the concept of NFL sports analysis is a hypothetical inference based on the importance of the home-field advantage. A sports analyst takes a random sample of NFL games and finds statistically significant results that a team playing on its home turf has around a 10-15% higher chance of winning. Because of the analyst's strong findings, they can provide reliable statistical inference across all NFL games (future or otherwise) that a team playing on their home turf has around a 10-15% higher chance of winning. Another important aspect of postpi is model bias, and it refers to the difference between average prediction and the correct observation the model is attempting to predict, high bias is due to an under-fitted prediction model and leads to high training/testing error. Variance estimation relates to a model's ability to predict on testing and unseen data, high variance is due to over-fitting on training data and leads to high test error.

6.2 Post Prediction Inference Method

With some background knowledge about postpi, it's time to dive into its actual steps and implementation. The first two steps are splitting the available data into train, test, and validation sets. A critical assumption of postpi is that the training and testing sets are complete and provided with both covariates and the observed outcome, while the validation set will solely contain covariates and no observed outcomes. The reasoning behind this distinction sources from another assumption of postpi, that it's used in the context that

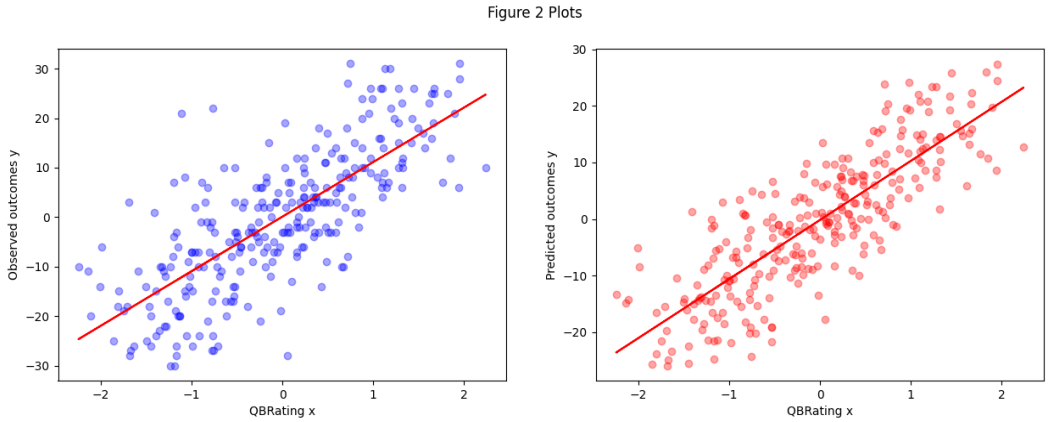
acquiring true observations for all (or perhaps any) of the available data would be infeasible. As such, the inference model must be prepared for downstream applications in which there are literally no observed outcomes collected for any of the covariates. NFL season matchups are prepared long in advance with each teams' stats widely known, it's impossible to collect the observations for future datasets unless you wait until the game has passed, and thus our subject of interest meets this postpi assumption well. After the datasets are properly set up, we fit our MLP NN prediction model on the training sets X_i covariates (ranging from quarterback rating to penalty yards) and y_i, observed game-spread outcomes, to extract a relationship between the covariates and their outcomes. The assumptions of postpi don't specify what type of machine learning model/algorithm can and can't be used as the prediction model, and to the contrary one of the greatest attributes of postpi is that it excels easily with nearly any model from regression to potentially million parameter algorithms such as K-Nearest Neighbors and Neural Networks. Once the prediction model is properly trained and ready for utilization, we use the testing set's X_i covariates to generate a set of predicted outcomes y_{pi} equal in size to the test's observed outcomes set.

The critical inquiry that postpi is designed to address is how to reduce incorrect model bias and variance so that downstream statistical inference provides more accurate results. Attempting to accomplish this by directly editing the prediction model itself is more achievable when using a simple regression model, but doing so on a high level model like our Neural Network would be nearly impossible. Postpi is useful because it doesn't operate on the prediction model, an essential part of its inference correction is based on a low-dimensional relationship model (in our case linear regression) generated between the testing set's observed and predicted outcomes. In the validation set, the prediction model generates a set of predicted outcomes y_{pi} on the validation set's covariate sample. A refined set of validation outcomes with reduced bias and better variance estimation is produced by plugging the predicted outcomes into the relationship model to reverse engineer "observed outcomes", or basically predicted outcomes that are much more similar to what actual true outcomes would look like. A potential limitation of postpi is that a critical factor in its successful implementation depends on the caliber of the relationship model between the testing tests y_i observed outcomes and y_{pi} predicted outcomes, yet this wasn't an issue with our data.

The relationship model provides a solid means of inference correction, but another and more advanced method of correction is the bootstrap based approach, and in our code the bootstrap function runs 100 times for each postpi iteration. This approach takes place after the relationship model has been implemented, and then the relationship model, the validation data featuring only the covariate of interest, and the uncorrected predicted outcomes, are passed into the bootstrap function. Random sample with replacement is taken from the validation data (covariate and predictions) equal in size to the validation data. Corrected predictions are generated using the relationship model and we produce an OLS regression model as our inference model. From this inference model, we extract the beta estimate, the standard error of the beta estimate, t-statistic, and the p-value. For

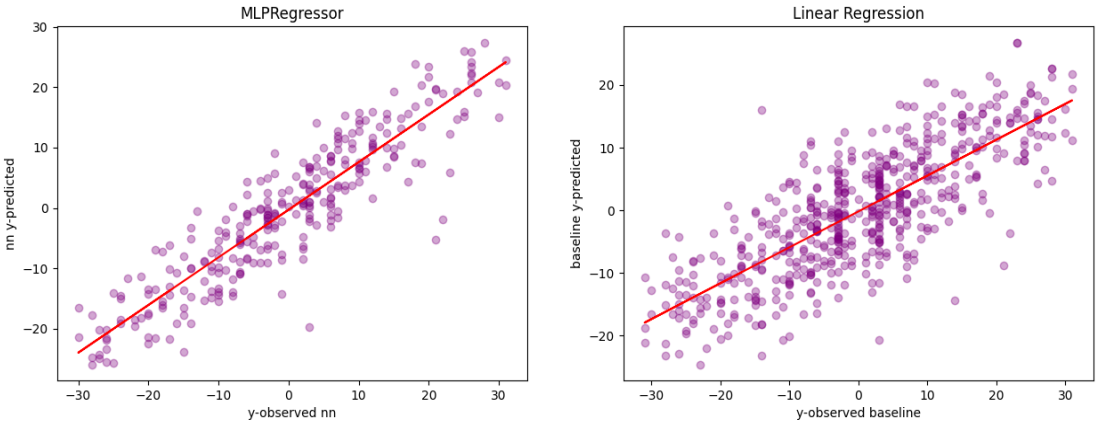
parametric bootstrap we return to the postpi function the median of the 100 beta estimates, the median of the 100 beta estimate standard errors, and the median of the p-value of the beta estimators. In the case of non-parametric bootstrap, the median of the beta estimates and the median of the p-values are also returned, however the standard error is calculated as the standard deviation of the 100 beta estimate standard errors. The t-statistic is calculated back in the postpi function by dividing the aggregated beta estimate with the standard error, and the final values for parametric and non-parametric bootstrap-based corrected beta estimates, standard errors, p-values, and t-statistics, are all appended to respective lists at the conclusion of each iteration of postpi.

6.3 Post Prediction Inference Application



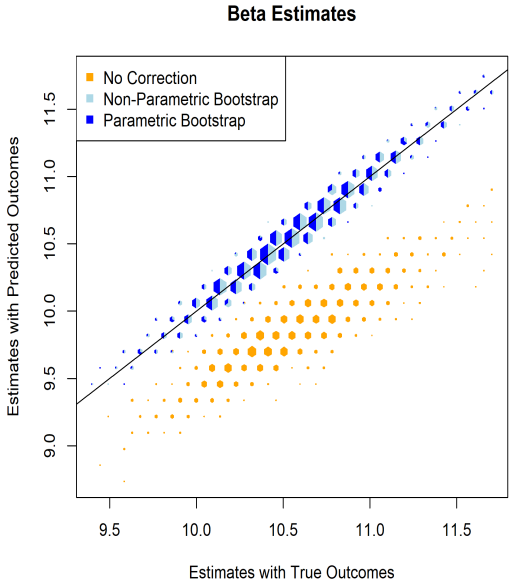
Our findings illustrate this process and describe our conclusions. In our data we ran postpi 1,000 times on each of our features separately as the covariate of interest, and chose to highlight our results using the feature (QBRating) that most strongly emphasized the results we found across the board from all covariates in our dataset.

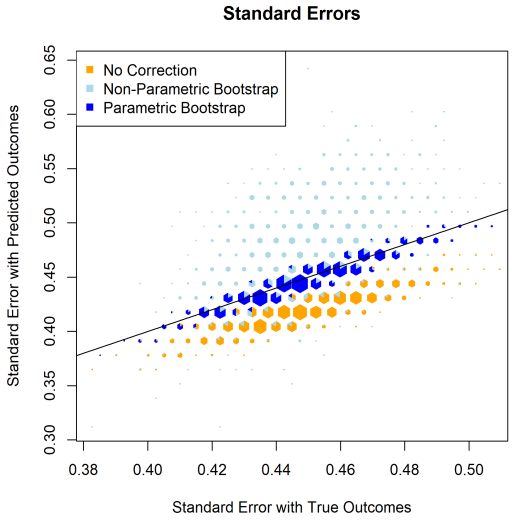
In the figure above, the left plot shows QBRating versus the observed game spread, and the plot to the right shows QBRating versus the predicted game spread. Something that instantly strikes out is how the predicted outcomes have lower variance and hug closer to the red line when compared to the observed outcomes. Another point of interest is that the predicted graph's overall shape and linear angle is highly similar to the observed outcomes' graph. The following figures are based on the test set's data.

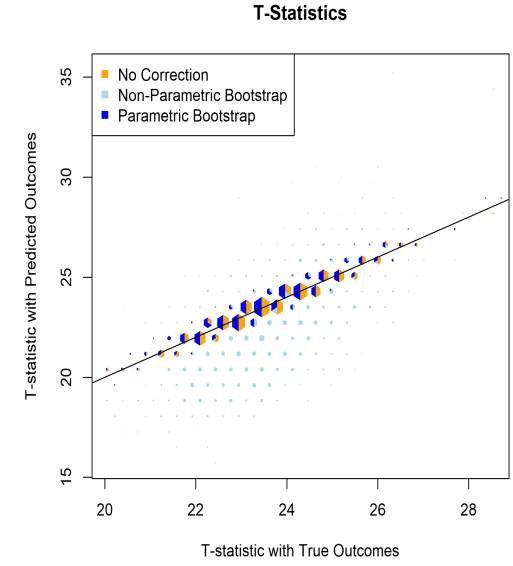


Both graphs above display the relationship between the test set's observed and predicted outcomes, with the graph on the left featuring the predictions from our MLP Neural Network model, and the graph on the right displaying the predictions from our linear regression

baseline model. We previously mentioned that the ability to easily model the relationship between predicted and observed outcomes despite the complexity of the prediction model was an essential component of postpi, and we can see that both our neural network and linear regression highlight a strong linear relationship between their respective observed and predicted outcomes. Another striking detail is that our prediction model has a far stronger relationship between the observed and predicted values than the baseline model does, simply due to the nature of the models themselves. Having concluded that a simple relationship model between the observed and predicted outcomes can be established from our Neural Network, we can derive that the grounds for inference correction via the relationship model is plausible and move on to bootstrap-based correction.

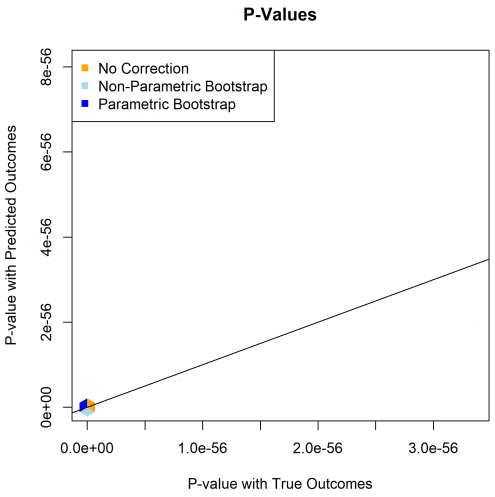






The graphs to the left are based on the validation set's data. There are three potential labels as can be seen in all of these graphs, non-parametric, parametric, and no correction, with each graph containing 1,000 data points for each label corresponding to the number of postpi iterations we ran. The two former have been earlier explained as the two types of bootstrap-based correction, but no correction is the control sample of predicted outcomes which received no correction (via bootstrap nor the relationship model), and serves to let us observe how the inference model fitted on predictions directly from our MLP Neural Network contrast to the inference models rendered via postpi's bootstrap based correction. No correction on the top plot is consistently lower than either the non-parametric or parametric bootstrap, with the latter two seeming to show nearly identical results, however they all share a very strong linear relationship. On the Standard Error plot, non-parametric standard error is all over the place, parametric is almost perfectly on the line, and no correction is now only slightly lower than the line and the values are much tighter. Again as with the beta estimates, no correction displays a strong linear relationship and consistent slope. Here's where the findings become quite clear, on the "T-Statistics" graph because the parametric and no correction t-statistics are both perfectly on the line with no major differences, again the

non-parametric values are more widely distributed. On the "P-Values" figure below, all three labels feature a p-value of approximately zero, as this is less than 0.05 we can reject the null hypothesis that the respective inference model does not accurately capture the relationship between the covariate of interest (QBRating) and the predicted outcome (game spread), and decisively conclude that each of the inference models for no correction, non-parametric bootstrap, and parametric bootstrap all reliably represent that relationship. The "P-Values" graph for all of the other features as well looked identical or very similar to this one, and as such they provide a clear answer to the question "to what degree, if any, will postpi provide statistical inference correction for our NFL sports analysis?" The answer to that being "not much", apparently. While the exact reasoning as to why postpi didn't help out much here can't be confidently stated, we can speculate that the very high accuracy of our MLP Neural Network prediction model effectively expressed the link between each of our covariates and the game spread outcome.



7. Conclusion

In conclusion, our project followed all the steps of a data science project. We began with collecting our NFL game data and compiled a final dataset by using all the different smaller datasets we collected. Then, we conducted exploratory data analysis to understand the relationships between variables in our data, and dealt with common pre-processing steps such as missing values and outliers. Once we had a strong understanding of our cleaned dataset, we built a simple linear regression model as a baseline to improve on to gain some understanding on how our covariates impact our response variable, Spread. Once we saw that our baseline model is not capturing all the relationships in our data, we developed a neural network model that performed much better than our baseline model. With this model we were able to achieve a near perfect prediction on Superbowl LVI, and an average error of 4 points, nearly half of our initial baseline model. We then set out to understand the importance of each of our features so that we can apply our research of post-prediction inference to better correct our inference on these variables. After using the method of postpi, we find that our neural network model results in a very accurate representation of our covariates and therefore statistical inference correction is not necessary in this case. Our post-prediction inference methods helped confirm that our model is quite accurate in inferring the relationship for each of our features that is used to determine the outcome of an NFL game.