# Capstone 1: Machine Learning

Now that we've used inferential techniques to explore our data further, we can proceed with the model building process. Due to the nature of our problem, we will be using supervised learning for numeric predictions. More specifically, we will be building regression models to predict points scored for away teams and home teams. After we have predicted the number of points scored for away teams and home teams, we will combine the predicted points scored for both teams to get a prediction for the total number of points scored in every game. This total can be compared to the total that was put out by the sports book, and we can make a decision to take the over or under based on the comparison.

We start with our dataset that was prepared in the last few sections, which we split into a dataset for away teams and home teams. In both tables, the variables that will be considered in our models are stDvoa, runMatchup, ptsMatchup, offDvoaMatchup, offMatchup, ovrMatchup, passMatchup, surface, pblkMatchup, roof, and totalDvoaMatchup. For the roof and surface variables, we transform them into binary variables, since each of them only has two possible classes (roof: dome or outdoors, surface: grass or turf). Now that we have all variables into number form, we proceed by splitting the datasets into train/test data. We use 80% of each dataset as training data, and 20% for testing.

For our first model, we use linear regression from Python's sklearn library. We fit the training data to the training target variable, points scored, for both away and home teams. We then cross validate our training data to ensure that we are not overfitting. For linear regression, we use R-squared score as our metric for cross-validation. The average score across our ten folds of data is roughly 0.11, and the R-squared range for the ten folds is from 0.05-0.18.

Next, we set up our model so that we can begin to eliminate variables that aren't significant in predicting our target. We use the backwards stepwise regression strategy to eliminate variables. We proceed by fitting the model with all variables, then eliminating the variable with the highest p-value, as long as the p-value is over a certain threshold. In this case, we use alpha = 0.05 as the threshold. For the away table, the variables that made it to the final model are ptsMatchup, offDvoaMatchup, passMatchup, and totalDvoaMatchup. The adjusted R-squared for this model is roughly 0.114, and the equation for the model is as follows:

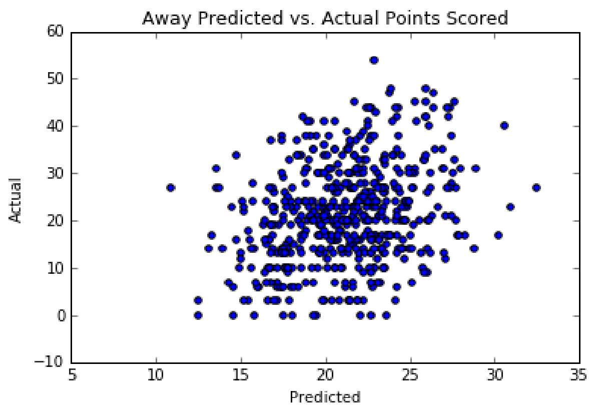
$$y_{away} = 0.195x_{ptsMatchup} + 2.566x_{offDvoaMatchup} + 0.099x_{passMatchup} + 4.957x_{totalDvoaMatchup} + 12.042$$

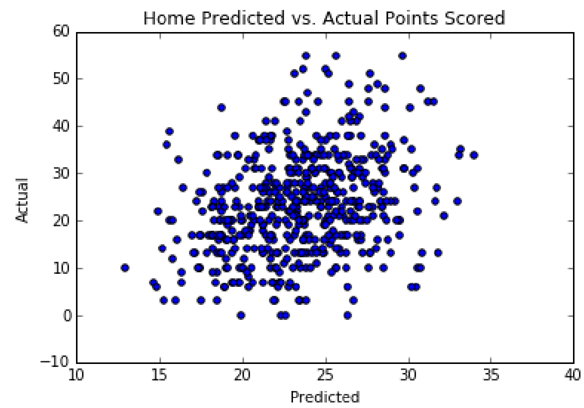
For the home table, the variables that made it to the final model are ptsMatchup, offDvoaMatchup, pblkMatchup, roof, and totalDvoaMatchup. The adjusted R-squared for this model is roughly 0.121, and the equation for the model is as follows:

$$y_{home} = 0.279x_{ptsMatchup} + 5.944x_{offDvoaMatchup} + 0.071x_{pblkMatchup} - 1.61_{roof} + 4.805x_{totalDvoaMatchup} + 11.919$$

Now that we have our equation for both away and home teams, we can make predictions on our test data, check our model assumptions, and evaluate our models.

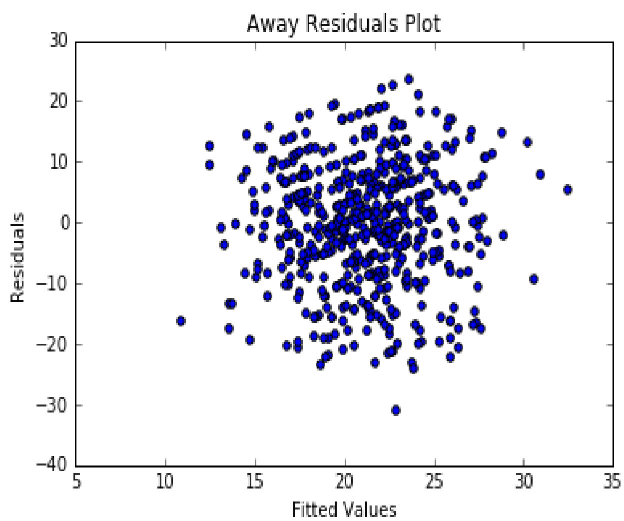
First, we plot our predicted vs actual for both models. We hope to observe a positive correlation so that we know our models are predicting with some degree of accuracy.

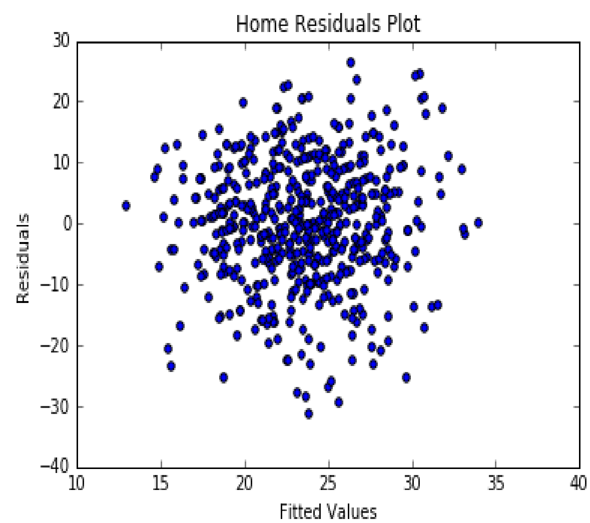





We can see a positive correlation for both the away and home models. For the away model, the pearson correlation is roughly 0.307, and for the home model, it is approximately 0.306.

Next, we plot the fitted vs residuals for both models to ensure there are no patterns with the residuals, heteroskedasticity, etc.






As we can see, there are no violations with our residuals plots. Observing the histograms of the residuals in our jupyter notebook, we conclude that both models have approximately normal residuals as well. From observing the scatterplot matrices in the last section, we also know that we have no multicollinearity issues for our model. Thus, all model assumptions are valid, and we can feel confident about deploying these models.

We next merge our predictions for away and home teams together based on gameId, so that we may add the two scores together to get a total to compare to the total put out by the sports book. After comparing the predictions to the totals put out by the sports book, a pattern was identified that helps making predictions on the over/under more accurate. Instead of betting the over if our prediction is greater than the sports book's total, or under if our prediction is less than the number, we notice that there is a huge edge when only betting games in which the prediction exceeds the sports book's total by five or more points. We call this the "5-point Overs Rule". We also test for a 3 and 1-point Overs rule, and do the same for unders. To gather results for our system, we train  21 random seeds of our dataset, and test on each seed's respective test set using the exact same variables for both the away and home teams. It was found that using the 5-point overs rule, our average win percent was 60.8%. This exceeds the required 52.4% needed to be long-term profitable when betting on NFL games. This would be good for a 8.4% return on investment if one were to follow this system's recommended plays. The results for the 3-point rule also proved to be profitable, but to a lesser extent. An average win rate of 57.2% was found using this rule, although there were nearly triple the amount of plays recommended using this system.

Using the 5-point overs rule, clients can expect to receive approximately 19 recommendations per season, or around 1 recommendation per week. Using the 3-point overs rule, clients can expect to receive approximately 52 recommendations per season, or around 3 recommendations per week. The results for these two rules are as follows:

| OVERS - 5 pt. rule | | | |
|---|---|---|---|
| RS | W | L | pct |
| 1020 | 21 | 19 | 0.525 |
| 2146 | 20 | 13 | 0.60606 |
| 710 | 28 | 20 | 0.58333 |
| 2482 | 20 | 16 | 0.55556 |
| 2107 | 25 | 6 | 0.80645 |
| 2601 | 25 | 5 | 0.83333 |
| 1876 | 20 | 18 | 0.52632 |
| 2721 | 26 | 14 | 0.65 |
| 1493 | 23 | 22 | 0.51111 |
| 827 | 22 | 19 | 0.53659 |
| 1958 | 19 | 15 | 0.55882 |
| 2182 | 26 | 9 | 0.74286 |
| 611 | 24 | 19 | 0.55814 |
| 2500 | 26 | 26 | 0.5 |
| 2723 | 21 | 12 | 0.63636 |
| 166 | 24 | 19 | 0.55814 |
| 74 | 24 | 12 | 0.66667 |
| 1172 | 27 | 13 | 0.675 |
| 863 | 27 | 11 | 0.71053 |
| 2790 | 18 | 14 | 0.5625 |
| 155 | 28 | 16 | 0.63636 |
| **Total** | **494** | **318** | |
| **pct** | **0.60837** | | |

| OVERS - 3 pt. rule | | | |
|---|---|---|---|
| RS | W | L | pct |
| 1020 | 61 | 58 | 0.51261 |
| 2146 | 55 | 41 | 0.57292 |
| 710 | 68 | 54 | 0.55738 |
| 2482 | 61 | 44 | 0.58095 |
| 2107 | 57 | 38 | 0.6 |
| 2601 | 57 | 30 | 0.65517 |
| 1876 | 52 | 42 | 0.55319 |
| 2721 | 59 | 47 | 0.5566 |
| 1493 | 67 | 50 | 0.57265 |
| 827 | 55 | 59 | 0.48246 |
| 1958 | 56 | 56 | 0.5 |
| 2182 | 71 | 50 | 0.58678 |
| 611 | 66 | 46 | 0.58929 |
| 2500 | 61 | 55 | 0.52586 |
| 2723 | 59 | 38 | 0.60825 |
| 166 | 58 | 43 | 0.57426 |
| 74 | 60 | 40 | 0.6 |
| 1172 | 62 | 41 | 0.60194 |
| 863 | 65 | 39 | 0.625 |
| 2790 | 55 | 37 | 0.59783 |
| 155 | 65 | 42 | 0.60748 |
| **Total** | **1270** | **950** | |
| **pct** | **0.57207** | | |

In the tables above, RS is the random seed of data we are testing on, W and L are the number of wins and losses that the random seed of data observed using this system, respectively. Each seed's win percent is shown on the right, and the total win percent is shown at the bottom of each table.

We can see that using the 5-point overs rule, we get only two of twenty-one random seeds of data that do not exceed the 52.4% win percentage mark needed to be profitable. Using the 3-point rule, we get three of twenty-one random seeds of data that aren't profitable. One could be confident that using our system will give them a very good chance of seeing a positive return on their investment.

We repeat this entire process of training our dataset and recording our results for both Random Forest and XGBoost regressors. Neither of these two regressors were able to exceed the ROI provided by using the multiple linear regression (MLR) model, but XGBoost was able to outperform the MLR model's unders predictions using a "Between-3-and-5-point Rule". This means that when the XGBoost model predicts a total that is between three and five points less than the total put out by the sports book, we should use this model instead of our MLR model. Roughly a 2.362% ROI was discovered using this XGBoost model rule. The results table for this rule using XGBoost are below:

| UNDERS - Between 3 - 5 pt. rule | | | |
|---|---|---|---|
| RS | **W** | **L** | **pct** |
| 1020 | 31 | 18 | 0.63265 |
| 2146 | 29 | 20 | 0.59184 |
| 710 | 18 | 21 | 0.46154 |
| 2482 | 20 | 16 | 0.55556 |
| 2107 | 25 | 17 | 0.59524 |
| 2601 | 26 | 34 | 0.43333 |
| 1876 | 26 | 17 | 0.60465 |
| 2721 | 28 | 19 | 0.59574 |
| 1493 | 27 | 16 | 0.62791 |
| 827 | 28 | 28 | 0.5 |
| 1958 | 24 | 15 | 0.61538 |
| 2182 | 28 | 22 | 0.56 |
| 611 | 23 | 20 | 0.53488 |
| 2500 | 25 | 25 | 0.5 |
| 2723 | 30 | 26 | 0.53571 |
| 166 | 23 | 22 | 0.51111 |
| 74 | 22 | 19 | 0.53659 |
| 1172 | 34 | 23 | 0.59649 |
| 863 | 27 | 28 | 0.49091 |
| 2790 | 30 | 32 | 0.48387 |
| 155 | 28 | 18 | 0.6087 |
| **Total** | **552** | **456** | |
| **pct** | **0.54762** | | |