Summer Internship Report

Name: Pratik Sanghvi

UB#: 50290451

Introduction

The summer internship project revolved around one of the pain points of Delaware North, to manage the workforce efficiently. The goal was to help with work force management by predicting number of transactions that would help with managing labor efficiently.

As an intermediate step was to predict the attendance at sporting venues, which would then be used to predict transactions.

Data Extraction:

The aim was to fetch and create features that we thought would be important and relevant in making strong predictions for attendance. For this, we fetched from internal as well as external sources.

Internal sources include fetching transaction information for all sporting venues. This information was to be used in the next phase of the project to predict transactions.

I started by fetching team performance statistics for NHL. I extracted this data from an API that provides detailed information for all teams for all seasons of NHL.

The API had several different end points to extract different kinds of statistics:

The endpoint below helps in pulling season standings, total wins, total losses, goals scored, goals conceded.

https://statsapi.web.nhl.com/api/v1/standings?expand=standings.record

The endpoint below helps in pulling game by game team statistics that include number of goals scored by home and away team, shots on goal by home and away team, game type(Regular or Playoff).

https://statsapi.web.nhl.com/api/v1/schedule?startDate=2017-10-01&endDate=2018-04-30&expand=schedule.linescore

I used these endpoints to pull the data and create features for the model:

1. Rolling average statistics like average goals scored by the home and away teams
2. Rolling average shots on goal
3. Rolling counts of the number of head-to-head wins with each opponent
4. Rolling average of number of goals scored in a match

5. Rolling average goal difference for a home team
6. Categorical variable for game type, regular of playoff
7. Categorical variable indicating if the home and away team are from the same division, same conference or different conference
8. Categorical variable for month in which the game is played
9. Categorical variable for day of the week the game is played
10. Categorical variable if it is a weekend game or not
11. Categorical variable if the game is played on a holiday or not
12. Categorical variable if it is a day game of not (before 5pm)

Methodology:

Exploratory Data Analysis:

The first step involved one-hot encoding the categorical variables.

Next, I then checked for missing values for all variables. Since all the continuous variables were calculated by me, there weren't any missing values.

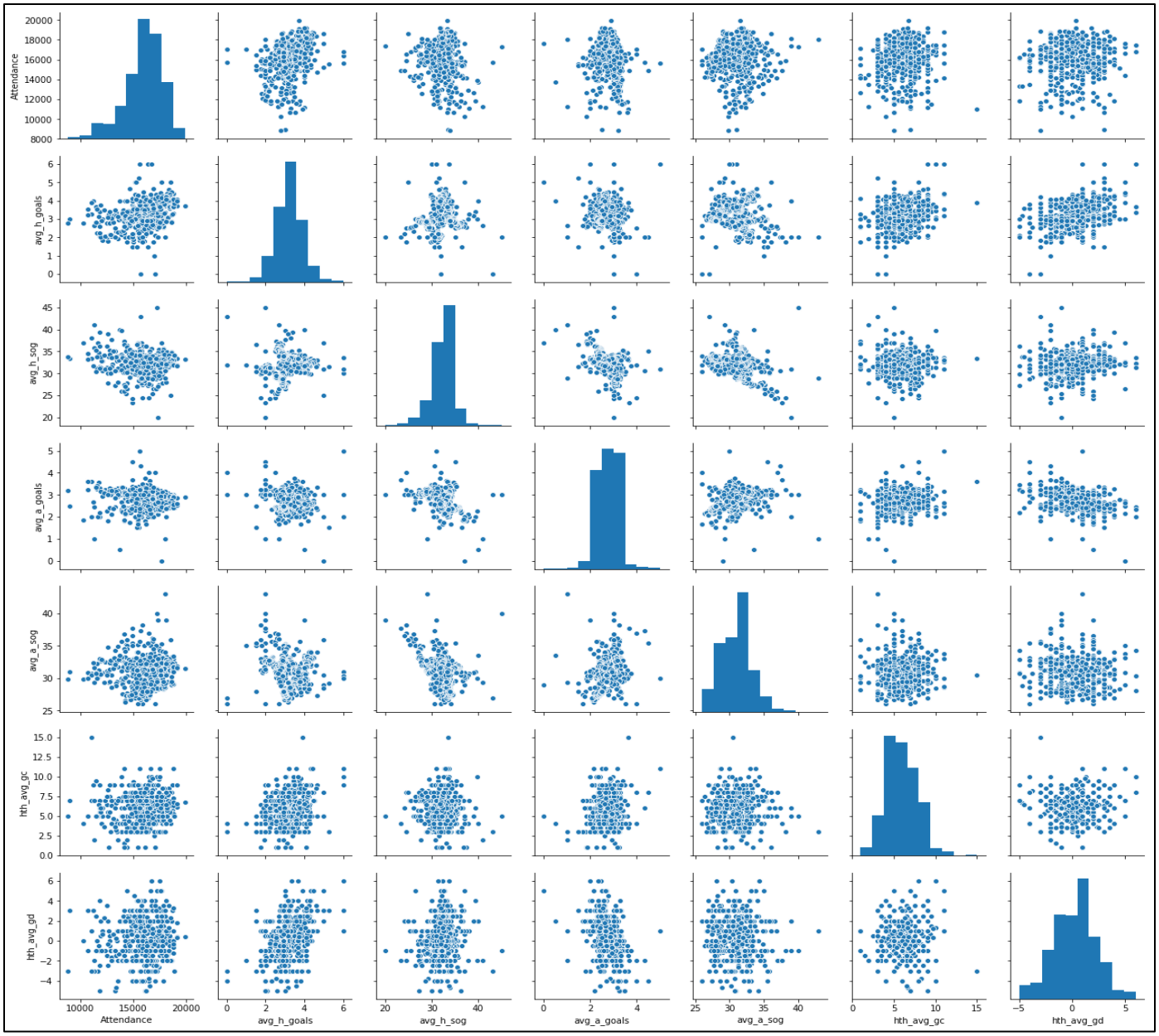
Now, we want to start by applying a linear model. For a linear regression, we have a few assumptions that need to be checked.

Linear Regression Assumptions:

1. To check for linear relationship between features and target variable. This can be checked with scatter plot
2. To check for no or little collinearity between features. This can be checked using a correlation matrix and variance inflation factor
3. Homoscedasticity

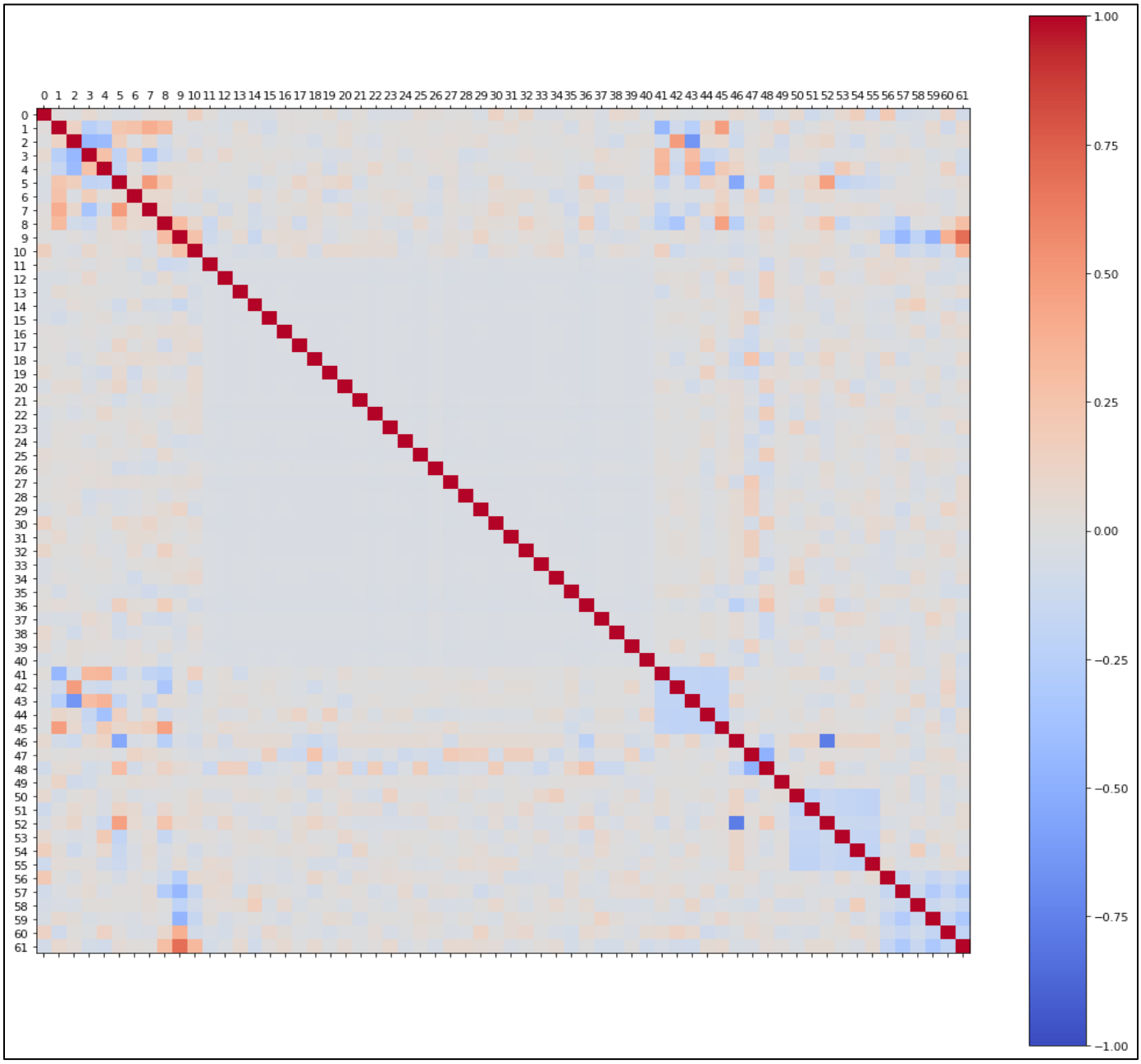
Next step was to understand the relationship between features. Understand the distribution of the independent and dependent variable.

The plot below is a scatter plot for all the features.

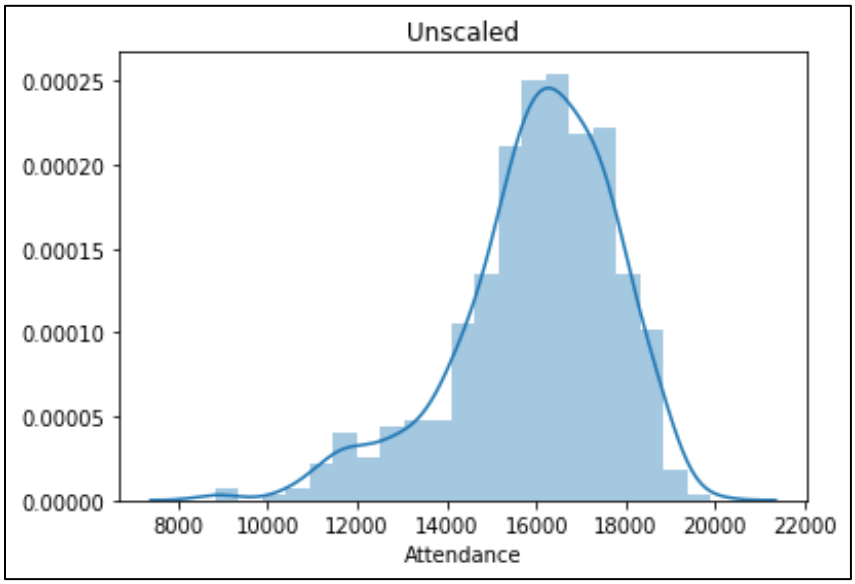


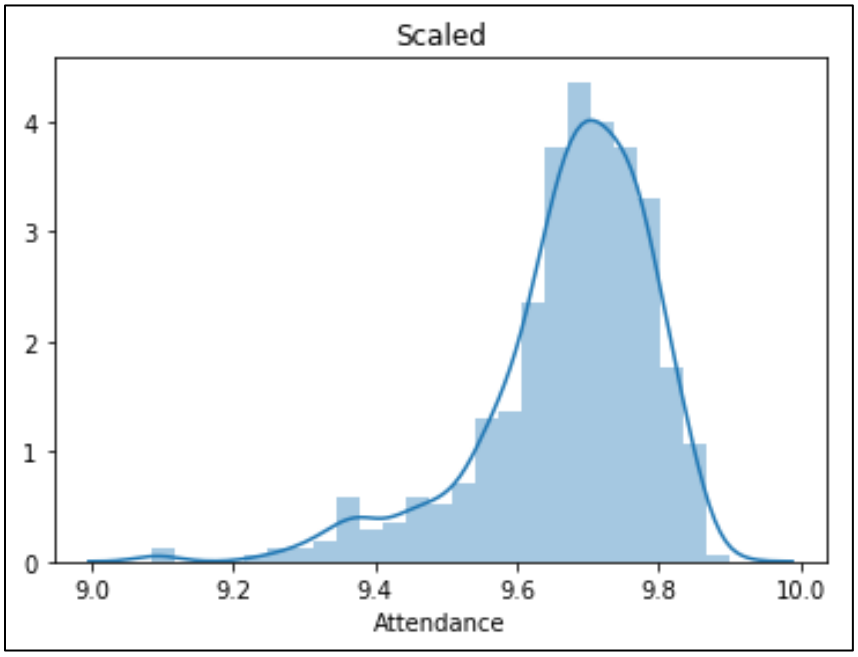
Looking at the pair plots for continuous features, we notice that we can't visually see a linear relation of all features with the attendance. This could mean that the variables don't have a linear relation with the target variable.

The plot below is the correlation matrix:



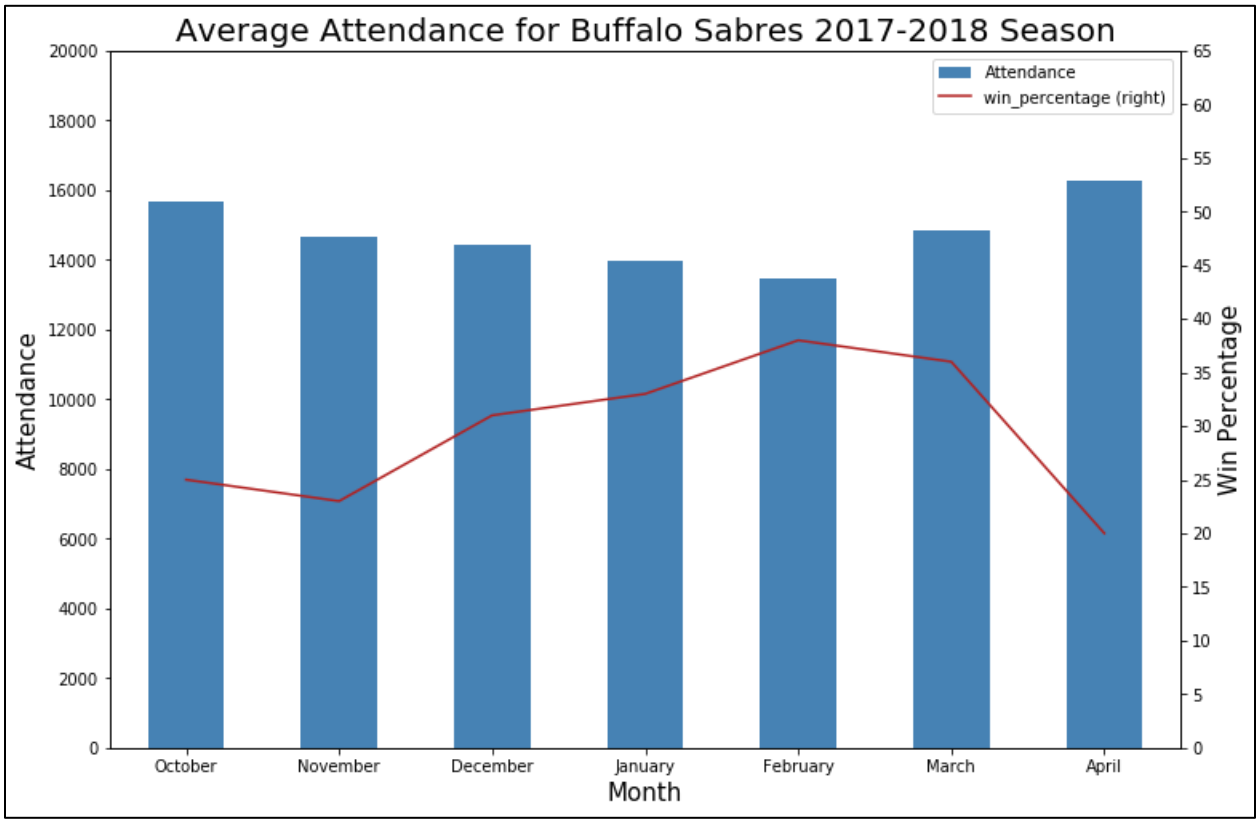
Below, we check the distribution of the dependent variable. Because the distribution of the dependent variable wasn't uniform, I tried log transformation. Below is the unscaled and scaled distribution:





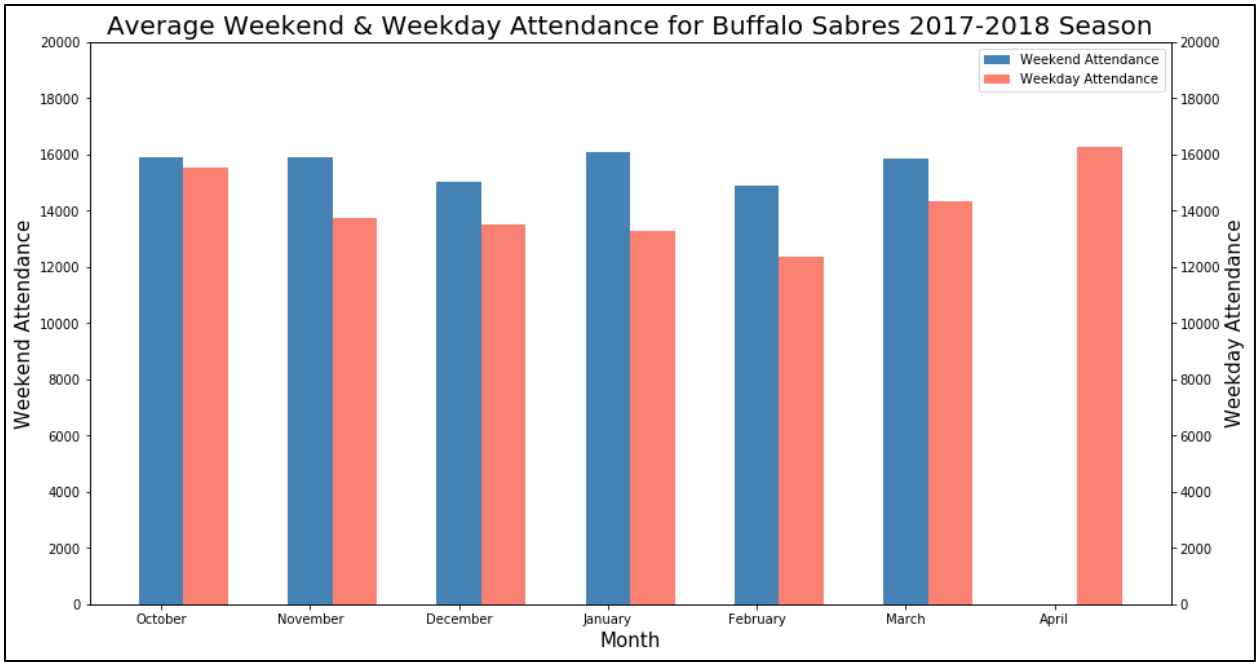
We can think of log transformation for the dependent variable and check the predictions, if it can help in improving accuracy of the model

Before going on to pre-processing, we want to see how certain features are affecting the attendance to study the trends. Below we see the average attendance by month for Buffalo Sabres for 2017-2018 season.



Looking at the graph above, we see that the average attendance dips as we move into the mid-season and again sees an upward trend as we approach the end of the season. The red line here is the win percentage. Surprisingly, win percentage and average attendance are not aligned in this case. This indicates there are some other features too that are affecting the attendance.

Now, let's look at another variable and its influence on attendance.



The blue bars are the average weekend attendance while the red bars are the average weekday attendance. A key observation that we can make is that the average weekend attendance is almost constant while the average weekend attendance suffers into the mid-season. So it is the weekday games that are affecting the attendance.

Pre-Processing:

Before we apply an algorithm to our data, it is important to scale and standardize the features. Standardizing is to subtract the mean and scale to unit variance. It is done independently on each feature; the mean and standard deviation are stored to be used later in the transform method.

Models:

We split the data with a 70:30 split for train and test respectively.

Fitting a multiple regression model on the data we get the following output:

Evaluating model performance by looking at RMSE and R squared

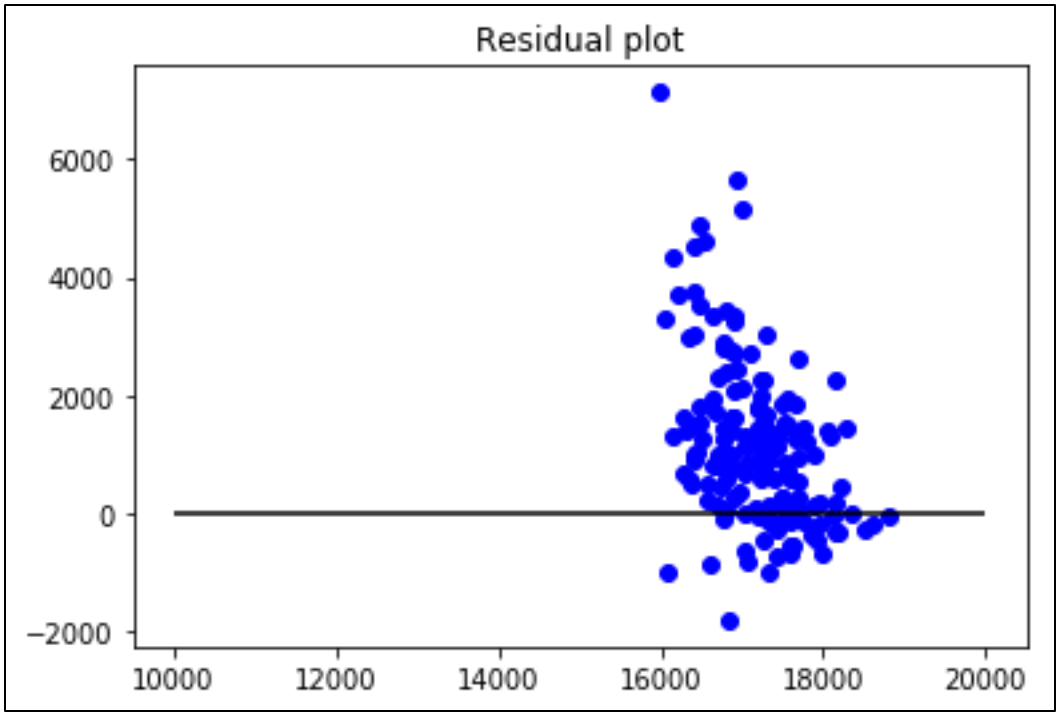
```
rmse = sqrt(mean_squared_error(Y_test, y_pred))
r2 = reg.score(X_test,Y_test)
print('Root Mean Squared Error:', rmse)
print('R Squared', r2)
```

```
Root Mean Squared Error: 1840.2106886663803
R Squared -0.12337214818749342
```

Looking at the R^2 value, indicates that may be the relationship between dependent and independent variable is not linear, we will now apply cross validation and regularizations to further deep dive.

Before applying regularizations, we look at residual plot to check for heteroscedasticity.

Below is the Residual Plot to check for homoscedasticity/ heteroscedasticity. Heteroscedasticity is all about checking for randomness in the residuals. If there is a funnel shape, or a linear relation, then heteroscedasticity exists. This indicates signs of non-linearity in the data that has not been captured. The plot below has enough randomness so we should not have much problem for heteroscedasticity.



Next, we use variance inflation factor to check for multi-collinearity. Variance inflation factor is a measure of multi-collinearity in a multiple linear regression model. It is calculated by taking the variance of given model's all of beta divide by a single beta if it were fit alone. Multi-

collinearity exists if we have variance inflation factor values of 5-10 or greater and we should consider dropping that variable.

Collinearity can only be checked for continuous variables. So, we only need to look at the vif factors for continuous variables.

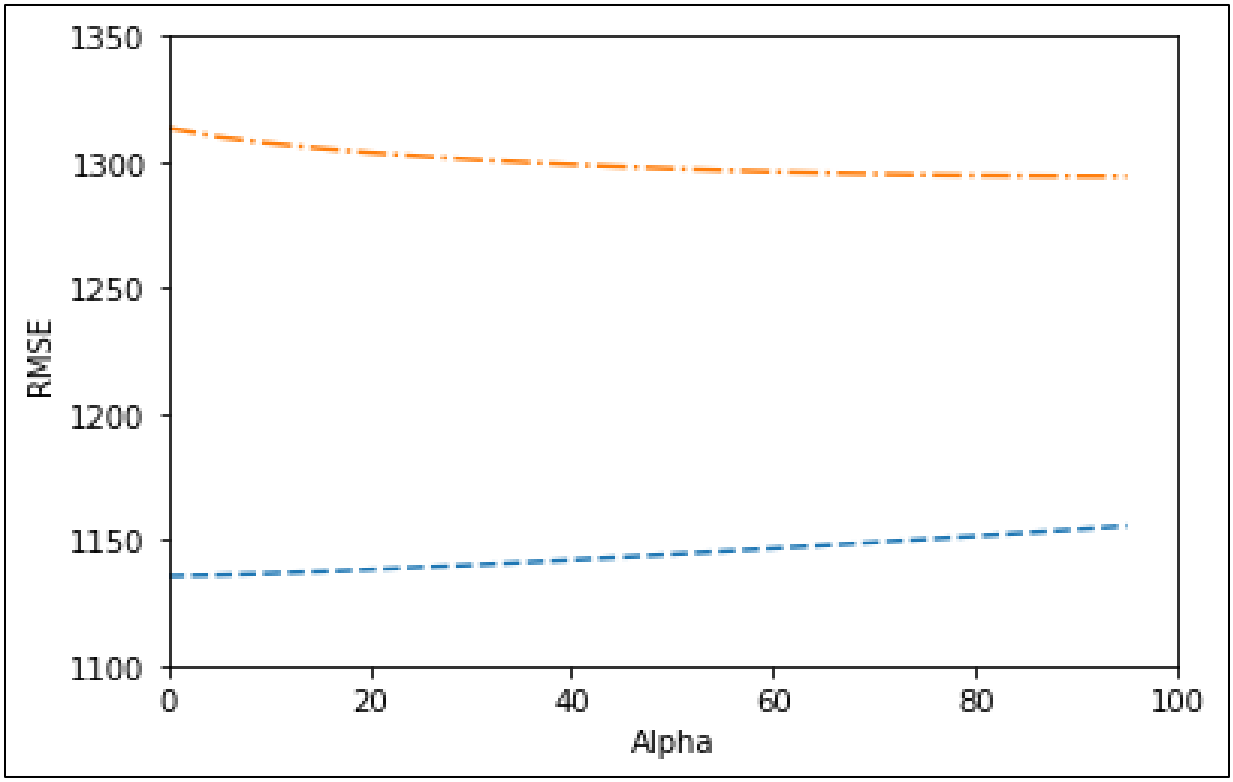
	VIF Factor	Features
0	792.800945	const
1	1.331291	avg_h_goals
2	1.434836	avg_h_sog
3	1.518339	avg_a_goals
4	1.256342	avg_a_sog
5	1.358917	hth_wins
6	1.151374	hth_avg_gc
7	1.606815	hth_avg_gd

As we can see above, there is no multi-collinearity present.

Ridge Regression and Lasso

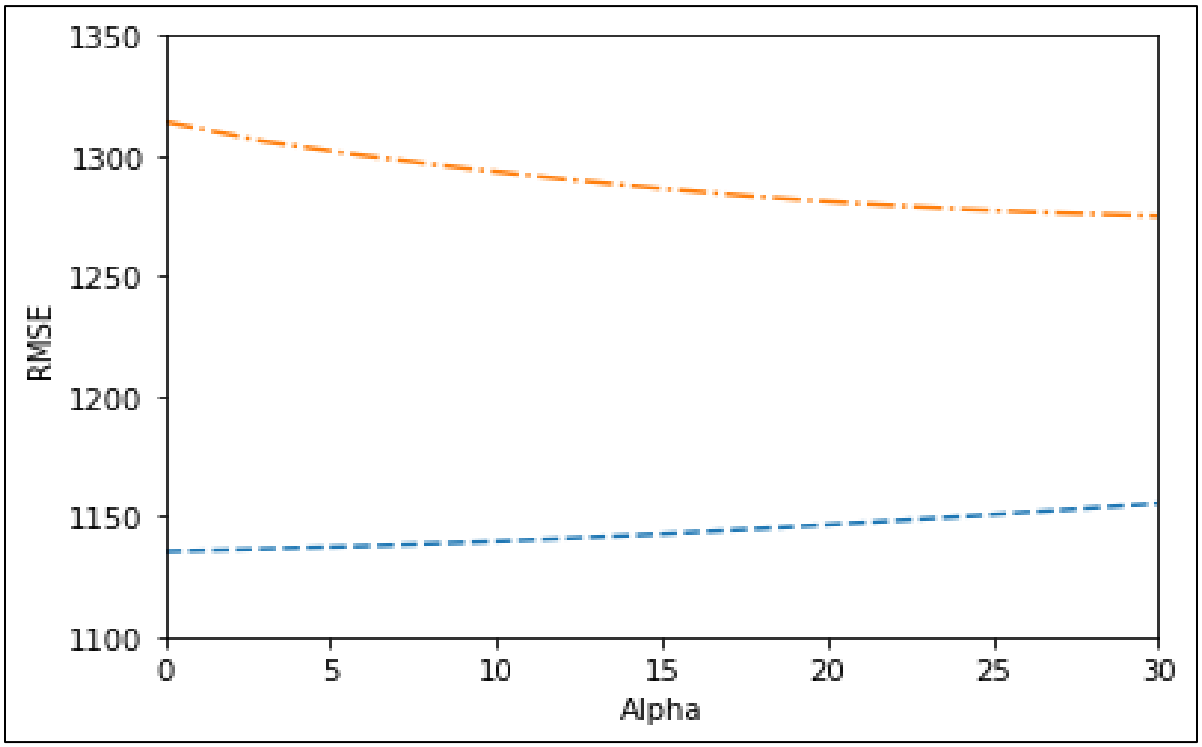
We now apply ridge and lasso for different values of alpha to select the optimum hyper parameter.

The following is the plot for root mean squared error versus alpha for ridge regression:



The blue one is for the training set and orange is for the test set. Looking at the graph above, we select an alpha value of 8 for ridge regression.

The following is the plot for root mean squared error versus alpha for lasso:



The blue one is for the training set and orange is for the test set. Looking at the graph above, we select an alpha value of 10 for lasso.

Using Pipeline:

The next step was to create a pipeline.

Pipeline helps in creating a standard sequence of steps to be applied. When we use a pipeline, we are creating a sequence in which we want our model to run. For each split, we initially apply transformation to the training set, then we fit the model on the transformed training set. The parameters for transformation (mean and standard deviation) are stored in the model. Now when we predict on the test set, we don't calculate a new set of values for transformation, that will create all together a new variable. We use the same parameters from of the train set and then transform and then make predictions

In our case we first transform our training set, then apply the model using k-fold cross validation and then predict on the test set.

We apply linear regression, ridge, lasso and the default random forest algorithms to compare their performances. Below is the result:

Algorithm	Training Set			Test Set		
	RMSE	R^2	MAPE (%)	RMSE	R^2	MAPE (%)
Linear Regression	1135.39	0.60	5.55	24338688474804.45	-1.53	19295499766.58
Ridge Regression	1136.18	0.60	5.59	1308.33	0.44	6.40
Lasso	1139.60	0.59	5.56	1293.24	0.46	6.32
Random Forest	440.45	0.94	2.13	1170.43	0.55	5.81

Looking at the results above, we see that random forest performs the best. Now, digging a bit deeper to find the best hyper parameters for tuning our random forest.

For this, we use RandomizedSearchCV, where we create a parameter grid to sample from.

We create an array for the parameters like n_estimators, max_features, max_depth, min_sample_split, min_sample_leaf, bootstrap.

During each iteration, the parameter values are selected randomly. Following is the grid:

```
{'n_estimators': [100, 311, 522, 733, 944, 1155, 1366, 1577, 1788, 2000], 'max_features': ['auto', 'sqrt'], 'max_depth':
[1, 3, 5, 7, 10, 12, 14, 16, 19, 21, 23, 26, 28, 30, 32, 35, 37, 39, 41, 44, 46, 48, 51, 53, 55, 57, 60, 62, 64, 67], 'min_
samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True]}
```

In all there are 10*2*30*3*3*1 setting and on each iteration the algorithm will choose a different combination of features.

Following are the best parameters and best model:

```
{'n_estimators': 1155, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 32, 'bootstrap': True}
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=32,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=5,
           min_weight_fraction_leaf=0.0, n_estimators=1155, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

We now use these parameter values and compare it with a base random forest model to compare the performance:

Algorithm	**Training Set**	**Test Set**
	MAPE (%)	MAPE (%)
Base Random Forest	2.23	6.16
Random Forest with best parameters	2.55	6.11

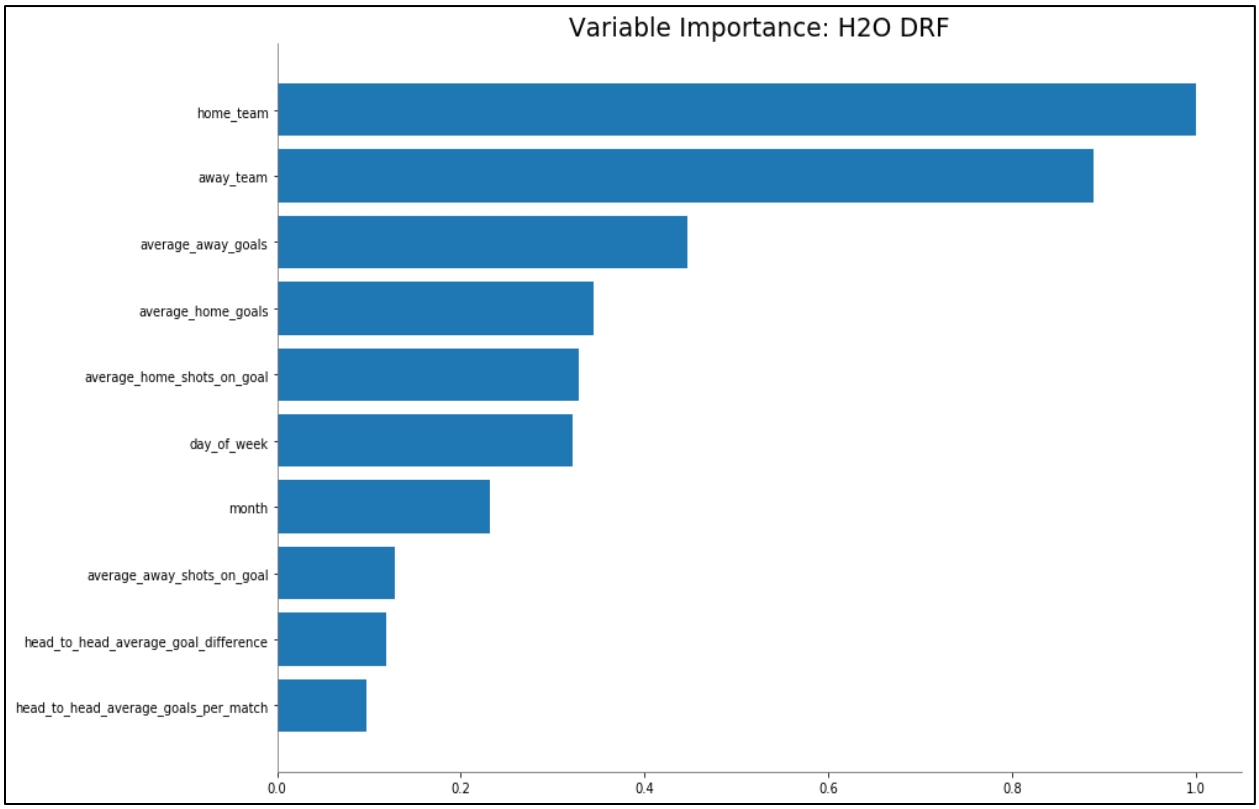
Looking at the MAPE values for the base model and the cross-validation model, we notice that the model is overfitting on the training set. This could be because of the opposition categorical variable that has been one hot encoded. That leads to an increase in features by 30, as there are 31 opposition teams

In order to reduce overfitting, we'll look at a package called H2O, random forest in H2O doesn't require one hot encoding for categorical variables and has had improved efficiencies.

Below is the performance of the model using H2O:

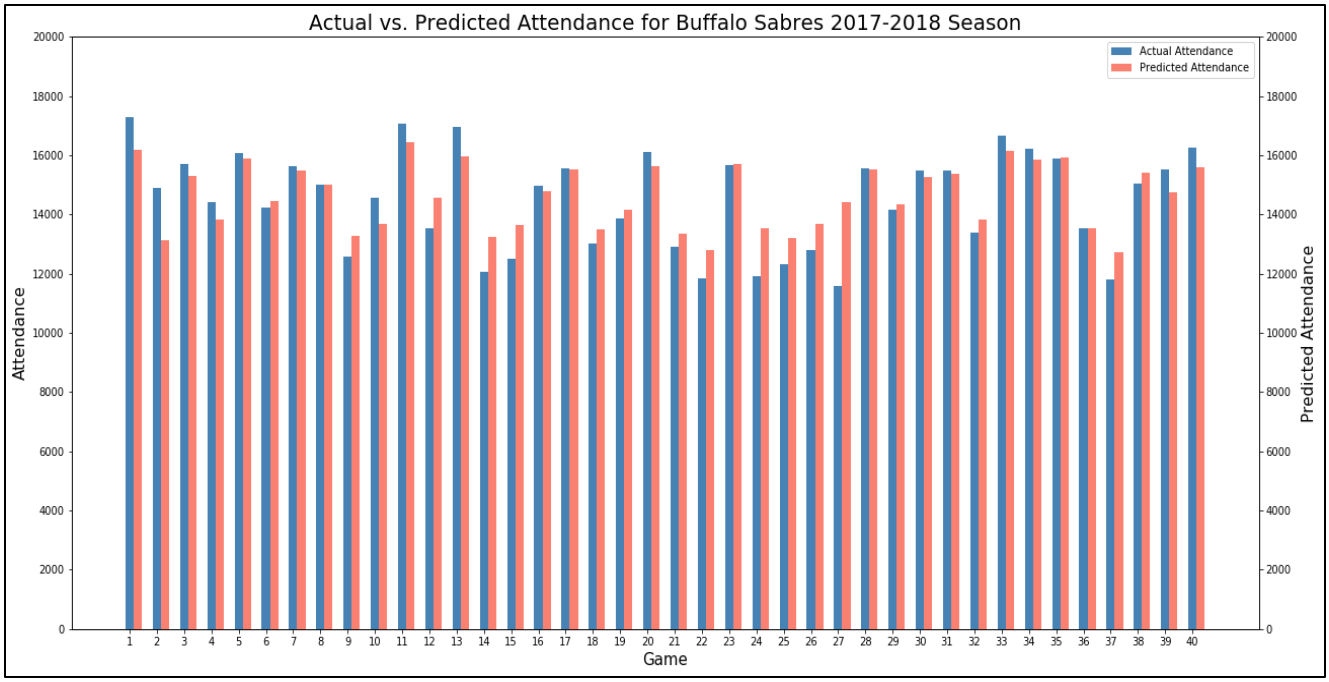
Algorithm	**Training Set**	**Test Set**
	MAPE (%)	MAPE (%)
Random Forest using H2O	2.22	5.35

Below, we see what according to the model are important variables in making prediction:



To now evaluate the performance and to put things into perspective for the higher management, we predict the attendance for the 2017-2018 season for a particular team Buffalo Sabres.

We then compare the actual attendance with the predicted attendance.



Challenges:

Extracting team performance statistics from the API was a challenge as I had to fetch information from different endpoints, bring them together and calculate the rolling aggregated statistics.

Overfitting was a big challenge. Because of one hot encoding, the model had too many features. At a point there were around 65 variables in the model due to one-hot encoding. This probably led to overfitting.

Outcome:

A lot of variables had to be dropped, but in the end, we got a model that could predict attendance with an error of just 5.2%.

These predicted values will now feed into a model to predict the transactions. The piece on predicting transactions couldn't be completed in time.

This is an important piece in several True North Initiatives which is the larger puzzle.

This is an intermediate step in several projects like predicting labor for work force management, predicting sales and subsequently helping pricing and many more.

Following are the contact details of my supervisors that helped me in this project.

Joseph Ramia, Data Scientist

Email: JRamia@delawarenorth.com

Lisa Lafferty, Data Scientist

Email: LLafferty@delawarenorth.com