Machine Learning Engineer Nanodegree

Capstone Project

David Kim
August 29th, 2017

I. Definition

Project Overview

The project's goal is to use machine learning to predict winners of NFL games against the point spread. The domain of this project is the NFL and sports betting. According to a Harris poll conducted in 2015, the NFL is America's favorite sport. 33% of survey respondents answered that the NFL is their favorite sport with based as a distant second with 15%.[1] The NFL is a professional American Football league that consists of 32 teams that was formed in 1922. Each regular season consists of 17 weeks that starts in September and ends in January. After regular season ends, there is a single elimination playoff that culminates in the finals, which is called the Super Bowl. The NFL is the most popular sport to bet on and the most popular game to bet on is the Super Bowl. For Super Bowl 51, there were $4.7 billion in bets according to the American Gaming Association.[2]

Sportsbooks are establishments that set the point spreads (or lines) and odds and take wagers from bettors. Sports betting is legal in only a few states like Nevada. [3] The sportsbooks make money by charging a percentage when a bet is won. In theory, the sportsbooks always try to set the lines and odds, so there is even money on both sides of the bet. Often times, the lines and odds change depending on betting patterns or when new information is available about the game like injuries. This ensures that there is even money on both sides of a bet. There are basic ways to place a bet:

- Moneyline – A bet on which team will win outright; however, the odds to place this bet changes depending on which team is favored. For example:
    - If moneyline for team A is +210, a $100 bet will win $210 dollars if team A wins
    - If moneyline for team B is -150, a $150 bet will win $100 if team B wins
- Point Spread (or line) – A bet on which team will win against the point spread or line. The point spread handicaps one team over another to account if the perception is that one team has a higher probability of winning. If a team has a negative spread value, the team is considered the favorite. If a team has a positive spread value, the team is considered the underdog. For example:
    - If a spread for team A is +3, a $100 bet will win $100 if team A wins or loses by less than 3 points. If team A loses by 3 points, it is considered a push.
    - If a spread for team B is -6.5, a $100 bet will win $100 if team B wins by 7 points or more. If team B wins by 6 points or less or loses, the bet is lost.
- Under/Over – A bet on the total points scored in the game. For example:
    - If the under/over is 39.5, a $100 bet will win $100 if both teams score a total of 40 points or more; otherwise, the bet is lost.

Note #1: Keep in mind if a bet is won that the sportsbook takes a percentage of the bet amount when you collect.

Note #2: There are instances where odds will be placed on a point spread bet; which is a practice done by sportsbooks to try to get even money on both sides of the bet without changing the point spread.

With my growing interest in football from playing fantasy football with friends and recently taking various classes in data analysis and machine learning, I wanted to find out if there was a way to systematically combine NFL statistics and betting lines to use machine learning to predict the winners with better accuracy than a baseline or chance. It is difficult to predict the winner against the spread since anything can happen during a game such as turnovers, in-game injuries, dropped balls, bad referee calls or last-minute plays; however, it would be interesting to figure out if there is a way to get an edge using publicly available information.

NFL statistical data and sportsbooks lines and odds need to be collected as part of the project. The NFL statistical data will be collected from http://www.statheads.com, which is a website that provides the ability query and download NFL data. The website uses APIs to retrieve NFL data and structures it into databases for ease of querying. The sportsbook lines and odds will be collected from http://www.fantasydata.com and http://www.footballlocks.com. These websites provide historical point spreads and lines for free. The NFL data will be used to produce features for the dataset and the sportsbook lines and odds will be joined to produce the target variable as well as additional features.

Problem Statement

Machine Learning Engineer Nanodegree

Capstone Project

The project is a classification project. The goal is that given the statistics of two NFL teams and the point spread and odds, the classification model can predict the winner of the game against the spread. The statistics, such as point spreads and odds, will be the feature variables of the data set. The winner of the game will be the target variable. The target variables can either be favorite (1) or the underdog (0). The classification model will be trained using a training dataset and tested using a test dataset. The predicted target variables will be compared against the test dataset and the accuracy will be calculated to evaluate how well the model performs. The accuracy will be evaluated against other models as well as the baseline.

Metrics

For this classification problem, the accuracy will be used as the primary evaluation metric. The target variable is a binary and there is no penalty on selecting one value over the other. The primary goal is to predict the correct target variable (winner). Accuracy is a value between 0 and 1 and also can be presented as a percentage. The higher the accuracy, the better.

Accuracy = number of items classified correctly / all items in the dataset

II. Analysis

Data Exploration

The data contains data points that are known prior to the game starting such as average points score, winning percentage, passing yards, etc. Any statistics of the actual game being predicted has been excluded. Each line represents that an NFL game. The target variable is called the "spreadflag", which is set to 1 if the favorite wins against the spread; otherwise, it is set to 0. For the purposes of this project, games that were considered a "push" have been removed. A "push" is when the point differential matches the spread for the game. In other words, nobody wins against the spread. The following table details all the fields within the dataset:

Variable Name	Feature/ Target	Variable Type	Definition	Example	Mean	Std. Dev
season	feature	integer	Year the season starts	2010	2012.948	1.9859092
week	feature	integer	Week number between 1 and 17	1	8.95716	4.9491764
gameweek	feature	integer	Combination of the season and week	201001	201303.7	198.46366
favorite	feature	categorical	Team considered the favorite for this game	NYJ		
spread	feature	float	Point spread or line	-1	-4.94249	3.2888327
underdog	feature	categorical	Team considered the underdog for this game	BAL		
total	feature	float	Total points odds	36	44.85358	4.237705
awayML	feature	integer	Straight up winning odds of the away team	-120	-271.191	362.56885
homeML	feature	integer	Straight up winning odds of the home team	100	202.7606	208.97221
favoritehome	feature	integer	Flag set to 1 if the favorite team is the home team	1	0.682512	0.4656358
spread_0to3	feature	integer	Flag set to 1 if the spread is within a field goal (0 to 3)	1	0.466549	0.4990262
spread_35to7	feature	integer	Flag set to 1 if the spread is within touchdown (3.5 to 7)	0	0.365023	0.4815781

spread_75to10	feature	integer	Flag set to 1 if the spread is within a touchdown and field goal (7.5 to 10)	0	0.102113	0.3028853
spread_105to14	feature	integer	Flag set to 1 if the spread is within 2 touchdowns (10.5 to 14)	0	0.056925	0.231767
spread_145plus	feature	integer	Flag set to 1 if the spread is more than 2 touchdowns (10.5 to 14)	0	0.00939	0.0964726
fav_as_fav_last_5_ats_percent	feature	float	Favorite’s against the spread (ATS) as a favorite over the last 5 games	0.8	0.463498	0.2184254
und_as_und_last_5_ats_percent	feature	float	Underdog’s ATS as an underdog over the last 5 games	0.2	0.488615	0.2238025
fav_last_5_percent	feature	float	Favorite’s straight up (SU) winning percent over the last 5 games	0.8	0.578052	0.2462867
und_last_5_percent	feature	float	Underdog’s SU winning percent over the last 5 games	0.6	0.407746	0.2477589
fav_last_5_ats_percent	feature	float	Favorite’s ATS winning percent over the last 5 games	0.8	0.514437	0.2203463
und_last_5_ats_percent	feature	float	Underdog’s ATS winning percent over the last 5 games	0.4	0.441667	0.2106413
fav_score_last5	feature	float	Favorite’s average points scored over the last 5 games	23.6	24.3868	5.4279092
und_score_last5	feature	float	Underdog’s average point scored over the last 5 games	26.8	20.81734	5.0675968
fav_spread_diff_last5	feature	float	Favorite’s spread differential (+/- points over opp minus the spread) scored over the last 5 games	-5	-0.60895	6.0594692
und_spread_diff_last5	feature	float	Underdog’s spread differential (+/- points over opp minus the spread) scored over the last 5 games	-10.4	-0.25543	5.9300863
fav_passyards_last5	feature	float	Favorite’s average passing yards over the last 5 games	117.2	243.637	46.431984
und_passyards_last5	feature	float	Underdog’s average passing yards over the last 5 games	163.8	224.2658	43.863255
fav_rushyards_last5	feature	float	Favorite’s average rushing yards over the last 5 games	196.4	116.1207	27.381917
und_rushyards_last5	feature	float	Underdog’s average rushing yards over the last 5 games	182.6	109.914	28.023699
fav_tolost_last5	feature	float	Favorite’s average turnovers (TO) lost scored over the last 5 games	0.8	1.439818	0.6368093
und_tolost_last5	feature	float	Underdog’s average TOs lost scored over the last 5 games	1.8	1.635593	0.6191402
fav_firstdowns_last5	feature	float	Favorite’s average first downs over the last 5 games	17.2	20.5145	2.8794223
und_firstdowns_last5	feature	float	Underdog’s average first downs over the last 5 games	19	19.0081	2.6623457
fav_passyards_allowed_last5	feature	float	Favorite’s average passing yards allowed over the last 5 games	131.5	233.2788	39.294272
und_passyards_allowed_last5	feature	float	Underdog’s average passing yards allowed over the last 5 games	198.4	234.2129	35.976744
fav_rushyards_allowed_last5	feature	float	Favorite’s average rushing yards allowed over the last 5 games	77.6	107.881	24.825791
und_rushyards_allowed_last5	feature	float	Underdog’s average rushing yards allowed over the last 5 games	83.6	118.3067	25.712909
fav_togained_last5	feature	float	Favorite’s average TOs gained scored over the last 5 games	2	1.623621	0.6371139

und_togained_last5	feature	float	Underdog's average TOs gained scored over the last 5 games	3	1.450411	0.5927407
fav_to_diff_last5	feature	float	Favorite's average TOs gained minus TOs lost scored over the last 5 games	1.2	0.183803	0.9578976
und_to_diff_last5	feature	float	Underdog's average TOs gained minus TOs lost over the last 5 games	1.2	-0.18518	0.8944004
fav_firstdowns_allowed_last5	feature	float	Favorite's average first down allowed over the last 5 games	10.2	19.46573	2.5335486
und_firstdowns_allowed_last5	feature	float	Underdog's average first down allowed over the last 5 games	17.6	20.04842	2.4380607
fav_as_fav_last_10_ats_percent	feature	float	Favorite's against the spread (ATS) as a favorite over the last 10 games	0.5	0.457277	0.1620657
und_as_und_last_10_ats_percent	feature	float	Underdog's ATS as an underdog over the last 10 games	0.2	0.482864	0.1598583
fav_last_10_percent	feature	float	Favorite's straight up (SU) winning percent over the last 10 games	0.6	0.574531	0.1933945
und_last_10_percent	feature	float	Underdog's SU winning percent over the last 10 games	0.6	0.412676	0.1934424
fav_last_10_ats_percent	feature	float	Favorite's ATS winning percent over the last 10 games	0.6	0.513439	0.1481185
und_last_10_ats_percent	feature	float	Underdog's ATS winning percent over the last 10 games	0.4	0.444131	0.1468825
fav_score_last10	feature	float	Favorite's average points scored over the last 10 games	23.4	24.27775	4.4916024
und_score_last10	feature	float	Underdog's average point scored over the last 10 games	22.2	20.85454	4.1078879
fav_spread_diff_last10	feature	float	Favorite's spread differential (+/- points over opp minus the spread) scored over the last 10 games	-5.5	-0.63513	4.3536344
und_spread_diff_last10	feature	float	Underdog's spread differential (+/- points over opp minus the spread) scored over the last 10 games	-6	-0.18265	4.1434912
fav_passyards_last10	feature	float	Favorite's average passing yards over the last 10 games	144.6	242.36	40.721137
und_passyards_last10	feature	float	Underdog's average passing yards over the last 10 games	180.9	224.7152	36.678174
fav_rushyards_last10	feature	float	Favorite's average rushing yards over the last 10 games	177.8	116.4406	22.434246
und_rushyards_last10	feature	float	Underdog's average rushing yards over the last 10 games	145.1	109.9774	22.500557
fav_tolost_last10	feature	float	Favorite's average turnovers (TO) lost scored over the last 10 games	1.6	1.449537	0.4848882
und_tolost_last10	feature	float	Underdog's average TOs lost scored over the last 10 games	1.5	1.642514	0.456309
fav_firstdowns_last10	feature	float	Favorite's average first downs over the last 10 games	18.2	20.43075	2.4232056
und_firstdowns_last10	feature	float	Underdog's average first downs over the last 10 games	18.2	19.03086	2.2192756
fav_passyards_allowed_last10	feature	float	Favorite's average passing yards allowed over the last 10 games	148.1111	232.2789	29.442311
und_passyards_allowed_last10	feature	float	Underdog's average passing yards allowed over the last 10 games	186.7	234.6101	27.330079

fav_rushyards_allowed_last10	feature	float	Favorite's average rushing yards allowed over the last 10 games	88.4	108.3915	19.217098
und_rushyards_allowed_last10	feature	float	Underdog's average rushing yards allowed over the last 10 games	94.5	118.1775	20.04107
fav_togained_last10	feature	float	Favorite's average TOs gained scored over the last 10 games	2.1	1.626774	0.4724178
und_togained_last10	feature	float	Underdog's average TOs gained scored over the last 10 games	2.3	1.464391	0.4387803
fav_to_diff_last10	feature	float	Favorite's average TOs gained minus TOs lost scored over the last 10 games	0.5	0.177237	0.7187016
und_to_diff_last10	feature	float	Underdog's average TOs gained minus TOs lost over the last 10 games	0.8	-0.17812	0.6505451
fav_firstdowns_allowed_last10	feature	float	Favorite's average first down allowed over the last 10 games	13	19.43222	1.9964787
und_firstdowns_allowed_last10	feature	float	Underdog's average first down allowed over the last 10 games	17.2	20.02302	1.8820486
spreadflag	target	integer	Set to 1 if the favorite wins against the spread or 0 if the underdog wins against the spread	0	0.48885	0.5000224

The target variable distribution is somewhat evenly distributed. The following table shows the distribution of the target variable for the entire dataset as well as the distribution by season:

Season	Favorite	Underdog	Total Season Records
2010	120	130	**250**
2011	118	127	**245**
2012	117	133	**250**
2013	132	116	**248**
2014	120	131	**251**
2015	107	123	**230**
2016	119	111	**230**
Total Records	833	871	**1,704**

Exploratory Visualization

The dataset was analyzed to determine if there are any particular features that would be predictive of the target variable. In summary, there was not a single variable that was highly predictive of the target variable. This was expected since NFL game outcomes can be unpredictable and expectations do not match the actual results of the game.

Figure 1 visualization shows the target variable by the point spread and the count by the point spread. The first plot displays target variable is represented as the favorite winning percentage. The second chart displays the count of games by point spread. The first plot shows that roughly 50% of games with spreads within -15 to 0 are won by the favorite. When the spread is -17,-18, or -19, the favorite wins 100% of the time; however, the second plot shows that very few games have a spread of those values. The second plot also shows that majority of the games have a spread value between -10 to 0. The count by spread has a left-skewed distribution.

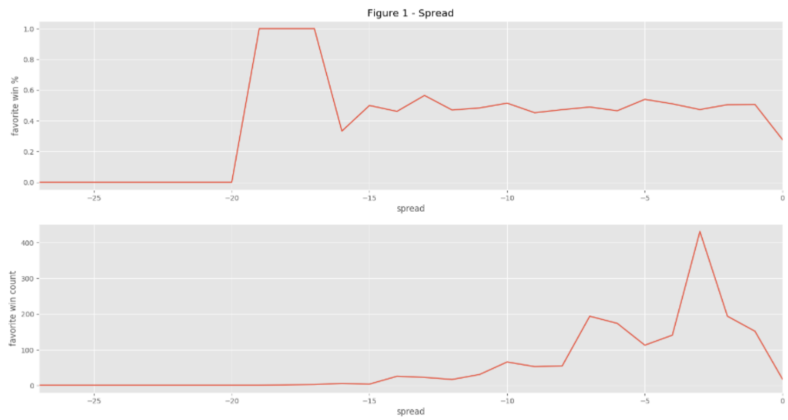


Figure 2 visualization shows that total is also not very predictive as well. There are some total values greater than 55 that result in a 0% and 100% winning percentage; however, there are some a few data points where these occur. The count by total is normally distributed.

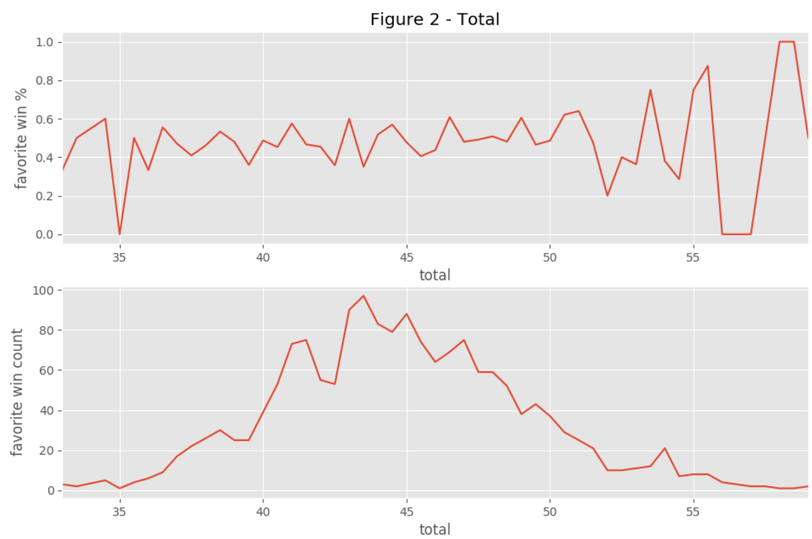


Figure 3 shows that the favorite winning % over the past 5 games is slightly predictive. There is slight variation in terms of favorite's winning percentage depending on the favorite's winning percentage over the last 5 games. The variation is within 2-3 percentage points within 50%. The favorite's winning % over the past 5 games is normally distributed.

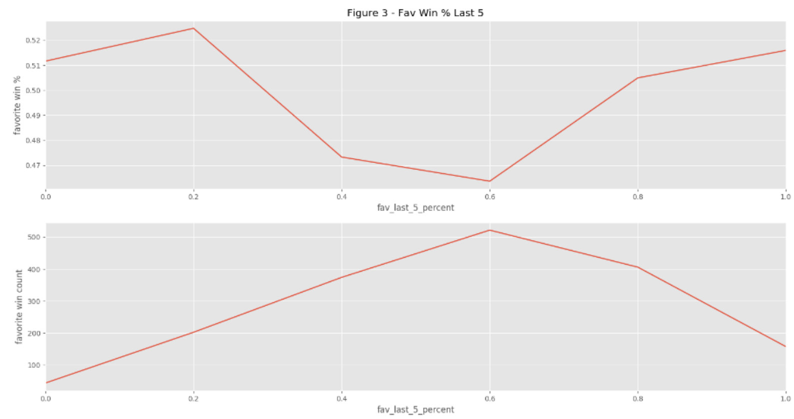
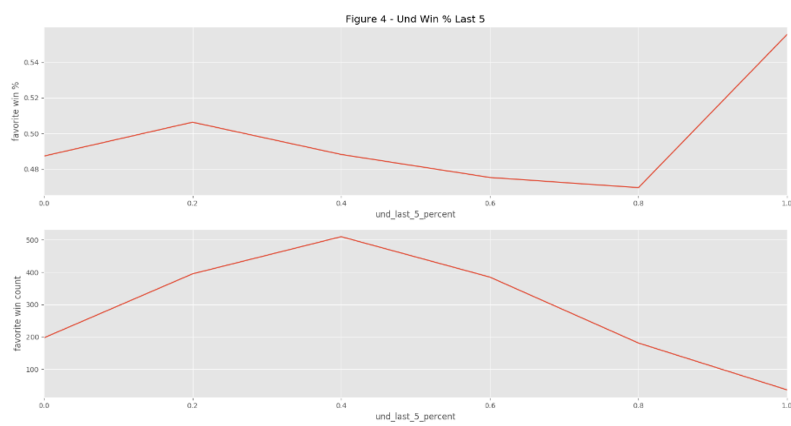


Figure 4 shows that the underdog winning % over the past 5 games is slightly predictive. There is slight variation in terms of favorite's winning percentage depending on the underdog's winning percentage over the last 5 games. Figure 4 shows that if the underdog has won the past 5 games; there is a higher probability that the favorite will win; however, there only a few data points for this instance. The underdog's winning % over the past 5 games is normally distributed.



In summary, the visualizations show that there is no single feature variable that is predictive on which team will win the game against the spread. There are some outliers cases where a clear boundary line can be drawn. Majority of the features have the target variable somewhat evenly distributed. This is expected since there is too much inconsistency in the outcome of an NFL game and the point spreads set the sportsbooks are designed to get an equal amount of money on both sides of the bet. The expectation is that the combination of the various feature variables can produce a predictive model that can predict better than the benchmark.

Algorithms and Techniques

The approach for this problem will start with the collection of the NFL data and the betting line and odds. The data will be aggregated and transformed to produce a dataset that can be fed into a machine learning algorithm. Each NFL game data will be included in a single line with all the features and a single target variable. The process of collecting the data required extensive manual data manipulation to ensure that the data was aggregated at the game level. The process of collecting the betting data for this dataset was a manual effort as well since the website used required manual scraping.

The data will be trained using classification machine learning algorithms with the goal of using one of the ensemble learning methods as the final solution. The commonly used classification algorithm, such as logistic regression, decision trees, and random forest, will be used to evaluate each individual performance. It is expected that logistic regression and decision trees will not perform well with the dataset since the dataset contains a target variable that is somewhat evenly distributed across a majority of the features. In particular, the Logistic

Regression model will underperform because the relationship between variables will be complex and may not be linear. The Decision Tree model is expected to overfit and result in poor testing performance. The ensemble methods, Random Forest, Ada Boost, and XG Boost, are expected to be the best fit for this dataset. The expectation is that the XG Boost will be the best performing algorithm.

Logistic Regression

Logistic Regression is a model that uses the logistic response function to produce a prediction between 0 and 1. Values above 0.5 are generally classified as a 1 and values below 0.5 are classified as a 0. [4]

Parameters[5]

- Penalty – The type for penalization for incorrect classifications during fitting

Decision Trees is a tree model that splits the variables based on decision rules to produce a prediction. The Decision Tree model is the easiest model to visually understand. [6]

Parameters[7]

- criterion - The metric used to measure the quality of the split. There is “gini” impurity and “entropy” for information gain
- max_depth - The maximum allowed depth for the tree
- min_samples_leaf

Random Forest Algorithm is an algorithm that builds a user inputted number of decision trees. Each decision tree is fully grown and when a prediction is made, each tree returns a prediction and the prediction will the most “votes” is returned as the final prediction.[8] Random Forest generally produces a more generalized model that can produce better predictions in comparison to the decision tree model.

Parameters[9]

- n_estimators – The number of trees to grow
- max_depth – The maximum allowed depth for every tree
- min_samples_split – The minimum number of children requires for a split node

Ada Boost is a boosted tree algorithm, which created decision trees that are grown sequentially and each tree grown are based on the information from the previous trees. Ada Boost grows trees and changes the weights of variables to account for misclassified target variables for the next tree. This process is repeated until the number of trees or “estimators” set by the user has been reached.[10] Adaboost is expected to perform better than the Random Forest algorithm.

Parameters[11]

- n_estimators – The number of trees to grow
- learning_rate – The variable that controls how fast or how slow the model will learn

XG Boost is a gradient boosted tree algorithm that uses gradient descent to minimize the training loss as it grows new trees. XG Boost also uses regularization to control overfitting of the model. XG Boost sequentially generates models and corrects the errors made from previous trees until the user-inputted number of trees are generated or until there is no improvement in the minimization of the training loss.[12]

Parameters[13]

- n_estimators – The number of trees to grow
- max_depth – The maximum allowed depth for every tree
- learning_rate – The variable that controls how fast or how slow the model will learn
- min_child_weight – The minimum weight required for a child node
- eval_metric – The evaluation metric used to evaluate the model. Default is ‘error’ or accuracy

Machine Learning Engineer Nanodegree

Capstone Project

The first approach will require data modeling using game data from 2010 to 2015 as the training dataset and the game data from 2016 as the testing dataset. As a part of the first approach, the various machine learning algorithm will be generated using the default settings. The next step is to generate models with the parameter tuned and the best performing algorithm will be selected. The feature selection will be applied to the best performing algorithm. Once that is completed, additional features will be added and the same model will be retrained and retuned along with feature selection.

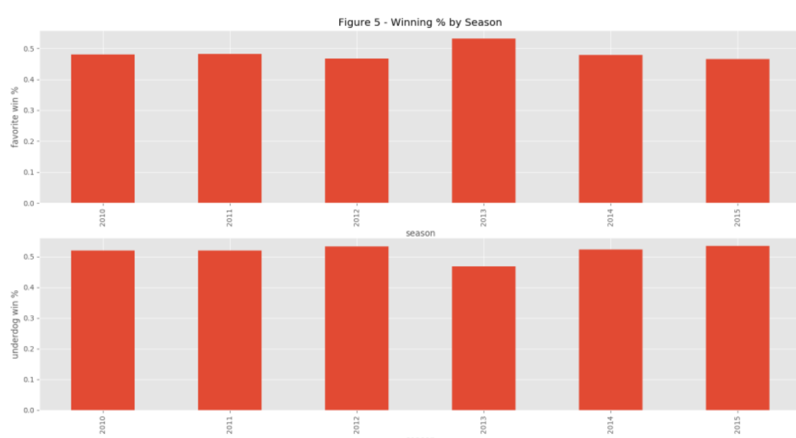
The second approach will use a weekly approach where the model is trained with the data up to the given week and the week's games will be predicted and then the model will be retrained with the latest available data and then predict the next week of games. This would mimic the real-world application of this algorithm where the model will be retrained and tested as the NFL season progresses. The second approach will be processed using the best performing model from the first approach. This approach is to prove or disprove a hypothesis that a model that is retrained with the latest data would be more predictive than the model trained on the data from the prior seasons. Once the ideal dataset and model have been identified, the accuracy will be compared against the baseline.

If the solution does not beat the baseline or to further improve accuracy, other features will need to engineered or collected such as player level statistics or weather. This will require an additional effort of identifying, collecting, and validating the new datasets. If new features are introduced to the dataset, the entire solution approach will be repeated.

Benchmark

The benchmark model will either be the overall accuracy or the average accuracy by season within the training dataset. The benchmark will be the better accuracy between always selecting the favorite or underdog. This naïve approach will be the best method of generating a realistic benchmark since randomly selecting the favorite or underdog will produce different results every time. The weekly accuracy will be averaged to account for the variance in accuracy of the naïve approach. Both baselines will be analyzed to determine which baseline would be appropriate for this problem. The expectation is that the baseline would be within 50% range.

Using the training dataset, the favorite and underdog winning percentages were calculated by year as well as the overall accuracy for each. Figure 5 shows that both winning percentages are very close to 50%.



The overall accuracy for the favorite is 0.48420 and the overall accuracy for the underdog is 0.51580. When comparing the years, the favorite and underdog winning percentages are within 3-4 percentage points of 50%. In order simplify the benchmark used for model evaluation, the overall accuracy of the best accuracy. Based on this finding, the naïve approach of selecting the underdog every time will be considered the benchmark. A benchmark of 0.51580 or (51.6%) will be used to evaluate the performance of the model.

Winning % by Season		
Season	**Favorite**	**Underdog**
2010	0.48000	0.52000

2011	0.48163	0.51837
2012	0.46800	0.53200
2013	0.53226	0.46774
2014	0.47809	0.52191
2015	0.46522	0.53478

Overall Favorite Winning %	
Favorite	**Underdog**
0.48420	0.51580

III. Methodology

Data Preprocessing

Since data has been structured to be manually processed with the intent of use in machine learning, there are a few data preprocessing steps to account for unexpected data issues and to handle the categorical variables within the dataset. Based on the data compiled, the following steps need to take to complete data preprocessing:

1. Fill any NaNs within the data
2. One-hot code the favorite and underdog feature variables since both variables are categorical.
3. Drop the original favorite and underdog feature variables
4. Add the one-hot coded variables into the dataset.

Implementation

Once the data was preprocessed, the dataset was split into a training and testing dataset. The training dataset contained data from the 2010 to 2015 seasons and the testing data contained the 2016 season. Each dataset was split into 2 dataframes: features and target. The features dataset contained all the variables except for the 'spreadflag' variable. The target dataframe contained only the 'spreadflag' variable only.

First Run – Default Settings

The first run ran several machine learning algorithms using the default settings to do an initial high-level evaluation of the performances of each algorithm. The expectation is that XG Boost would be the best performing algorithm. The following machine learning algorithms were used for the first run:

Machine Learning Algorithm	Library
Logistic Regression	from sklearn.linear_model import LogisticRegression
Decision Tree Classifier	from sklearn.tree import DecisionTreeClassifier
Random Forest Classifier	from sklearn.ensemble import RandomForestClassifier
Ada Boost Classifier	from sklearn.ensemble import AdaBoostClassifier
XG Boost	import os import xgboost as xgb

Before the machine learning algorithms are processed, the data was split into the training dataset and testing dataset. The training data contained 2010 to 2015 season data and the testing dataset contained 2016 season data. For each dataset, the data was split into two dataframe: features and target. The features dataframe contains all variables except for the spreadflag variable and the target dataframe contains the spreadflag variable. For each machine learning algorithm, the algorithm was initialized, fitted against the training data, and then scored using the training and testing dataset. The training and testing accuracies are compiled into a dataframe and outputted into a CSV. There were no challenges faced with the coding process of the first run.

The first run using defaults only produced unexpected results. The best performing machine learning algorithm was Ada Boost with a testing accuracy of 0.54348 and the worst performing algorithm was Logistic Regression with a testing accuracy of 0.47391. Unexpectedly, XG boost had a poor testing accuracy of 0.49130. When comparing the training and testing accuracy, it appears that the Decision Tree, Random Forest, and XG Boost models are highly overfitted. Overall, the Ada Boost model is the best model with a training accuracy of 0.675712347 and a testing accuracy of 0.543478261. These are the training and testing accuracy of the first run with default settings:

Machine Learning Algorithm	Training Accuracy	Testing Accuracy
log_reg	0.567842605	0.473913043
d_tree	1	0.5
rf_tree	0.983039349	0.543478261
Adaboost	0.675712347	0.534782609
xg_boost	0.883989145	0.491304348

Second Run – Parameter Tuning

The second run included the same machine algorithms with the parameters tuned. The same libraries from the first run are used to initialize the model, but the model creation and parameter tuning was done in a separate function. The parameter tuning was processed by creating functions borrowed from the Boston Housing project.

Purpose	Library
Grid Search Cross Validation	from sklearn.grid_search import GridSearchCV
Split the dataset for Cross Validation	from sklearn.cross_validation import ShuffleSplit

The function used the GridSearchCV function to process user-inputted tuning parameters and returned the best performing model. With XGBoost, there were two ways to use cross validation to tune the model. There is one using GridSearchCV function and the xgboost.cv function within the xgboost library. Using the xgboost.cv function required a lot of trials to tune the parameters, but it produced the best performing model. The difficulty of the second run was the amount of time it took to select the tuning parameters that would influence the accuracy as well as the array of values to tune. There was a lot of trial and error in the process of selecting the tuning parameters and the values to pass for each one. The second run also took a lot of time to process and by far was the run that took the most amount of time to complete. The machine learning algorithm that took the most time to process was the Random Forest algorithm because the performance time was dependent on the number of trees generated and each tree generated was fully grown.

The second run with the parameter tuning produced different, but expected results. Overall, the parameter tuning produced more generalized models. The testing accuracy of the Logistic Regression and XG Boost increased. The rest of the machine learning algorithm had a decrease in testing accuracy; however, the models were less overfitted compared to the models from the first run. The best performing model was XG Boost with a training accuracy of 0.64654 and testing accuracy of 0.56087. The model is still slightly overfitting, but it has produced the highest accuracy so far. These are the accuracy for the second run with parameters tuned:

Machine Learning Algorithm	Training Accuracy	Testing Accuracy	Best Parameters
log_reg_tuned	0.619402985	0.504347826	{'penalty': 'l1', 'random_state': 319}
d_tree_tuned	0.689280868	0.491304348	{'max_features': 'auto', 'random_state': 319, 'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 4}
rf_tree_tuned	0.983039349	0.543478261	{'bootstrap': False, 'min_samples_leaf': 4, 'n_estimators': 50, 'random_state': 319, 'max_features': 'auto', 'max_depth': 3}

ababoost_tuned	0.553595658	0.526086957	{'n_estimators': 10, 'learning_rate': 0.01, 'random_state': 319}
xg_boost_tuned with Grid Search CV	0.645861601	0.556521739	{'n_jobs': 4, 'learning_rate': 0.0001, 'min_child_weight': 1, 'n_estimators': 100, 'random_state': 319, 'max_depth': 6}
xg_boost_tuned with xgboost.cv	0.646540027	0.560869565	

Refinement

Third Run – Feature Selection

Based on the first two runs, it was identified that a parameter tuned XG Boost using the xgboost.cv function produced the best accuracy in comparison to the other machine learning algorithms. Therefore, the further refinement was focused only XG Boost. The first step taken was to do feature selection on the feature dataset to remove variables that are not very predictive.

Purpose	Library
Feature Selection	from sklearn.feature_selection import SelectFromModel

The third run applied feature selection on the dataset prior to training the XG Boost model. The feature importance scores were pulled from the model and then the importance score of every feature variable was used as a threshold to filter the feature dataset using the SelectFromModel function from the feature_selection library. Then xgboost.cv is executed against the filtered feature dataset. This process was repeated until every importance score of every feature variable was used as a threshold. This is the distinct result of every iteration:

Importance Score	Training Accuracy	Testing Accuracy
0	0.646540027	0.560869565
0.001650165	0.645861601	0.556521739
0.00330033	0.645861601	0.556521739
0.004950495	0.642469471	0.573913043
0.00660066	0.642469471	0.569565217
0.008250825	0.641112619	0.569565217
0.00990099	0.646540027	0.565217391
0.011551155	0.679782904	0.52173913
0.013201321	0.672320217	0.504347826
0.014851485	0.666214383	0.560869565
0.01650165	0.660786974	0.543478261
0.018151816	0.665535957	0.543478261
0.01980198	0.6614654	0.547826087
0.021452146	0.6614654	0.543478261
0.02310231	0.663500678	0.543478261
0.024752475	0.665535957	0.582608696
0.026402641	0.673677069	0.508695652
0.028052805	0.671641791	0.482608696
0.029702971	0.659430122	0.47826087
0.0330033	0.645861601	0.473913043

0.034653466	0.647218453	0.460869565
0.036303632	0.594979647	0.513043478
0.037953794	0.581411126	0.513043478
0.041254126	0.571913161	0.469565217

The ideal importance score to use a threshold was 0.024752475. This value was selected to produce an updated dataset and to run xgboost.cv one more time. The updated feature dataset contains 39 features compared to the original dataset, which contained 71 features. This is the final training and testing accuracy of the updated model was 0.66553 and 0.58261 respectively.

Training Accuracy	Testing Accuracy
0.665535957	0.582608696

Fourth Run – Add New Features

The fourth run included the inclusion of additional features with the expectation that providing additional data will improve the testing accuracy of the model. The date and time variables were manually extracted from the footballlocks.com and the time zone variables were extracted from the existing data from statheads.com. The reason behind adding these variables is to see if the day or time of the game can improve predictability of the outcome of the game. Also, the time zone variables were added to see if the time changes for traveling teams can improve accuracy.

Variable Name	Feature/ Target	Variable Type	Definition	Example
Sunday	feature	integer	Flag set to 1 if the day of game is Sunday	0,1
Monday	feature	integer	Flag set to 1 if the day of game is Monday	0,1
Thursday	feature	integer	Flag set to 1 if the day of game is Thursday	0,1
Morning	feature	integer	Flag set to 1 if game time is in the morning	0,1
Afternoon	feature	integer	Flag set to 1 if game time is in the afternoon	0,1
Night	feature	integer	Flag set to 1 if game time is at night	0,1
Central-Central	feature	integer	Flag set to 1 if the away team's time zone is Central and home team's time zone is Central	0,1
Central-East	feature	integer	Flag set to 1 if the away team's time zone is Central and home team's time zone is East	0,1
Central-Mountain	feature	integer	Flag set to 1 if the away team's time zone is Central and home team's time zone is Mountain	0,1
Central-West	feature	integer	Flag set to 1 if the away team's time zone is Central and home team's time zone is West	0,1
East-Central	feature	integer	Flag set to 1 if the away team's time zone is East and home team's time zone is Central	0,1
East-East	feature	integer	Flag set to 1 if the away team's time zone is East and home team's time zone is East	0,1
East-Mountain	feature	integer	Flag set to 1 if the away team's time zone is East and home team's time zone is Mountain	0,1
East-West	feature	integer	Flag set to 1 if the away team's time zone is East and home team's time zone is West	0,1
Mountain-Central	feature	integer	Flag set to 1 if the away team's time zone is Mountain and home team's time zone is Central	0,1
Mountain-East	feature	integer	Flag set to 1 if the away team's time zone is Mountain and home team's time zone is East	0,1
Mountain-Mountain	feature	integer	Flag set to 1 if the away team's time zone is Mountain and home team's time zone is Mountain	0,1

Mountain-West	feature	integer	Flag set to 1 if the away team's time zone is Mountain and home team's time zone is West	0,1
West-Central	feature	integer	Flag set to 1 if the away team's time zone is West and home team's time zone is Central	0,1
West-East	feature	integer	Flag set to 1 if the away team's time zone is West and home team's time zone is East	0,1
West-Mountain	feature	integer	Flag set to 1 if the away team's time zone is West and home team's time zone is Mountain	0,1
West-West	feature	integer	Flag set to 1 if the away team's time zone is West and home team's time zone is West	0,1

The fourth run was the same as the third run, but with a new dataset. Xgboost.cv was used to tune the model and feature selection was done using the feature importance score as a threshold. For feature selection, the best threshold is 0.0, which means that none of the variables are excluded.

Importance Score	Training Accuracy	Testing Accuracy
0	0.647896879	0.591304348
0.001672241	0.647896879	0.586956522
0.003344482	0.647896879	0.586956522
0.005016722	0.647896879	0.582608696
0.006688963	0.643826323	0.573913043
0.008361204	0.643147897	0.560869565
0.010033445	0.641791045	0.560869565
0.011705685	0.667571235	0.504347826
0.013377926	0.662822252	0.504347826
0.015050167	0.664857531	0.552173913
0.016722407	0.656716418	0.560869565
0.018394649	0.651967436	0.47826087
0.020066889	0.648575305	0.491304348
0.025083613	0.649253731	0.491304348
0.026755853	0.649253731	0.491304348
0.028428094	0.65468114	0.486956522
0.030100334	0.659430122	0.460869565
0.031772576	0.668928087	0.47826087
0.033444814	0.656716418	0.486956522
0.036789298	0.62550882	0.517391304
0.040133778	0.57734057	0.5

The final testing accuracy improved from 0.58260 to 0.59130, which is the highest testing accuracy of all the models generated. Also, this model is slightly less overfitting in comparison to the previous model with a training accuracy of 0.6479 compared to 0.66554.

Training Accuracy	Testing Accuracy
0.64789687924	0.591304347826

Fifth Run – Weekly Iteration

The fifth run was to identify if the model will produce better testing accuracy if the model is retrained on the data prior to a given week and a prediction is produced. Then the process was repeated until all weeks have been predicted. The XG Boost model tuned from the third run was used. This model with the weekly iteration actually performed worse with an overall testing accuracy of 0.50. It appears that loaded the most recent data for retraining is causing the model to overfit and does not generalize better than the model produced in the fourth run. It appears that a better testing accuracy can be achieved if the parameters were tuned for every week, but changing the parameters on a week to week basis seemed like a poor solution with a lot of volatility.

Week	Testing Accuracy
1	0.428571
2	0.3125
3	0.466667
4	0.533333
5	0.461538
6	0.692308
7	0.6
8	0.384615
9	0.6
10	0.357143
11	0.571429
12	0.533333
13	0.4
14	0.625
15	0.4375
16	0.625

Overall_testing_acc
0.5

IV. Results

Model Evaluation and Validation

The final XG Boost produced from the fourth run aligned with solution expectations. The final model parameters were adjusted until the optimal testing accuracy was achieved. The number of boosting rounds was set to 50 and the learning rate was set to 0.0005, which achieved the ideal combination of parameter values to produce the best testing accuracy. The final model applied feature selection to remove features that did not improve performance. This ensured that the feature set was optimized even though no features were removed. The final model was tested using the entire 2016 season data, which was a enough data points to be able to clearly evaluate the models and be able to pick the best model.

The final model appears to robust enough to predict the entire season of games. The fifth run of weekly iterations of retraining and predicting produced a poorer testing accuracy of 0.50. This proves that the final model generalizes better than the model produced from weekly iterations even though the weekly iterations process incorporated the most recent data. The final model can be trusted to a certain extent since a lot of things can happen within a single NFL season such as injuries. It would be ideal to testing this model with the upcoming 2017 season to determine the trustworthiness of the model.

Machine Learning Engineer Nanodegree

Capstone Project

Justification

The final model produced a testing accuracy of 0.59130, which is equivalent to an increase of 7.5% over the benchmark accuracy of 0.51580. If the naïve approach of picking underdogs was done for the 2016 season, the testing accuracy would have been 0.48260. The final model performed 10.9% better than the naïve approach. The final solution is significant enough that someone could use the model to can make money betting on NFL games with the assumption that the person bets on every single game. The model generalizes well, so using the model to predict a small set of games may result in poor results since game outcomes can be unpredictable.

V. Conclusion

Free-Form Visualization

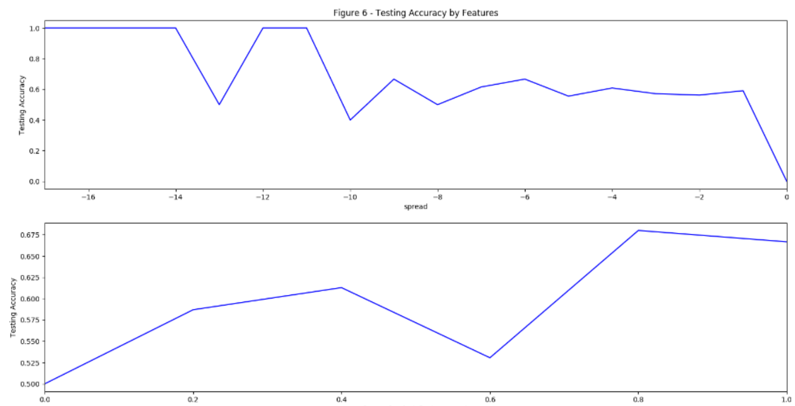


Figure 6 contains two visualizations of the testing accuracy compared to two features. The first visualization shows the testing accuracy over spread values. It shows that the model has a higher accuracy for spread values less than -10 and any spread value less than or equal to 10 has an accuracy that is around 50%. This shows that the model can predict games more accurately simply based on the spread value if the spread is less than -10. The second visualization shows the testing accuracy over the favorite ATS winning percentage as a favorite. It shows that model has a higher tendency to make a correct prediction of the favorite's ATS winning percentage is 0.8 or 1.0. It indicates that team's a higher ATS winning percentage can be indicative of future outcomes.

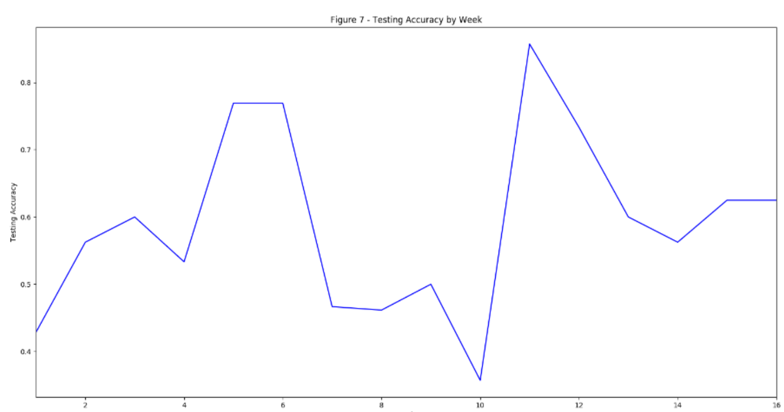


Figure 7 shows the testing accuracy by week. It visualizes the fluctuation of the testing accuracy, which indicates that this model is probably not meant for use of predicting a single or subset of games. In Weeks 1,7,8, and 10, there is a high probability that the person using this algorithm will actually lose money. The real-world application of this model would require the person to beat an equal amount of money on every single game for the model to be useful.

Reflection

In summary, the NFL predicting model against the point spread was developed using a sequential process of running and tuning machine learning algorithms and evaluating the accuracy. After establishing a benchmark using a naïve approach to predicting NFL games, Logistic Regression, Decision Trees, Random Forest, Ada Boost, and XG Boost machine learning algorithms were generated using the default settings to obtain a general idea of how the models performed. Then, the parameters were tuned and the same set of models were reprocessed, which provided a better understanding of what model would be the best performing. Based on the testing accuracy, the XG Boost was selected. In order to further improve the testing accuracy, feature selection was completed using the feature importance scores from the feature importance attribute of the XG Boost model, which resulted in an increase in testing accuracy. Then to further improve testing accuracy, additional features were added and feature selection was completed against the new dataset. This resulted in another increase in testing accuracy. The final step was to determine if retraining the model using the latest data for every week will result in improved performance. However, the testing accuracy dropped significantly, which meant that the model using the latest data did not perform better than the XG Boost model based on the prior season data only.

The interesting aspect of this project was that the amount of data used in training the model can have a drastic impact on the performance of the model using the NFL dataset. Using the prior 5 seasons data, the model performed much better than retraining the model on a weekly basis. In order words, it is not always necessary needed to provide up to the minute data to produce the best predictions and that using the prior 5 seasons of data is sufficient to build a generalized model.

It was also interesting to see the changes in the accuracies of the several machine learning algorithms in comparison to the training and testing accuracy. Using the default settings, the Decision Tree and Random Forest were highly overfitted, which resulted in a larger variance between the training and testing accuracy. By parameter tuning these models, the models became more generalized; however, it did not necessarily result in an increase in testing accuracy.

Improvement

In regards to improvement, the process of generating the final model appears to be solid; however, there are a lot of opportunities on the data collection aspect of this problem. There are a lot more data points that can be captured such as key injuries, weather, player level statistics, and power rankings (arbitrary ranking provided by sports writers at a given point in time). Some of this data might be difficult to gather for historical games and in some cases, some of the data may not be possible to collect in a timely manner. Some injuries news occurs a few hours before game time. Collecting new feature variables for this project may produce a minor increase in testing accuracy.

Machine Learning Engineer Nanodegree

Capstone Project

Citations

[1] Shannon-Missal, L. (2016, January 26). Pro Football is Still America's Favorite Sport. Retrieved from, http://www.theharrispoll.com/sports/Americas_Fav_Sport_2016.html

[2] Super Bowl 51 – By the Numbers. (2017, January 17). Retrieved from https://www.americangaming.org/research/infographics/super-bowl-51-numbers

[3] Sports Betting. (n.d.) In Wikipedia. Retrieved August 8th, 2017, from https://en.wikipedia.org/wiki/Sports_betting

[4] Analytics Edge – Unit 3 - An Introduction to Logistic Regression. (n.d.) In EDx. Retrieved August 31st, 2017 from https://courses.edx.org/courses/course-v1:MITx+15.071x_3+1T2016/course/

[5] sklearn.liner_model.Logistic Regression. Retrieved August 31st, 2017, from http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6] Analytics Edge – Unit 4 – An Introduction to Trees. (n.d.) In EDx. Retrieved August 31st, 2017 from https://courses.edx.org/courses/course-v1:MITx+15.071x_3+1T2016/course/

[7] sklearn.tree.DecisionTreeClassifier. Retrieved August 31st, 2017, from http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[8] Analytics Edge – Unit 4 – An Introduction to Trees. (n.d.) In EDx. Retrieved August 31st, 2017 from https://courses.edx.org/courses/course-v1:MITx+15.071x_3+1T2016/course/

[9] sklearn.ensemble.RandomForestClassifer. Retrieved August 31st, 2017, from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] Statistical Learning - 8.5 Boosting. (n.d.) In Stanford Lagunita. Retrieved August 31st, 2017 from https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/info

[11] sklearn.ensemble.AdaBoostClassifier. Retrieved August 31st, 2017, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[12] Brownlee, Jason. A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved August 31st, 2017 from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[13] Python API Reference - Scikit-Learn API. Retrieved August 31st, 2017 from http://xgboost.readthedocs.io/en/latest/python/python_api.html