Capstone Technical Report

Problem Statement: How accurately can we predict outcomes of NFL games by focusing on quarterly outcomes?

Executive Summary

I’ve always been interested in football and the NFL, so it was natural for me to choose an experiment about this topic. Inspired by an article by a naïve logistician who focused on correlating first-quarter NFL scores with their games’ final outcomes, I decided to broaden the scope of that experiment to include all quarterly outcomes.

My approach did not involve studying the discrete quarterly scores of each team during each NFL game; rather, it focused on the difference between the quarterly scores of each team which I referred to as the quarterly outcome. In other words, I only noted whether the visiting team scored more or fewer points than the home team in each period (and I also ignored overtime periods).

To obtain the data, I utilized some web scrapers to grab the quarterly scores and other information from two different sports websites. I collected the quarterly scores of every regular and postseason NFL game from the 2000 season through the 2019 season – over 5300 games in all.

I feature engineered four new “result” columns in the dataset, the values of which consisted of one of seven different “winner codes” denoting which team scored more points in the quarter and by how wide the margin was (as well as tied quarter scores). Then I added a target column to the dataset which indicated whether the home or visiting team won the game or whether it was tied after four quarters of play.

After subsetting the winner columns and one-hot encoding the dataframe, I grouped the resulting 28 features into what I called “clumps,” which represented all 15 combinations of the four quarters’ winner columns. This allowed me to model on combinations of quarterly outcomes rather than just individual ones.

The start of the modeling process utilized nine different estimators modeled over all of the quarterly clumps by themselves. The next pass included the week of the season in which each game was played, while the third pass did the same with the season year of the contest. The final pass modeled the quarterly clumps with all 32 NFL teams that played in those games. Each of the more than 12,000 distinct modeling passes was recorded in a database along with its accuracy score.

I began my data analysis with the first pass of modeling results involving only the clumps of quarterly outcomes. I then took the mean accuracy scores for all 14 clumps and put the results into a line graph along with an illustration of the maximum accuracy scores by clump. As expected, the clump with the most variables (i.e., all four quarters of data) had the highest accuracy scores, followed by the clumps with trios of quarters, then by clumps with pairs of quarters, and finally by single-quarter data clumps.

When incorporating the season week data into the modeling process along with the quarter clumps, I graphed only the individual quarter clumps along with the baseline model (with an accuracy score of 53.63%). I observed that only two of the 68 regular season quarterly clumps had accuracy scores below the baseline; however, nine of the 16 postseason quarterly clumps did not outperform the baseline model. In a sense, this was expected since we might assume that teams in the postseason play differently than they do in the regular season because of the possibility of elimination that comes with a defeat.

Then I swapped out the season week data with the season year data and repeated the modeling process once again. Like the two previous modeling runs, the maximum accuracy scores tended to come from RandomForestClassifiers, while the minimum accuracy scores came exclusively from AdaBoost classifiers. After graphing the individual quarterly clumps against the season year, I noted the top three and bottom three accuracy scores and from which clumps in which years they were associated with.

Finally, I modeled the quarterly clumps with all 32 NFL teams that played the games in the dataset. Then I determined the maximum overall accuracy scores for each team (both when they were the visiting team and the home team). I then recorded the highest and lowest overall accuracies for each individual quarterly clump, as well as the “first half” and “second half” clumps and then all of the quarterly combinations. Then I took the average of all of the maximum accuracy scores associated with every team (as home as visiting teams): I dubbed this metric the “mean-max” accuracy score.

In addition, I constructed a web application on Streamlit and backstopped it with a RandomForestClassifier estimator with all combinations of the quarterly clumps so as to maximize the app’s accuracy. The user moved a slider bar to pick a row number from the original NFL dataset I collected, and the app displayed information about when the game was played and who won – and then ran the model to see if its prediction matched the actual outcome of the contest.

I came out of this experiment with the confirmation of what I felt going into it: that no matter how much data you may have or how meticulous your modeling process may be, it is quite difficult to predict the outcomes of NFL games. There are so many real-world variables to consider that the models I used did not take into account, which likely prevented me from posting higher accuracy scores than I did.

Should I choose to expand on this experiment in the future, I will aim to incorporate more variables and groupings of variables into the set of features used for modeling. I might also alter the winner code level cutoff amounts or use the raw numerical quarterly scoring margins instead. Finally, I could hypertune the models’ parameters using grid searching in order to boost accuracy scores a bit more.

As with many modeling experiments more questions than answers were generated. But thanks to my lifelong love of football, I’m heartened by the challenge and fascinated by the complexity of the scoring patterns of my favorite sport.

Table of Contents

* Background
* Assumptions
* Data Collection
* Feature Engineering
* Data Encoding and Grouping
* Data Dictionary
* Modeling
* Imported Libraries
* Analysis
* Streamlit app
* Conclusions
* Future Recommendations

Background

Before I became a data scientist, I had already amassed a great deal of experience in sports-related pursuits. My radio, TV, and Internet sports broadcasting career spans almost three decades and includes play-by-play and color analysis of high school and college sports of all kinds, including football. I’ve also been fortunate to work as an in-venue public address announcer for amateur and professional sporting events. Most importantly, I’ve been a lifelong NFL and football fan. So it’s easy to see why I would be drawn to an experiment involving NFL scoring patterns.

After exploring the idea of trying to predict specific quarterly football scores of NFL games, I came upon a June 2018 article on the ital-*towards data science-ital* blog written by Phillip Hale entitled, “NFL — Does winning in the first quarter really matter in the NFL?” [‘https://towardsdatascience.com/nfl-which-quarters-correlate-most-with-winning-87f23024c44a’] In the piece, Hale (a naive logistician) produced Python code for a supervised regression machine learning task that examined quarterly scores and several other variables in an attempt to determine whether they correlated with the outcome of the NFL game.

Hale’s dataset only [‘https://github.com/naivelogic/NFL-smarter-football/blob/master/nfl\_team\_stats.csv’ ] only used 288 regular season games over a period between 2009 and 2017. I decided to create an experiment that encompassed all NFL games dating back to 2000 and forward to 2019, including playoff contests.

Assumptions

Before I proceed any further, let me provide a quick primer into the structure of the NFL game schedules. There are 32 NFL teams, each of which plays 16 regular season games over the course of 17 weeks (including one ‘bye’ week). Teams which are eligible for the playoffs can play up to four additional games, the last of which is the world-renowned Super Bowl. Each game is divided into four 15-minute quarters, the scores of which will be the prime focus of this experiment.

My approach did not involve studying the discrete quarterly scores of each team during each NFL game; rather, it focused on the difference between the quarterly scores of each team which I refer to as the quarterly outcome. In other words, I only noted whether the visiting team scored more or fewer points than the home team in each period.

Even though NFL games that are tied after four quarters of play proceed to an overtime period to determine a winner, no overtime stanzas were considered for this experiment; instead, these contests were designated as a “tie” game. In addition, the master dataset contained additional information about each game, including date, day of week, and time of day of each game; the season year and week number of the season in which it was played; and the final score difference.

Data Dictionary

**QuarterlyWins.csv [‘data/QuarterlyWins.csv’]**

|Variable|type|Description|

|GameID|string| A combination of letters and numbers which served as each post's unique identifier.|

|Week|integer| The week number of the NFL season in which the game was played (week 1, week2, etc.) Three-digit numbers represent playoff weeks: 100 for the first (wild-card) round, 200 for the quarterfinal (divisional) round, 300 for the semifinal (conference championship) round, and 400 for the championship (Super Bowl) game.|

|Day|String|A three-character abbreviation of the day of the week on which the game was played.|

|DateStamp|string| The date of the game represented in YYYY-MM-DD format.|

|Vteam|string| The mascot of the NFL team who was designated as the visiting team in the game.|

|VQ1, VQ2,VQ3,VQ4|integer|The discrete numbers of points scored by the visiting team in the 1st, 2nd, 3rd, and 4th quarter, respectively.|

|Hteam|string| The mascot of the NFL team who was designated as the home team in the game.|

|HQ1, HQ2,HQ3,HQ4|integer|The discrete numbers of points scored by the home team in the 1st, 2nd, 3rd, and 4th quarter, respectively.|

|Date|string| The date of the game represented in M/D/YYYY format.|

|Time|string| The start time of the game.|

|RQ1, RQ2,RVQ3,RQ4|string|A code to represent which team scored the most points in the 1st, 2nd, third, and 4th quarter, respectively. HHH = home team scored at least 15 points more than the visiting team; HH = home team scored between 8 and 14 (inclusive) more points more than the visiting team; H = home team scored between 1 and 7 (inclusive) more points than the visiting team; T = the two teams scored the same amount of points; V = visiting team scored between 1 and 7 (inclusive) more points than the home team; VV= visiting team scored between 8 and 14 (inclusive) more points than the home team; VVV = visiting team scored at least 15 points more than the home team.|

|Win|integer| The difference between the sum of the scores in the VQ1 to VQ4 columns and the sum of the scores in the HQ1 to HQ4 columns. Therefore, a negative value signified a win by the home team, while a positive value corresponded to a victory by the visiting team.

|Winner|string|A code denoting which team won the game. HW = home win, VW = visiting win, T = tie.|

Data Collection

Because obtaining data directly from NFL-approved APIs was cost prohibitive, I constructed a couple of web scrapers to grab the data from two different sports websites. This block of R code pulled quarterly scores and related data from CBSSportsline.com’s NFL Scoreboard pages:

```

install.packages('rvest')

library(rvest)

library(stringr)

wk <- 15

season <- 2020

regular\_or\_playoffs <- 'regular'

if (wk > 17) {

regular\_or\_playoffs <- 'postseason'

}

surl <- paste0("https://www.cbssports.com/nfl/scoreboard/",season,"/",regular\_or\_playoffs,"/",wk)

scores\_pg <- read\_html(surl)

head(scores\_pg)

qtr\_scores <- html\_nodes(scores\_pg,'.scores')

qtr\_scores

raw\_teams <- html\_nodes(scores\_pg,'.team')

raw\_teams

parsed\_teams <- html\_text(raw\_teams,'')

teams\_playing <- parsed\_teams[-c(1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63)]

teams\_playing

teams <- qtr\_scores [-c(125,130)] #only if need to remove OT scores

teamsscores <- html\_text(teams,"") %>%

str\_extract("[0-9]{1,}") %>%

as.numeric()

teamsscores

max\_length <- 4 # of row

sequence\_teamsscores <- seq\_along(teamsscores)

split\_teamsscores <- split(teamsscores, ceiling(sequence\_teamsscores/max\_length))

transposed\_teamsscores <- t(data.frame(split\_teamsscores))

transposed\_teamsscores

week\_teams <- data.frame(matrix(teams\_playing, nrow = 32, ncol = 1))

week\_teams # column of teams that are playing

scorelines <- cbind(week\_teams, transposed\_teamsscores)

colnames(scorelines) <- c("Team", "Q1", "Q2", "Q3", "Q4")

scorelines

week\_season<- data.frame("Week" = wk, "Season" = season)

week\_season\_rows <-week\_season[rep(seq\_len(nrow(week\_season)), 32), ]# check to see if there are 32 teams playing this week

WEEKLY\_SCORES <- cbind (scorelines,week\_season\_rows )

WEEKLY\_SCORES

writeURL <- paste0("week\_",wk,"\_of\_",season,".csv")

write.csv(WEEKLY\_SCORES,writeURL)

```

I used the above code to web scrape scores and data for the 2015 through 2019 NFL seasons. For the 2000 through 2014 seasons, I used this block of R code to pull quarterly scores and related data from Pro-Football-Reference.com’s NFL Schedule pages:

‘’’

install.packages('rvest')

library(rvest)

library(stringr)

url = 'https://www.pro-football-reference.com/years/[season year]/games.htm'

read\_xml(x, encoding = encoding, ..., as\_html = TRUE, options = options)

webpage = read\_html(url)

table\_links = webpage %>% html\_node("table") %>% html\_nodes("a")

boxscore\_links = subset(table\_links, table\_links %>% html\_text() %in% "boxscore")

boxscore\_links = as.list(boxscore\_links)

bscorelinks <- str\_extract(boxscore\_links, "(boxscores).\*(htm)")

links <- NULL

gamesscores <- data.frame(matrix(ncol = 10, nrow = 0))

colnames(gamesscores) <- c("VQ1", "VQ2", "VQ3", "VQ4", "HQ1", "HQ2", "HQ3", "HQ4", "Vteam", "Hteam", "GameID")

count = 0

gamesscores[1, ] <- 99

GamesScores <- data.frame("VQ1"=99, "VQ2"=99, "VQ3"=99, "VQ4"=99, "HQ1"=99, "HQ2"=99, "HQ3"=99, "HQ4"=99, "Vteam"="VTEAM", "Hteam"="HTEAM", "GameID" = "GAMEID", stringsAsFactors = FALSE)

GamesScores

for (link in bscorelinks) {

count = count + 1

scorelink <- paste(link)

scorelink

gid <- str\_extract(scorelink,"[0-9]+[a-z]+")

scoresurl <- paste('https://www.pro-football-reference.com/boxscores/',gid,'.htm', sep="")

scorespg <- read\_html(scoresurl)

head(scorespg)

qscores <- html\_nodes(scorespg,'.center') %>% html\_text()

qscores

if('OT' %in% qscores == 'FALSE') {

vqscores <- qscores[2:5]

hqscores <- qscores[8:11]

} else {

vqscores <- qscores[2:5]

hqscores <- qscores[9:12]

}

vqscores <- as.numeric(vqscores)

hqscores <- as.numeric(hqscores)

vqscores

hqscores

links <- html\_nodes(scorespg, '.section\_anchor')

visiting\_link <- str\_subset(links, "vis\_starters\_link")

splitvisiting\_link <- strsplit(visiting\_link,"\"")

unlistvisiting\_link <- unlist(splitvisiting\_link)

sixthvisiting\_link <- unlistvisiting\_link[6]

unlistsixthvisiting\_link <- unlist(sixthvisiting\_link)

splitsixthvisiting\_link <- strsplit(unlistsixthvisiting\_link, " ")

unlistsplitsixthvisiting\_link <- unlist(splitsixthvisiting\_link)

visitingteam <-unlistsplitsixthvisiting\_link[1]

visitingteam

home\_link <- str\_subset(links, "home\_starters\_link")

splithome\_link <- strsplit(home\_link,"\"")

unlisthome\_link <- unlist(splithome\_link)

sixthhome\_link <- unlisthome\_link[6]

unlistsixthhome\_link <- unlist(sixthhome\_link)

splitsixthhome\_link <- strsplit(unlistsixthhome\_link, " ")

unlistsplitsixthhome\_link <- unlist(splitsixthhome\_link)

hometeam <-unlistsplitsixthhome\_link[1]

hometeam

gsgsrow <- c(vqscores, hqscores, visitingteam, hometeam, gid)

if (count < 2){

GScores <- rbind(GamesScores, gsgsrow)

}

else {

GScores <- rbind(GScores, gsgsrow)

}

}

GScores <- GScores[-1,]

write.csv(GScores,"NFL\_games\_[2020]season year].csv")

```

Once all of the necessary data was scraped, it was stored in a master database which consisted of more than 5,300 NFL games over the 20-season span.

Feature Engineering

Though the scraped data was robust, the need still remained for some feature engineering before modeling could begin. So I began by loading the master dataset into a Python pandas dataframe and dropping a few extraneous columns. Then I set about adding four new columns which would better facilitate the modeling process.

These “result” columns simply computed the difference between the quarterly scores of the two teams for each of the four quarters. For example, if the visiting team scored 10 points in the first quarter of the game and the home team scored 7 points in that period, the new corresponding column would display a 3 on that row. Of course, a negative value would signify that the home team outscored the visiting team during that quarter.

Next, another new column was added to record the number of points by which the winning team secured victory. The feature was computed by summing the four home team quarterly scores from the visiting quarterly scores. Again, a positive number represented a visiting team win, while a negative value did the same for a home team victory. (A value of zero meant that the two teams were tied after four quarters of regulation. Though an overtime period was played to determine a winner, these extra periods were not part of our experiment.)

Finally, the target column for my subsequent modeling was added to the dataset. Based on the penultimate column, the target column would display a code of ‘HW’ to indicate a home team victory, ‘VW’ to indicate a visiting team victory, or a ‘T’ to indicate a tie after regulation play ended.

Though I was finished adding columns, I still needed to encode the first four new columns in such a way so it would better standardize the values for our modeling process. Therefore, I replaced every numerical value in each of the four result columns with one of seven character codes:

+‘HHH’ if the home team scored at least 15 points more than the visiting team

+‘HH’ if the home team scored between 8 and 14 (inclusive) more points more than the visiting team

+‘H’ if the home team scored between 1 and 7 (inclusive) more points than the visiting team

+‘T’ if the two teams scored the same amount of points

+‘V’ if the visiting team scored between 1 and 7 (inclusive) more points than the home team

+‘VV’ if the visiting team scored between 8 and 14 (inclusive) more points than the home team, or

+‘VVV’ if the visiting team scored at least 15 points more than the home team.

Data encoding and grouping

These four result columns made up the first grouping of variables that was to be modeled against the target variable. But because the variables were strings, they needed to be transformed into integers so that the models could be fitted properly. So after subsetting those four result columns from the master dataframe, I then leveraged the OneHotEncoder transformer from Python’s scikit-learn package.

The result was a 28-column dataframe (four features times the seven code levels) filled with zeros and ones for each of the 5300+ observations. After the target column was concatenated back onto the encoded dataset, its values were also changed to an integer for modeling purposes: a 1 for each VW, a -1 for each HW, and a 0 for each T.

Since I didn’t want to restrict my experiment solely to individual quarters, the final steps before modeling started were to group the 28 dataframe columns into what I referred to as “clumps.” These clumps represented the variables related to the four quarters in every possible combination (15 in all). The clumps were labeled as follows:

|Clump #|Quarters in clump|

|0|Q1|

|1|Q2|

|2|Q3|

|3|Q4|

|4|Q1/Q2|

|5|Q2/Q3|

|6|Q3/Q4|

|7|Q1/Q3|

|8|Q1/Q4|

|9|Q2/Q4|

|10|Q1/Q2/Q3|

|11|Q1/Q3/Q4|

|12|Q1/Q2/Q4|

|13|Q2/Q3/Q4|

|14|Q1/Q2/Q3/Q4|

Then all of the clumps were put into a single list object which could be looped over during modeling.

Modeling

Quarterly data

I began the modeling process by choosing the estimators that I would be using^. They are:

1. LogisticRegression
2. KNeighborsClassifier
3. BaggingClassifier
4. RandomForestClassifier
5. ExtraTreesClassifier
6. DecisionTreeClassifier
7. AdaBoostClassifier
8. MultinomialNB
9. GradientBoostingClassifier

^-I also attempted to build neural networks on which to model the data, but the accuracy results were so abnormally low that I soon abandoned this line of experimentation.

I chose percentage accuracy as the metric by which I would evaluate each modeling pass. The modeling results of each estimator for each clump were recorded in a separate dataframe along with the clump number and estimator and then saved to the master database.

Weekly data

Next, the above data grouping and encoding process was repeated, but this time the Week feature was also included to see if it would have any impact on the accuracy scores. This feature consisted of 21 different integers, with the first 17 corresponding to the week number of the regular season during which each game was played (i.e., 1 was for the first week, 2 was for Week 2, etc.). The final four integers were expressed differently to denote playoff rounds: 100 for the first (wild-card) round, 200 for the quarterfinal (divisional) round, 300 for the semifinal (conference championship) round, and 400 for the championship (Super Bowl) game.

Then I subsetted the master dataset by season week and placed the resulting 21 dataframes into a single list object which could be looped over during modeling. Using a pair of nested loops (one for each clump and another for each week of the season), I reran all the estimators and recorded their accuracy scores. Before entering all 2800+ observations into the master database, I changed the four playoff week codes from their three-digit entries into character codes which better represented their rounds: WC for wild card round, DV for divisional round, CC for conference championship, and SB for Super Bowl.

Yearly data

Next, the above data grouping and encoding process was repeated, but this time the Year feature was included instead of the Week feature to see if it would have any impact on the accuracy scores. These four-digit numerical values represented the season year in which these games were played (from 2000 to 2019). It’s important to note that some of the regular season games and most playoff games are played in January and February of the following calendar year, but these contests are categorized under the year during which the season began.

Then I subsetted the master dataset by season year and placed the resulting 20 dataframes into a single list object which could be looped over during modeling. Using a pair of nested loops (one for each clump and another for each season year), I reran all the estimators and recorded their accuracy scores along with the clump number and season year. Then I entered all 2700 observations into the master database.

Teams data

The final leg of the modeling process involved incorporating all 32 teams into the working dataset and modeling them with the quarterly outcomes to see if they would have any impact on the accuracy scores. After a list object of all 32 teams was created, a loop was constructed so that each team would be entered as a key into a dictionary object along with its corresponding encoded dataframe (using the above encoding process). This loop was run twice – one for each team as the visiting team in the data set and the other for each team as the home team in the dataset. Because not all of the 28 encoded columns were included for each team (since their quarterly result values did not contain data which corresponded to these columns), a pass was made for each team (as visiting and home team) to add in these missing encoded columns to make the modeling process more consistent.

Then I used a trio of nested loops - one for each team, another for each clump, and a third for a list of the nine modeling estimators I used. This process was run once for visiting teams and again for home teams, and all accuracy scores were once again recorded along with the clump number and estimator. Then I entered all 8600+ observations into the master database.

When modeling was complete, I had made over 12,000 modeling passes, each with its own distinct accuracy score.

Imported Libraries

```

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier, AdaBoostClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import OneHotEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import MultinomialNB

```

Analysis

Then came the long-awaited but daunting challenge of analyzing all of the modeling data that had been assembled. I started this process the way all these analysis tasks should: by fleshing out a baseline model. This was accomplished by simply returning to the pre-feature engineered master dataset and noting the number of home wins, visitor wins, and ties in the dataset’s target category. The raw and normalized values looked like this:

|Category|Code|Count|%age|

|Home wins|HW|2855|53.63%|

|Visitor wins|VW|2136|40.12%|

|Ties|T|333|6.25%|

A baseline model is intended to represent what the accuracy percentage would be if a person were to randomly guess the outcome of each game; in this case, he or she would only have to guess “home win” each time to be correct 53.63% of the time. That figure is what I used for my baseline model.

(One interesting insight from these numbers: between 2000 and 2019, about one out of every 16 NFL games went into overtime – or on average, one in a typical NFL week during which all 32 teams played.)

Quarterly data

I began my data analysis with the first pass of modeling results involving only the clumps of quarterly outcomes. After assessing the maximum and minimum accuracy scores for each clump, I was surprised to see that for all 14 clumps, the same estimator produced the maximum and minimum score: RandomForestClassifier and AdaBoost classifier, respectively. I then took the mean accuracy scores for all 14 clumps and put the results into a line graph [/images/AccuracyGraph1.png] along with a graph of the maximum accuracy scores by clump. This metric indicated that, given the clump of variables measured, the model predicts the accurate final outcome of the NFL game that percentage of the time (with all other variables being equal).

As expected, the four single-quarter clumps produced the lowest accuracy scores of between 60% and 63%, while the next six clumps – which contained pairs of quarterly outcomes – posted higher accuracy scores of around 67%. Higher still were the accuracy scores of the trios of quarterly outcomes in the next four clumps, which came in between 71% and 73%. Finally, the clump which took into account all four quarterly outcomes bested all of the other clumps with an accuracy score of 81%. This pattern follows the maxim that the more variables you incorporate into your model, the better your accuracy scores will be (to a point, of course).

Weekly data

I followed a similar analysis process for the modeling results that included the week number in the groups of independent variables. Initially, I noted similar results: the maximum accuracy scores by clump were all produced by a RandomForestClassifier, while the minimum counterparts all came as the result of modeling with an AdaBoost Classifier.

For this part of the analysis, I created a line graph [‘/images/EvenBetterWeeklyGraph.png’] with each of the first four (individual quarter) clumps graphed by week of the season and accuracy score. I also included a horizontal line at 53.63% to denote the baseline model, as well as a vertical line between week 17 and 18 to separate regular season games from those played in the postseason.

Tracking the progress of the four clumps on the graph didn’t yield any particular patterns. But I did notice something interesting when I counted the points on the graph which fell below the baseline model.

To the left of the vertical line (i.e., the weeks of the NFL’s regular season), only two of the 68 quarterly clumps (or 2.9% of them) posted an accuracy score lower than the baseline score: the “fourth-quarter” clump in week 4 and the “first quarter” clump in week 6. But looking at the right side of the vertical line (i.e., the four weeks of playoff games in the NFL), I noted nine different quarterly clumps below the baseline accuracy score out of the 16 that were on the graph (yes, 56.25% of them). They were: the “fourth-quarter” clump in the Wild Card week, the “second quarter” and “third quarter” clumps in the Divisional Round, all four quarterly clumps in the Conference Championship Round, and the “first quarter” and “second quarter” clumps during Super Bowl weekend.

Therefore, our models with only single-quarter outcomes as independent variables turned out to be much better predictors of the winners of regular season NFL games than they did of playoff contests. In a sense, this may not be too startling of a revelation, since we might assume that teams in the postseason play differently than they do in the regular season because of the possibility of elimination that comes with a defeat.

Yearly data

Next, I followed a similar analysis process for the modeling results that included the season year in the groups of independent variables instead of the week of the season. First, I simply looked at the mean accuracy scores for each season year in the experiment. Three of the models did not beat the baseline model accuracy score of 53.63%: those for season years 2012 (52.43%), 2013 (49.75%), and 2018 (52.74%). The two highest accuracy scores were associated with the data from 2006 (74.33%) and 2010 (72.47%). So during those three years in the second decade of the 21st century, the outcomes of NFL games were more difficult to predict with these models; while those two in the previous decade were much easier to predict.

As with the weekly data, I created a line graph [‘/images/YearlyGraph.png’] with each of the first four (individual quarter) clumps graphed by season year and accuracy score, and also included a horizontal line at 53.63% to denote the baseline model. Again, tracking the progress of the four clumps on this graph didn’t yield any distinct patterns, so I focused more on the graph’s maxima and minima.

Out of the 80 data points tracked on the graph, a grand total of six single-quarter clumps (or 7.5%) posted accuracy scores below the baseline; the three worst performing were the “fourth quarter” clump in 2005 (46.1%), the “third quarter” clump in 2018 (46.2%), and the “first quarter” clump in 2013 (49.8%). On the other hand, two of the three highest accuracy scores were posted by quarterly clumps in the 2006 season: the “second quarter” clump (77.5%) and the “third quarter” clump (74.1%). Sandwiched between those two models was the “second quarter” clump in 2016 with an accuracy score of 74.9%.

Team data

Perhaps the most sought-after insights came from analyzing the team data results. Prior to undertaking this task, it was necessary for me to put a new column on the results dataset. The ‘HV” column denoted whether the accuracy score recorded with the observed team came when the team was playing at home or on the road.

Like the quarterly data and the weekly data, the RandomForestClassifier was responsible for all the maximum accuracy scores by team and clump, while the AdaBoost Classifier was associated with all of the minimum accuracy scores in this grouping. Also, as part of the analysis process, I created a function which produced the mean accuracy score by clump for an inputted team and home/visitor designation.

Then I constructed a pair of nested loops which took in each clump number and NFL team to create a dataframe with the estimator and maximum accuracy score by team for each clump. (I also ran these loops to get the data associated with the minimum accuracy scores.) Unlike the homogeneous results from the previous data groupings, the estimators for the teams’ maximum accuracy scores varied widely and encompassed all of the estimators that were modeled.

As with the other data groupings, I chose to focus primarily on the single-quarter clumps. First, I picked out the highest and lowest overall team accuracy scores for each “quarterly” clump:

|Quarter of Clump|Highest/Lowest|Team|Accuracy Score|Home/Visitor|

|1|Highest|Browns|80.49%|V|

|1|Lowest|Seahawks|55.81%|H|

|2|Highest|Saints|83.72%|H|

|2|Lowest|Titans|53.49%|V|

|3|Highest|Browns|80.49%|V|

|3|Lowest|Chiefs|51.22%^|V|

|4|Highest|Ravens|78.57%|H|

|4|Lowest|Eagles|54.55%|H|

^-below baseline model

Keep in mind that these accuracy scores should not be interpreted to identify any aspect of a team’s overall ability or success rate. Rather, the higher the accuracy score, the more predictable that the ultimate outcome of the game will be influenced by the result of the given quarter. Conversely, lower accuracy scores indicate less predictability of whether a team wins or loses a game based on their performance in the corresponding quarter.

To satisfy my own interests, I also took note of the highest and lowest overall team accuracy scores based on their performance in the first half of a game (the “first quarter” and “second quarter” clumps), the second half of a game (the “third quarter” and “fourth quarter” clumps), and the entire list of quarterly outcome clumps. Here’s what I discovered:

|Part of Game|Highest/Lowest|Team|Accuracy Score|Home/Visitor|

|1st half|Highest|Eagles|84.09%|H|

|1st Half|Lowest|Titans|62.79%|V|

|2nd half|Highest|Texans|81.08%|V|

|2nd half|Lowest|Cardinals|63.41%||V|

|All combos|Highest|Jaguars|80.49%|V|

|All combos|Lowest|Chiefs|74.42%|V|

Finally, I took the average of all of the maximum accuracy scores associated with every team (as home as visiting teams). I discovered that by this metric, the highest “mean-max” accuracy score was 81.21% turned in by the Lions, while the lowest such score was 66.99%, courtesy of the Titans.

Streamlit App

In an effort to better illustrate my experiment and modeling process, I constructed a web application using an open-source Python library called Streamlit. Since the RandomForestClassifier estimator seemed to produce the best results, I picked the model which used that estimator and set all combinations of the quarterly clumps as independent variables so as to maximize the app’s accuracy. On average, this model gets the NFL game outcome correct about 87% of the time.

I set up the app to allow the user to move a slider bar along a range of numbers that corresponds to the number of rows in my master dataset. When a number is selected and the button labeled ‘Generate’ is pressed, the app picks out that row number from the dataset and displays information about the real-world game that took place, including the home and visiting teams, the week and season during which the game took place, which team won the game, and how large the winning margin was. Then the model takes the information and returns its prediction of which team would win and compares it against the actual outcome of the game.

Conclusions

I came out of this experiment with the confirmation of what I felt going into it: that no matter how much data I may have or how meticulous my modeling process may be, it is quite difficult to predict the outcomes of NFL games. Applying this process to high school or college football games might produce better results, but that would likely be a function of the disparity of talent in a significant number of games. For example, if the previous year’s champion played a team that only won a few games the previous year, that outcome might be easy to predict – especially when you have the quarterly outcomes to model on.). In contrast, the level of parity in the NFL makes it that much harder to produce accurate predictions based on quarterly outcomes.

To be sure, none of the models produced and tested in this experiment accounted for numerous real-world (and arguably more relevant) factors in the outcome of an NFL contest. Some of these variables might include injuries, player talent, coaching ability, weather, or days of rest since the last game. Moreover, it’s quite possible that even the models which produced more favorable results might not be effective going forward. That’s because the NFL is changing its scheduling format beginning with the upcoming season. In 2021, all NFL teams will play 17 regular-season games in 18 weeks instead of last season’s 16 games over a 17-week span. Though these models would get an additional 16 games’ worth of data per season, the assumptions which underpinned them might change drastically and/or reduce the level of accuracy scores that they could produce.

Future Recommendations

Nevertheless, I would consider conducting future experiments of this type in order to discover models and variables that are more correlative with the outcomes of NFL games. Obviously, I would endeavor to incorporate more variables than just quarterly outcomes, especially those which were already in the original dataset (day of week, game time, date, and opponent). Similarly, I might also construct different groupings of the variables I did use, such as a “clump” of season weeks or consecutive years. But I might also pay more attention to conditional probabilities (such as whether a team scores first in a given quarter) to see if they affect the accuracy of any modeling I might undertake.

In addition, I have the option of changing up the levels represented by the winner codes. For example, instead of having seven winner codes, I could increase or decrease that numberwhile also changing the cutoff margins between these levels (say, designating a new level for every three points of scoring margin instead of seven). A more direct approach might be to abandon scoring levels altogether and simply incorporate numerical scoring margins for each quarter.

It should be noted that all of the estimators I utilized were tested using their default hyperparameters. But these hyperparameters can be tuned with the goal of pushing the accuracy scores of their estimators a little bit higher, so any future experiments should allocate time for such tuning. And if I want to take yet another step, I could try and devise models which could predict that actual numerical scores of each quarter. While undoubtedly difficult, any successes could have practical impacts on the growing world of sports gambling.

As with many modeling experiments more questions than answers were generated. But thanks to my lifelong love of football, I’m heartened by the challenge and fascinated by the complexity of the scoring patterns of my favorite sport.

Thanks

I would like to extend a heartfelt “thank you!” to my instructors at General Assembly for their assistance with this project. Without Jeff Hale (no relation to author Philip Hale), Jacob Koehler, and Eric Bayless, I would not have been able to coax this project over the finish line. I’m also grateful for the skills they taught me and the knowledge they imparted to me which also aided me throughout my experiment.