Exploring Shot Difficulty on the PGA Tour Using Generalized Additive Modeling and Hierarchical Effects

---

Benedict Brady

Exploring Shot Difficulty on the PGA Tour Using
Generalized Additive Modeling and Hierarchical Effects

A thesis presented by
Benedict Brady
In the Department of Statistics

In partial fulfillment of the requirements
for the degree of Bachelor of Arts with Honors in the
subject of Statistics

Harvard College
Cambridge, Massachusetts
April 1, 2019

Abstract

Modeling shot difficulty in golf is a uniquely challenging problem. Although the data is well structured, it is uncommon for two shots to be taken from the same place. In this paper, I investigate improvements to existing models by using hierarchical structures to account for the bias from the course and player effects. I also employ smoothing splines in generalized additive models to more accurately depict the nonlinear relationship between distance metrics and shot difficulty. Among other things, by using these models, I am able to rank course and hole difficulty, while providing definitive evidence that shot difficulty varies as a function of time of day. I also propose a new feature, that I term effective green, which incorporates previously unused spatial information into my analysis. Finally, I introduce a new method for course difficulty visualization by imputing the spatial attributes of many out-of-sample points.

Acknowledgements

I could not have written this thesis without the guidance of my advisor, Kevin Rader. I spent a large part of the past four years picking his brain about sports analytics, and no one has offered a more consistently insightful and supportive perspective than Kevin.

I am grateful to the Harvard Sports Analysis Collective, which both solidified my love of statistics and my love for sports. I would not be as interested in the field of golf analytics if not for the pioneering work of Mark Broadie, nor would I have been as passionate about the game if not for the countless hours I spent watching Phil Mickelson and Tiger Woods with my father and uncles as a child.

My friends and roommates offered invaluable support, both academic and otherwise, during the most difficult parts of this process. A special thanks to Aren Rendell who stayed up until all hours of the night to talk about the finer points of the relationship between time of day and dew on the green.

My parents and siblings are the reason that I do everything that I do today. They have supported me not only on this thesis, but on every endeavor my entire life.

Contents

1 Introduction

1.1 Motivation

Wandering the halls of a sports statistics conference, it is not uncommon to hear the refrain that sports are the perfect natural experiment. The initial conditions are generally the same, and the game is run over and over again throughout the season. These conditions are easy to observe in a sport like basketball. Golden State Warriors All-Star Steph Curry will take close to 1000 threes from the top of the key throughout his career, and while the defense and exact location might change slightly, the hoop size, distance, weather, and location on the court will all remain functionally the same. These conditions allow a researcher to study, in a rigorous way, the impact of small changes in defense. While the quality and quantity of this type of analysis has increased dramatically over the past 10 years, golf has notably escaped extensive analysis. This is because golf is consistently played on different courses and under different conditions.

There are a host of other explanations for why golf statistics have not progressed as quickly as other American sports: lower viewership, a less statistically savvy fan base, fewer incentives for the league and competitors to publish or encourage research[1]. But much of the absence of robust golf statistics can be attributed to the fact that it is quite difficult to answer the

[1] In other sports, the teams commission research to gain competitive advantages. Golf, however, is an individual sport, removing some of the possible infrastructure for analytics research.

simple question "how hard is a 100-yard chip shot on hole 14 of Augusta National for Jordan Spieth?"

In sports statistics there are two main steps of analysis: the first is understanding the environment in which the game is played, and the second is understanding how the players interact with the environment. Due to the difficulty of the first step caused by variation in courses, holes, and weather, the second step of analysis in golf has always been incomplete. To solve this problem, golf statisticians have made valuable attempts to standardize certain metrics over courses in order to perform player level analysis.

In the early days of golf, player strength was generally measured solely on how well an individual could score relative to par [2]. While this does not account for the difficulty of the course, it is a reasonably good measure of how well a player performed in a given tournament relative to the rest of the field. The obvious drawback of this metric is that due to inconsistencies in par ratings and the lack of any rigorous definition, strokes relative to par do not have strong predictive power tournament-to-tournament. This metric also does not provide insights into how strongly each player plays from different parts of the course. Metrics like putts per hole and greens in regulation[3] were developed to provide some insight into player strength on different areas of the course. With the development of shot tracking, the PGA Tour began to be able to record driving distance to measure performance off the tee[4].

In the early 2000s, the PGA Tour started a massive operation, using lasers and recruiting thousands of volunteers to start tracking stroke-level data

[2]Par is the expected score of a scratch golfer on a given hole.
[3]Greens in regulation is defined as getting to the green in three strokes on a par five, two strokes on a par four, and one strokes on a par three.
[4]These stats can be found on the PGA Tour website here: `https://www.pgatour.com/stats.html`.

more precisely[PT]. They were able to pinpoint the location of both the beginning and the end of a given stroke within a few inches of accuracy. This data was then vetted for errors live at the tournament by a team of data specialists and published after every tournament to individuals with a license to access the data. While this allowed the Tour to publish more precise shot metrics, the first notable academic study using this data wasn't published until 2011 by Fearing et al. [Fea+11] in the *Journal of Quantitative Analysis in Sports*. Summarizing the state of golf analytics, they offer the following perspective:

> Other popular sports, such as baseball, basketball, and American football, have developed loyal followings of fans who pore over statistics on a regular basis. But, statistical analysis of the game of golf has lagged...Unfortunately, golf analysis currently suffers from a few significant drawbacks. The first is that data officially reported by the PGA TOUR is limited to a small number of aggregate statistics including Drive Distance, Drive Accuracy, Greens in Regulation, and Putting Average. The second issue is that the statistics that are reported do not do a particularly good job of differentiating golfer performance. For example, there is no way to tell whether a golfer's low Putting Average is due to exceptional putting performance or equally impressive performance on approach shots. The third issue is that the reported individual statistics are heavily biased by the difficulty of the courses the golfer has played, and professional golfers play in different sets of tournaments over the course of the year. [Fea+11]

Fearing et al. [Fea+11] focuses primarily on predicting putting ability. Using the distance from the hole and a generalized linear model approach, they attempt to predict both the probability of a putt being successful and the distance left to putt, conditional on the putt being missed. They use both of these predictions along with a Markov model to attempt to simulate the remaining number of strokes on a given hole. These models include both player effects and course effects to control for bias. With this model, Fearing et al. [Fea+11] were able to remove some of the bias in putting statistics

that came from players with better approach shots appearing to be stronger putters purely by ending up with easier putts.

In 2012, Mark Broadie released easily the most influential and widely cited paper in golf to this day [Bro12], which he eventually turned into a book named *Every Shot Counts* [Bro14]. Broadie built off a paper he had written in 2008 doing some preliminary work to analyze shot accuracy and relative shot type importance[5] [Bro08]. Broadie [Bro12] marked the first widescale attempt to estimate shot probability using distance all across the PGA Tour, and he used this strategy to determine the difficulty of specific shots. Broadie relied on estimating six different types of golf shots: tee, fairway, rough, sand, recovery, and green. The green model was fit using an estimate for one putt probability and an estimate for three putt probability, and combining these things to determine the remaining number of strokes. For the other five shot types, he fit a polynomial to estimate shot difficulty based on distance.

Broadie then continued on to model course hole difficulty and player difficulty at a macro level, using the results from his distance model to infer player strength. These models are fit iteratively, and there is also no time component to player skill level; it is modeled statically for a given sample. The release of this paper led the PGA Tour to adopt a metric called "Strokes Gained[PT]," which compares a given shot to an expected baseline to determine how much value it added or subtracted from a player's total score over the 18 holes. A version of this metric is now included in the official data released.

While incredibly innovative for the time, Broadie's model fell short in a few important ways. First and foremost, besides small course adjustments made

[5]Broadie has led the modern school of thought that putting ability is much more variable than other stroke types, meaning that it is less important to consistent success on the PGA Tour than driving ability or iron play.

after the distance model was fit, this model did not incorporate any differences in difficulty for shots of the same length and surface type. Especially on the green, every golf enthusiast understands that some 20-foot putts are straight and quite simple, while others have lots of break or have a downhill slope that make the touch much more challenging. Second, Broadie's model used polynomial functions to fit the difficult equation with respect to distance. Since the release of this paper, much more sophisticated splining techniques have become computationally feasible and readily available that can manage overfitting and nonlinear relationships in a more rigorous manner. Finally, this model fit its course and player effects iteratively (probably due to computational difficulty) which meant that a player's strength was not taken into account when trying to determine the difficulty of an 30-yard pitch shot he is taking. However, the main insight of this paper still stands today: the largest determinant of the difficulty of a shot in golf is the location.

Around the same time as the release of Broadie [Bro12], a few researchers attempted to build more complicated frameworks to describe the location of a shot than purely distance from the hole and surface type. In 2013, Stöckl et al. [Stö+11] propose the ISOPAR[6] method which used smoothing splines to divide up the course into contours based on shot difficulty. This model failed to take into account strength of field or distance/location type metrics, and as a result provides some additional insight to the spatial structure of the course without incorporating what are now known to be critical features in shot difficulty prediction.

Following this, in 2013 Yousefi and Swartz [YS13] proposed an improvement to existing putting metrics by modeling putts as a truncated poisson and splitting up the putting green into 8 quadrants to perform a Bayesian analysis.

[6]Named after Isobar contours in weather plots.

This model allowed Yousefi to capture some location effects on the green, but while it might increase prediction accuracy, an arbitrary eight quandrant structure does not provide any information as to what makes a putt difficult. Additionally, this model did not consider the difficulties arising from field strength or the information that can be learned from strokes that are not putts. Still, while this model is not widely cited, it is likely the most comprehensive structure of analysis for putting available today.

Finally, in 2017 Levin [Lev17] made the first broad attempt to combine distance and location effects into one player ranking. Levin considered location type (green, fairway, etc.), distance, course, round and day indicators, and an engineered featured called "Green to Work With[7]" and used a gradient boosting machine to classify shot difficulty. He then took every stroke on the tour and paired it with other shots taken on the same day in the same location and used network effects to attempt to infer player strength. Levin selected a gradient boosting machine because of its accuracy, but also because it does not require the modeler to input any structure, instead inferring the interaction effects from the data[8]. This approach has the benefit of requiring less tinkering with the initial setup, but also makes interpretation of many effects much more difficult and less rigorous. Levin's shot difficulty metric considers a broader feature set than previous research, and except for player effects, Levin [Lev17] estimated all of the features and interactions simultaneously, which is a much more rigorous practice.

As shown by this progression of golf research, beyond the idea that distance and location surface are important, there is not much agreement as to what the best modeling techniques are and what additional features should be

[7]If a straight line is drawn between the shot location and the hole, this is the section of green that the line crosses. I discuss it in greater depth in later sections.
[8]Levin told me this in a phone interview I conducted with him on March 14, 2019.

incorporated. Looking at the PGA Tour ShotLink data[PT], there are a multitude of features that do not appear in any papers. Additionally, no tools exist to visualize the difficulty of varying locations on a given hole.

Looking at the landscape of golf discussion, I determined four areas where there is room for improvement in the field of golf analytics:

1. There are is significant untapped potential in inferring location attributes about the 2D structure of the golf course.

2. Whether it be time of day or a given distance metric, functionally no continuous variables in golf vary linearly with shot difficulty.

3. Golf strokes fall into a multitude of non-nested hierarchical structures. Every hole is a subset of a given course, a player's ability in the bunker is a subset of his ability as a golfer overall, and all shots taken on the same day are generally influenced by the same weather patterns.

4. Building off the point above, golf is extremely susceptible to selection bias. Strong players all play the same tournaments, or better golfers are more likely to end up with shorter putts on the green. The implication of this was that inference done on models that iteratively fit correlated variables, such as course and player, were likely to have strong underlying directional biases.

1.2 Problem Statement

When considering shot difficulty on the PGA Tour, what is the most rigorous way to model both the hierarchical structures in the data and the non-linearity of the continuous features, what additional spatial features can be added to accurately describe a shots location, and what does this tell us about what makes a shot difficult on the PGA Tour?

1.3 Terminology

Golf rounds are scored relative to the "expected" number of strokes, or par. Par is given for each hole, and the net score for a given round is the golfer's total number of strokes minus par. A golf course consists of 18 holes, each of which is played twice by every golfer (normally on Thursday and Friday) and then two more times by the people who make the cutoff (normally on Saturday and Sunday).

On a given golf hole, there are a few different surfaces on which a ball can rest. The ball starts out in the tee box on every hole, a pre-specified starting box that shifts slightly between rounds. The hole is located somewhere on the green, a hard surface that has short grass and accounts for roughly 40 percent of all ball locations. In between the green and tee there is a fairway, a cut of ground up the middle with well-trimmed grass. Off to the sides there is the intermediate and full rough that have much thicker grass and penalize players for missing the fairway. In between the fairway and the green there is a lip of grass with height between the green and the fairway that is called the fringe. In some situations golfers play this like the green, using a putter,

and in other cases they use a wedge and play it like the fairway. Finally, there are obstacles such as water and sand traps that can set players back much further because they are difficult to hit out of. A small percentage of shots do not fall into any of these categories and are logged as "other."

Golfers can be penalized a stroke or two for a bad shot or for breaking the rules. There are also other shot types such as provisional or drops that will be ignored for the sake of this analysis.

A drive is a shot taken off the tee, an approach shot is a shot not off the tee that is approaching the green, and a putt is a shot on the green. Additional terminology will be explained throughout the course of this paper.

2 Data

2.1 Raw Data Format

This data came from the Shotlink Intelligence program from the PGA Tour[PT]. This data set represents roughly 20 million strokes on the PGA Tour since 2000. This analysis focused primarily on the data from 2018 due to computational limitations and the added complexity that came from time series analysis. In a raw format, the data came with certain variables about each stroke. Most variables fall into a few groups that will be explained below.

2.1.1 Categorical Variables

The first group is broad categorical variables. To explain this data, I will use a 2018 shot fromPGA Tour Professional golfer Phil Mickelson as an example, as seen in Figure 2.1. This shot is from TPC Sawgrass[1], Hole 5, Round 1. The image of this course is generated using a KNN procedure that will be outlined in a later section. The four defining features of this shot included in the data are `Course`, `Hole`, `Round`, and `Player`, of which this shot clearly has the values TPC Sawgrass, 5, 1, and Phil Mickelson respectively. Slightly more granularly, this data includes four more categorical variables that were used for prediction: `Date`, `Location.Scorer`, `Slope`, and `Elevation`. `Location.Scorer` is the location type recorded for this shot, e.g. Green, Fairway, etc. `Slope` is slope of the ground on which the shot is taken, and

[1]Home of THE PLAYERS CHAMPIONSHIP.

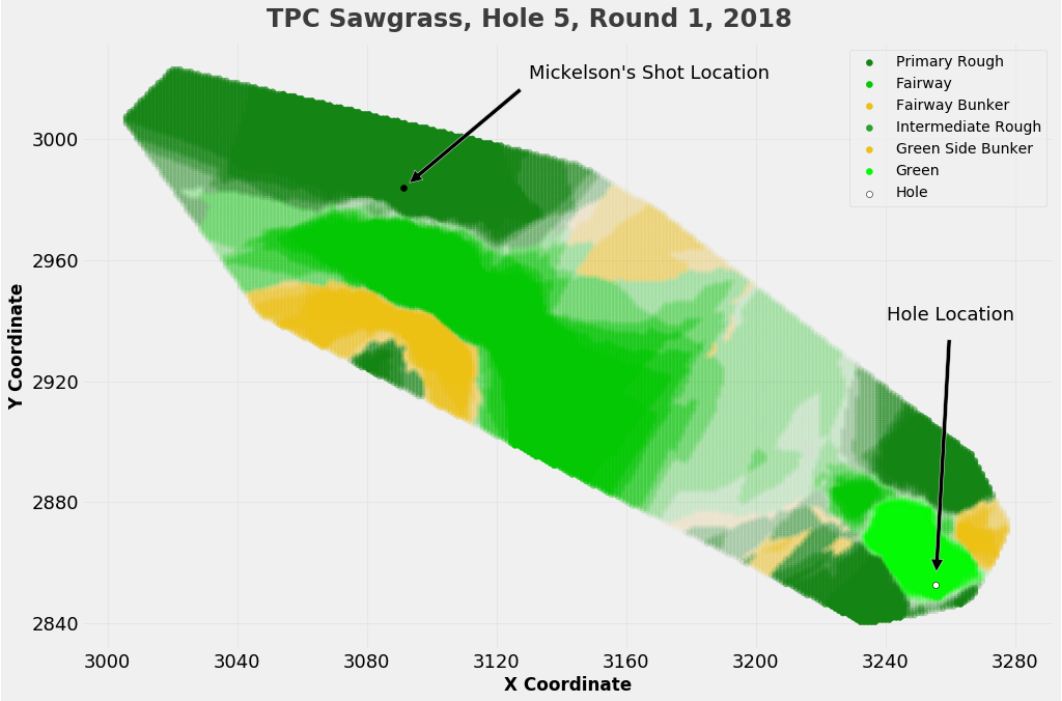

Figure 2.1: Phil Mickelson's second shot on TPC Sawgrass, Hole 5, Round 1, plotted on top of a course estimate made by my K Nearest Neighbors algorithm.

`Elevation` is elevation of the player relative to the ball. For this shot, these attributes took on the values 5/10/2018, Primary Rough, Uphill, and Above Ball respectively.

2.1.2 Continuous Variables

The next classification of variables are continuous variables, normally pertaining to location attributes about the ball. There were 7 variables that encoded the location information about the ball: `Distance`, `Shot.Length`, `Distance.to.Center`, `Distance.to.Edge`, `X`, `Y`, and `Z`. The remaining continuous variable was `Time`, the local time that the shot was taken. Tracing back to Mickelson's shot at TPC Sawgrass, Figure 2.2 shows the four distances measured with respect to his shot, as well as the XY grid that determine the shot's coordinates. The variable `Distance` measures the distance to the hole, `Shot.Length` measures the distance that his shot traveled,

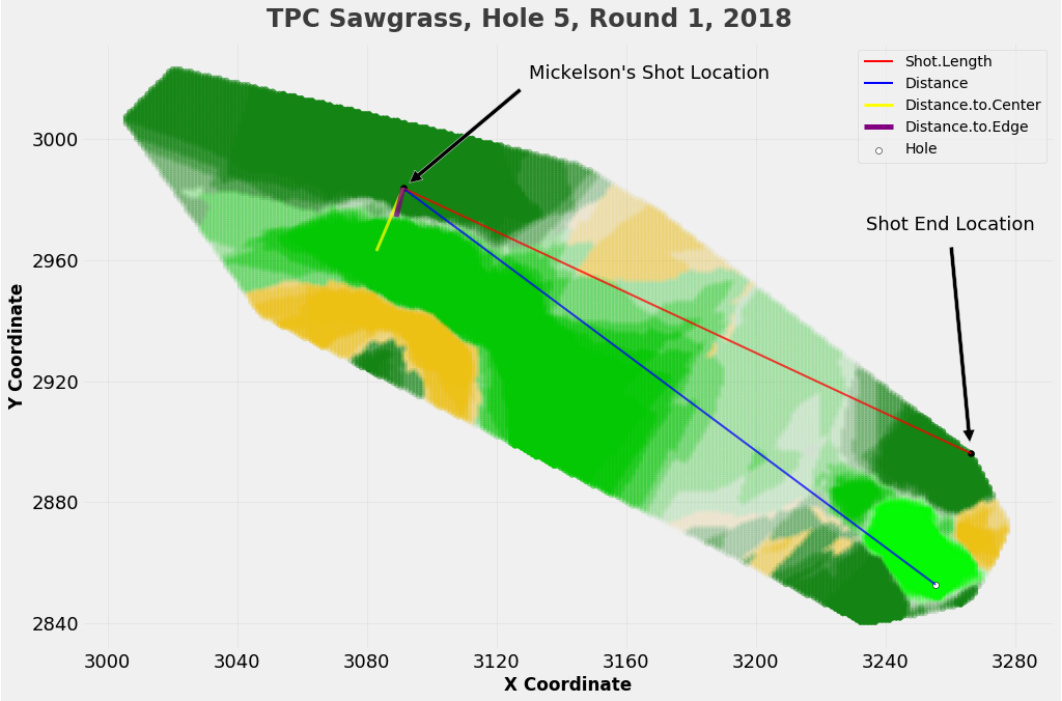

Figure 2.2: A diagram of `Shot.Length`, `Distance`, `Distance.to.Center`, and `Distance.to.Edge` for Phil Mickelson's second shot on TPC Sawgrass, Hole 5, Round 1.

`Distance.to.Center` measures the distance from his ball to the center of the fairway, and `Distance.to.Edge` measures his location to the edge of the fairway. As we can see, this is quite a bad shot.

In reality, ShotLink provided `X`, `Y`, `Distance.to.Center` , and `Distance.to.Edge` for the end location of the shot, not the beginning. In a later section I will discuss how this was realigned to the current shot to help eliminate forward looking features. Finally, this data also contained a `Z` variable to plot the shot location in a third dimension.

2.1.3 Bookkeeping Variables

The final class of variables that I used from the raw data were "bookkeeping" variables. These variables are best thought of as variables that helped keep the

data coherent for cleaning or help with model evaluation, but were not used for prediction, feature engineering, or visualization in any meaningful way. These variables include `Shot`, `Hole.Score`, `Shot.Type`, `Number.of.Strokes`, and some extraneous `Strokes.Gained` features. `Shot` and `Hole.Score` are self explanatory and formed the basis of the dependent variable in this analysis after some cleaning. `Shot.Type` and `Number.of.Strokes` determine the type of stroke as it relates to penalties or provisionals, and the number of strokes assessed on the penalty, respectively. `Strokes.Gained` is a metric developed by Mark Broadie that will be used as a comparison for the models built in this paper [Bro12].

2.1.4 Variable Exclusions

It is important to note that this is not the full feature set given by ShotLink, and a brief explanation will be given of the decision to remove the remaining features. One group was forward looking versions of already existing variables, such as To.Location (the location the ball lands) and Distance to the Hole after the Shot. The next category is redundant variables that will help with interpretation down the line, such as Course Name, Player Name and Tournament Name. The other group was location variables that have some combination of missingness and or interpretability issues. I excluded these mostly for sake of simplicity.

The first of these excluded location variables is Lie. Lie is a categorical variable that has a high level of missingness and a skewed distribution. Instead of assigning the missing variables an arbitrary category or creating a new category for missing data, this variable seemed to be relatively uninformative and was therefore dropped for simplicity. The next variable was Left.Right. This variable is intended to be a flag determining the side of the course,

but it did not seem to be very accurate after a few attempts to graph it. Additionally, it has substantial missingness that cannot be arbitrarily slotted into left or right. Finally, the laser tagging system had engineered a feature called Location.Enhanced that is supposedly a more accurate and granular version of Location.Scorer. While this location estimate seemed to be precise, over a third of the locations were logged as unmapped. Without proper coverage, Location.Scorer was used for simplicity.

2.2 Cleaning

The raw data provided by the PGA Tour had three main issues. First, the data had some alignment problems and forward-looking features. For example, the `X`, `Y`, `Z` coordinates provided were recorded after the shot is finished. The second major issue was that this data has selective missingness for certain columns. Finally, due to the overdefinition of the feature set, it was possible to highlight some existing inconsistencies[2]. In the next five sections I will discuss the strategies used to deal with these problems.

The 2018 PGA data starts out with 1,181,633 observations. As we progress through the data cleaning section, I will note the data loss from each transformation, and the total remaining number of observations.

2.2.1 Missing Hole Score

The first step in cleaning this data was to coerce the columns into the correct data formats. Out of the full data set, 4,147 observations were missing a

[2]When seven location attributes are measured for a variable that exists in at most three dimensions, there are often a lot of ways to find inconsistent data.

score for the hole. Additionally, there were no observations that came from the same player playing the hole in a given round, allowing for imputation; when a hole was missing a score, the score was missing for every stroke. These observations were all restricted to two courses, TPC Louisiana and Austin Country Club. While it was possible to count up the number of strokes recorded and impute that value, I instead assumed that this was missing for a specific reason and dropped these observations. At the end of this process, the remaining data had 1,177,486 observations.

2.2.2 Data Alignment

The second issue to note with this data was that observations have instances of a higher shot number than the score on the given hole. In Levin [Lev17], this data was dropped, "In order to maintain the integrity of the data, all player-holes for which the number of shots in the data did not match the recorded score of the player on the hole were dropped." What this paper failed to take into account was that the data is numbered in such a way that provisional shots and drops were labeled as iterative shots but did not contribute to the stroke count. Additionally, there are penalties issued that count as one stroke instance but were assessed two strokes of penalty. The stroke count needed to be adjusted accordingly to deal with this inconsistency.

Every observation had a `Shot.Type` attribute to explain the specifics of the shot with regard to penalties, provisionals, etc. The breakdown of shot types can be seen in Table 2.1 and number of strokes assessed in Table 2.2. From these charts, I was able to piece together a picture of when these shots occur. The union of the set of dropped shots and provisional shots makes up the set of shots where zero strokes are assessed. For sake of simplicity and coherence,

Shot.Type	Count
Stroke	1161079
Drop	8039
Penalty	6418
Conceded	1720
Provisional	230

Table 2.1: Frequency of stroke, dropped shots, penalty shots, conceded shots, and provisional shots in the 2018 PGA Tour ShotLink data.

Number.of.Strokes	Count
0	8269
1	1169208
2	9

Table 2.2: Frequency of observations assigned zero, one, and two strokes in the 2018 PGA Tour ShotLink data.

I dropped all of these shots and renumbered the shot sequences assuming that these shots did not exist.

The second subset of problematic shots were conceded shots. In match play, points are only assigned based on who finishes the hole using fewer strokes. That means that if Player 1 finishes hole in 4 strokes and Player 2 takes longer, then it will often be written that Player 2 finished the hole in 5 strokes with the last event being a Conceded shot. Since Player 2 often would have finished in more than 5 shots, this data will taint the sample. Additionally, throwing out the conceded shot sequences will skew the data toward the player who had a better performance, introducing some unpredictable selection bias. Since there was only one course in 2018 with match play (Austin Country Club), I dropped all of the observations from this course for the sake of this analysis. Analyzing the unique strategy of match play will be left up to a future researcher.

Finally, the last shot type to deal with was penalty shots. As far as this analysis is concerned, there are two types of penalties. Those that were assigned one stroke, and those that were assigned two strokes. Due to the infrequent nature of two strokes penalties (only 9 in the entire data set), every shot sequence with a two stroke penalty was removed. It is unlikely that these shots had any noticeable influence on coefficient estimates.

Second, there were many instances of one stroke penalties. Intuitively, I would not like to attempt to predict the existence of penalties in any direct way, but instead aim for the following effect: if a golfer is stuck in a location where there is a distinct probability they will be assessed a penalty stroke down the line, this location should be rated as more difficult. To achieve this effect, I removed the penalty shots from the data, but left the shot sequence the same. This way, the data could skip from the second stroke to the fourth stroke because a penalty was assessed at the initial location. The location of the second stroke was then two more `Strokes.Remaining` than the location of the fourth shot, so it was assumed to be a more difficult location.

Once all the shot numbers were adjusted and the data set is boiled down to only real strokes, the `Strokes.Remaining` variable was calculated as `Hole.Score` − `Shot`. This was the dependent variable for the majority of this analysis.

Next, the forward-looking bias in the columns `Shot.Length`, `X`, `Y`, `Z`, `Distance.to.Center` and `Distance.to.Edge` needed to be considered. All six of these features were measured by a laser after the ball came to a rest. Instead, it was much more intuitive to have these variables from the location that the stroke is taken. To solve this problem, I created an algorithm that identified the previous shot and transposed the values in these columns to the next shot.

When dealing with spatial variables[3] , the tee shots were unique because they had roughly similar values for every shot on a given `Hole` and `Round`. Because of this , all of these spatial variables were set to N/A and not used for the majority of this analysis.

The exclusion of penalty shots removes 26,586 observations, leaving a total of 1,150,900.

2.2.3 Categorical Features

The raw data had essentially eight categorical variables that could potentially be used for prediction and inference: `Date`, `Course`, `Hole`, `Round`, `Player`, `Location.Scorer`, `Slope`, and `Elevation`. The first five columns had no missingness or irregular values, and were integral to the structure of the data. However, the remaining three categorical features required some cleaning. While these variables were never reported as missing, there were a few classifications that needed to be investigated. Figure 2.3 shows the breakdown of the `Location.Scorer` attribute. Here, I decided to group together the Other and Unknowns. Both represented a small portion of the data and they seemed to encapsulate the same idea. Next, Figure 2.4 is the distribution of the `Slope` attribute. A small number of these observations were logged as Unknown. For sake of simplicity and computational efficiency, these 426 observations were reclassified as Level.

The `Elevation` feature had a similar structure and is dealt with the same way as `Slope`.

[3] Coordinate data, `Distance.to.Center`, etc.

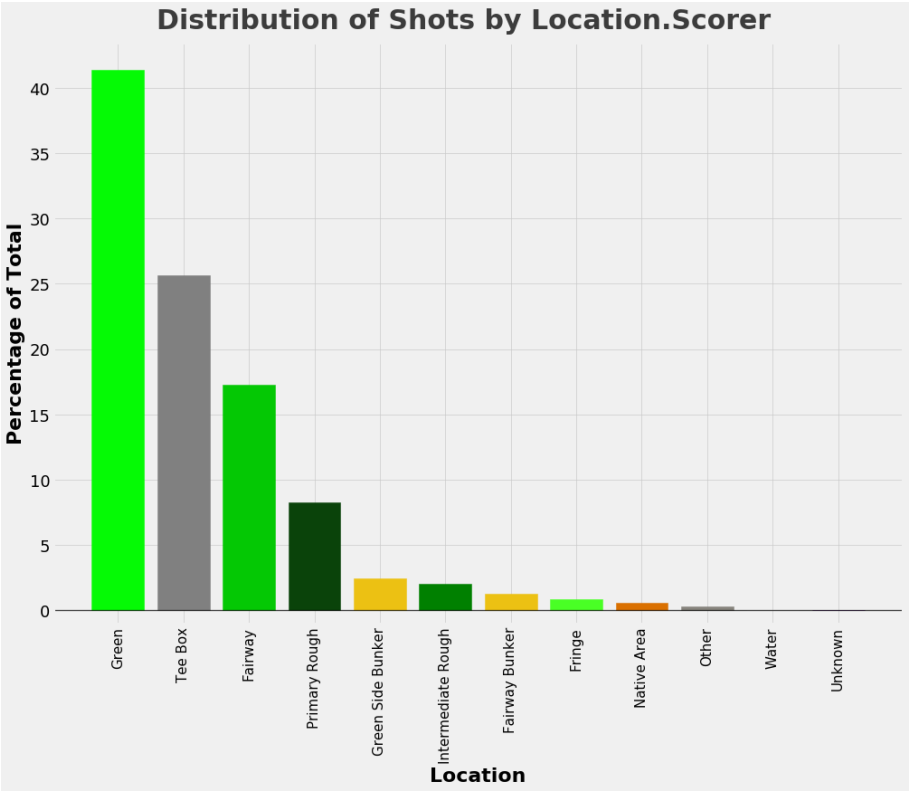

Figure 2.3: Distribution of shots by `Location.Scorer` for all shots in the 2018 PGA Tour ShotLink data.

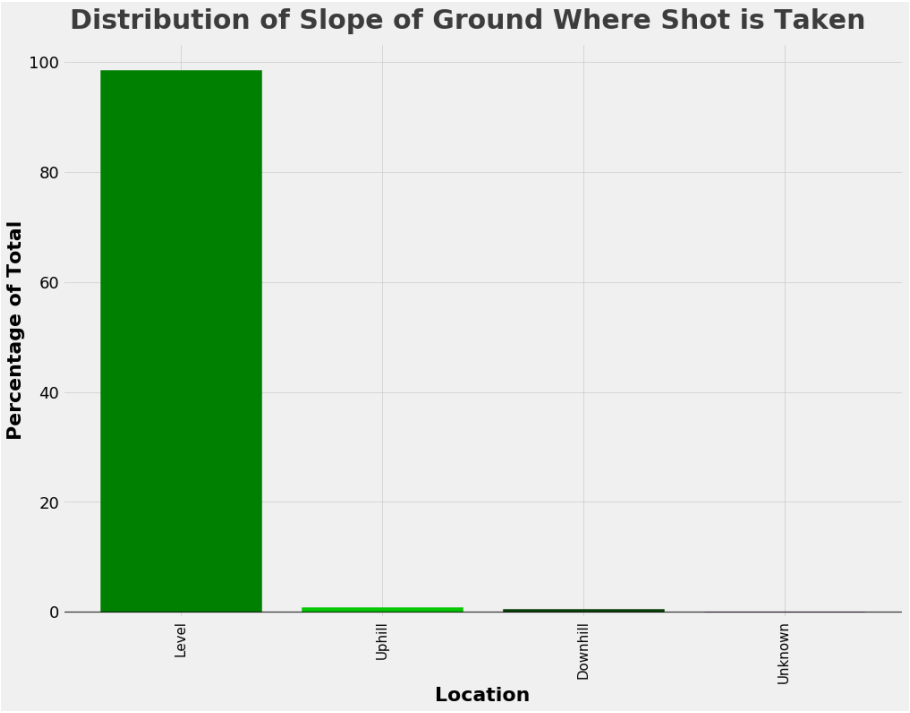

Figure 2.4: Distribution of shots by `Slope` for all shots in the 2018 PGA Tour ShotLink data.

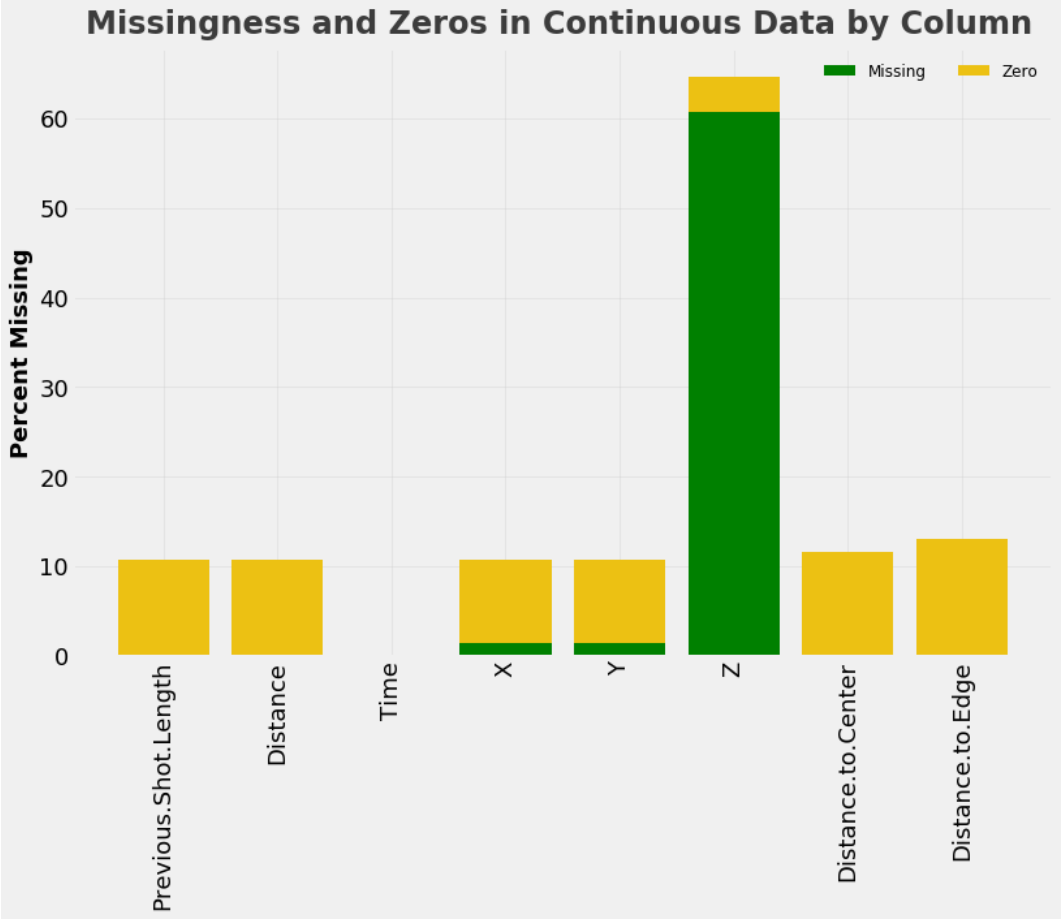

Figure 2.5: Frequency of both missingness and zeros in the continuous features of the 2018 PGA Tour ShotLink data. The columns are stacked such that the top is the two quantities added together.

2.2.4 Continuous Features

The next step in this analysis involved addressing inconsistencies in the continuous features. There are two possible red flags in continuous data that this analysis flagged. The first is data that was missing or logged as N/A. The second is data that was entered as zero, often because an event with no data was entered as a zero. To explore this further, I plotted the frequency of both zeros and missing data in the eight continuous variables being used for prediction in Figure 2.5. As we can see, `Previous.Shot.Length`, `Distance`, `X`, and `Y` all had a similar level of suspicious data. Upon further investigation, all

of these zeros and N/As came from the same eight courses [4], all of which had no location data recorded. Due to the location specific nature of this analysis, I removed these eight courses from the data set, leaving three columns with zeros or missingness.

Once these courses were removed, I was left with a small number of observations with 0 recorded for `Distance.to.Center` or `Distance.to.Edge`, and still roughly 50%-60% of the `Z` coordinates missing. Looking deeper into the first two columns, there did not seem to be a pattern to the missing values. Paired with the fact that it seemed plausible to record the true value as zero for these attributes, I decided to leave them in the data. At this point in the analysis, I determined that the `Z` coordinates were not missing with an obvious pattern, and that it was going to be too intensive to impute them across all courses with a high enough level of accuracy to include the values into a prediction accuracy[5], although it was still used sporadically for distance estimates.

This step removed 123,832 observations, leaving a total of 1,027,068.

2.2.5 Data Inconsistencies

Due to the structural complexity of this data, it was possible to determine certain rows that were invalid due to inconsistencies in the measurement. Before looking for inconsistencies, it was important to make sure that the location data all had the same units. `Distance`, `Distance.to.Center`, `Distance.to.Edge`, and `Previous.Shot.Length` were recorded in inches,

[4] Corales Golf Club, El Camaleon GC, Torrey Pines (North), Spyglass Hill GC, Sea Island Resort (Plantation), Monterey Peninsula CC, Nicklaus Tournament Course, and La Quinta CC

[5] For an interested reader, I found that smooth bivariate splines work quite well for the nonuniform but still tightly packed data on the green

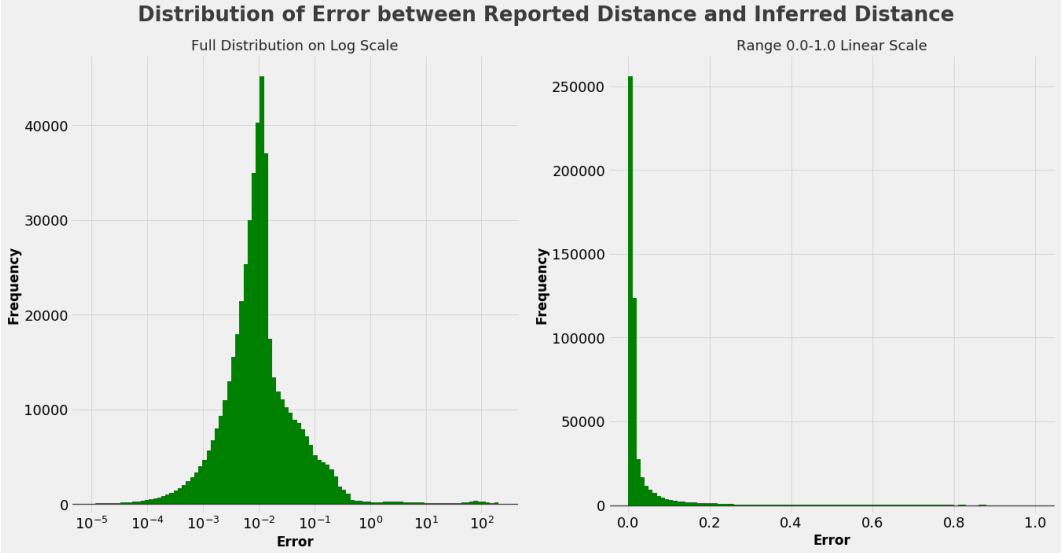

Figure 2.6: The figure on the left shows the distribution of error between the estimated shot length using the coordinate system and the recorded `Shot.Length` variable plotted on a logarithmic scale. The figure on the right is the same data plotted on a linear scale between 0 and 1.

while `X`, `Y` and `Z` are recorded in feet and yardage was clearly recorded in yards. Since most of the common conversation around golf seems to revolve around yardage, all location data was converted into yards.

While there were potentially a lot of complex methods that could be used to estimate course features and look for inconsistencies, this paper focused on one obvious form of discrepancy. Since the beginning and the end of many shots are logged through a coordinate system and the distance of the shot is recorded, it is not too difficult to scan for inconsistent measurements[6]. Figure 2.6 is a graph of the distribution of errors in the location estimates relative to coordinates. These two graphs show a relatively low average error rate of between 10^{-2} and 10^{-3} yards. There are, however, a reasonable number of outlier points with errors that needed to be addressed. For sake of simplicity, the cutoff was set at 1 yard. Since it did not make sense to remove individual strokes from a sequence for a particular golfer, if a golfer's

[6]This was inspired by the data cleaning appendix in Levin [Lev17].

sequence on a given hole had a single inconsistency of over 1 yard, the entire hole for that round was thrown out.

At the end of this step, 27,466 observations were thrown out for inconsistencies, leaving 999,602 observations in the final data set.

2.3 Feature Engineering

After cleaning up the data, a few features could be engineered to improve analysis and visualization.

2.3.1 Aggregated Categoricals

Often in the process of model fitting, a categorical attribute with minimal observations will hurt inference and prediction ability because there are not many other observations to fit the coefficients. This problem is magnified when considering interaction terms, which form an even more granular representation of each category. While this was not a problem for most of the features in this data, two issues jumped out that required a closer look: `Location.Scorer` and `Player`. As shown in an earlier section, there were four categories in`Location.Scorer` that have fewer than 10,000 observations: fringe, native area, other, and water. In addition to estimating each of these categories individually, I created a second variable called `Location.Scorer.Agg` which assigned the four small categories in the following way: water and native area were combined into other because they were anomalous shot types that should be much more difficult than average. Fringe was combined with fairway. The fringe sits between the fairway and

the green, and professional golfers try to avoid putting on the fringe, treating it more like the fairway than the green. Finally, I combined both green side bunker and fairway bunker into one large category of bunker since the main difference between these two location types is distance, something that the models attempted to control for.

The next variable I considered was `Player`. This feature was uniquely imbalanced because some players played almost every weekend, while others only played once or twice a year. Instead of generating a player strength coefficient for some of the unknown players, another approach was to create a replacement level player for players that have played fewer than a certain number of courses in a given year. To get a sense of the distribution of courses that players have played, I plotted Figure 2.7,which shows a large number of players who have player one or two courses, and a relatively flat distribution after that with some concentration around PGA Tour regulars between 18 and 25 courses. I categorized every player who has played fewer than 10 courses as replacement level. The new feature with this distinction was named `Player.Agg`.

2.3.2 Hole Location

No location data was recorded when the ball went in the hole, so in order to perform accurate visualization the hole location needed to be estimated. This `Hole.Location` estimate was also critical to engineering the `Eff.Green` variable in a later section. Given `Distance` and the `X` and `Y` coordinates of given shots, I employed a crude method for learning `Hole.Location` by taking every shot within half a yard of the hole and averaging their `X` and `Y` coordinates. Even with a strong structural bias in the course this should impute a value relatively close to the actual hole. This value was calculated for

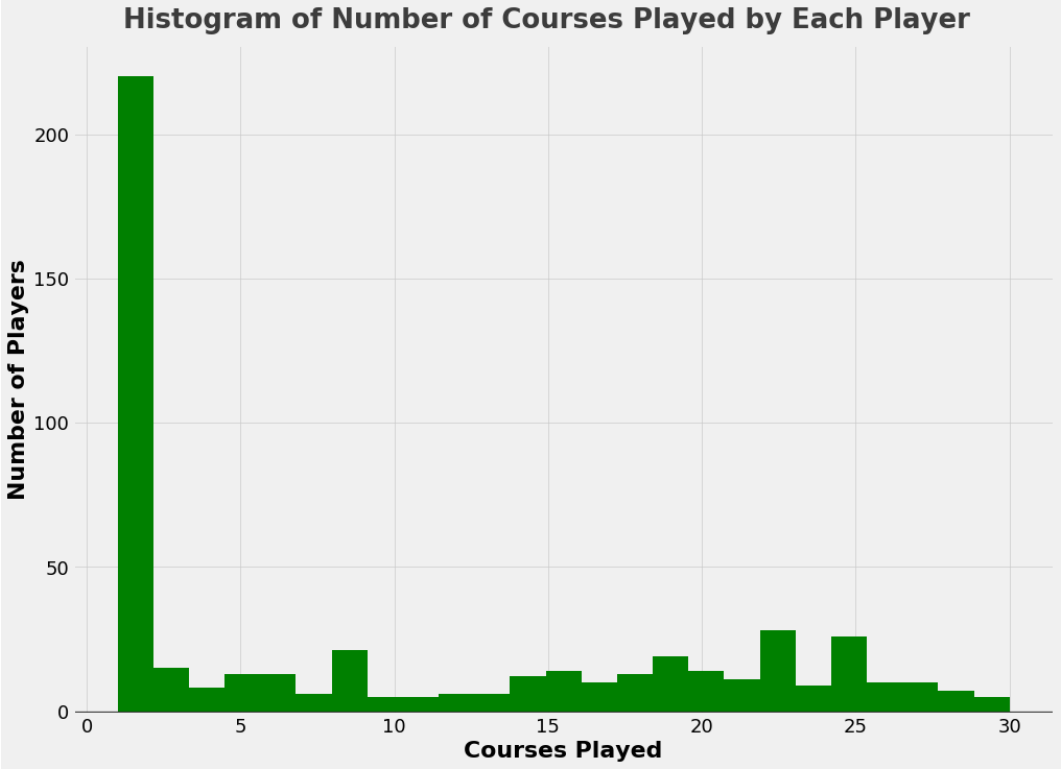

Figure 2.7: Histogram of the number of courses played by each player on the PGA Tour in the 2018 season.

every hole on every round that it was played (since the hole location moves). An improvement on this method could include incorporating `Distance` more directly and using an algorithm similar to the one described in the next section.

2.3.3 Tee Location

The tee location is a bit more complicated than the hole location for two reasons. First, there were very few shots within 50 yards of the tee. Second, the tee is not a fixed point, but instead a line that normally shifts back each round a small amount[7]. In order to learn these lines, I created an algorithm

[7] I do not know many details on how this works, just that it has been confirmed to me by a few golfers and is clearly visible in the data

that took the following steps 1000 times in order to generate a set of points that represented the tee line.

1. Isolate all of the second shots on a given hole in a given round and sample two points.

2. Use the `Previous.Shot.Length` and the `X`, `Y` coordinates of each shot to draw two circles containing possible tee locations.

3. Find the two intersections of these circles.

4. Select a third point and measure the distance to each of the two candidate tee locations.

5. Select the point with the measured distance that is closer to the `Previous.Shot.Length` closest to the recorded value.

An example of this process is shown in Figure 2.8. The intersections of the green circles represent the initial candidates for tee location. Since the grey circle nearly intersects the bottom point, this point was chosen as the candidate tee location. As is shown, this point was very close to the other candidate points that were selected in the other iterations. Once this set of points was generated for a given round on a hole, the points more than 10 yards from the average tee location were removed, and an OLS model was fit to approximate the tee line[8].

[8] I was told that a tee line is definitely not longer than 20 yards, something I saw empirically in the data, but a deeper investigation into the specific rules around this line would yield a better set of assumptions.

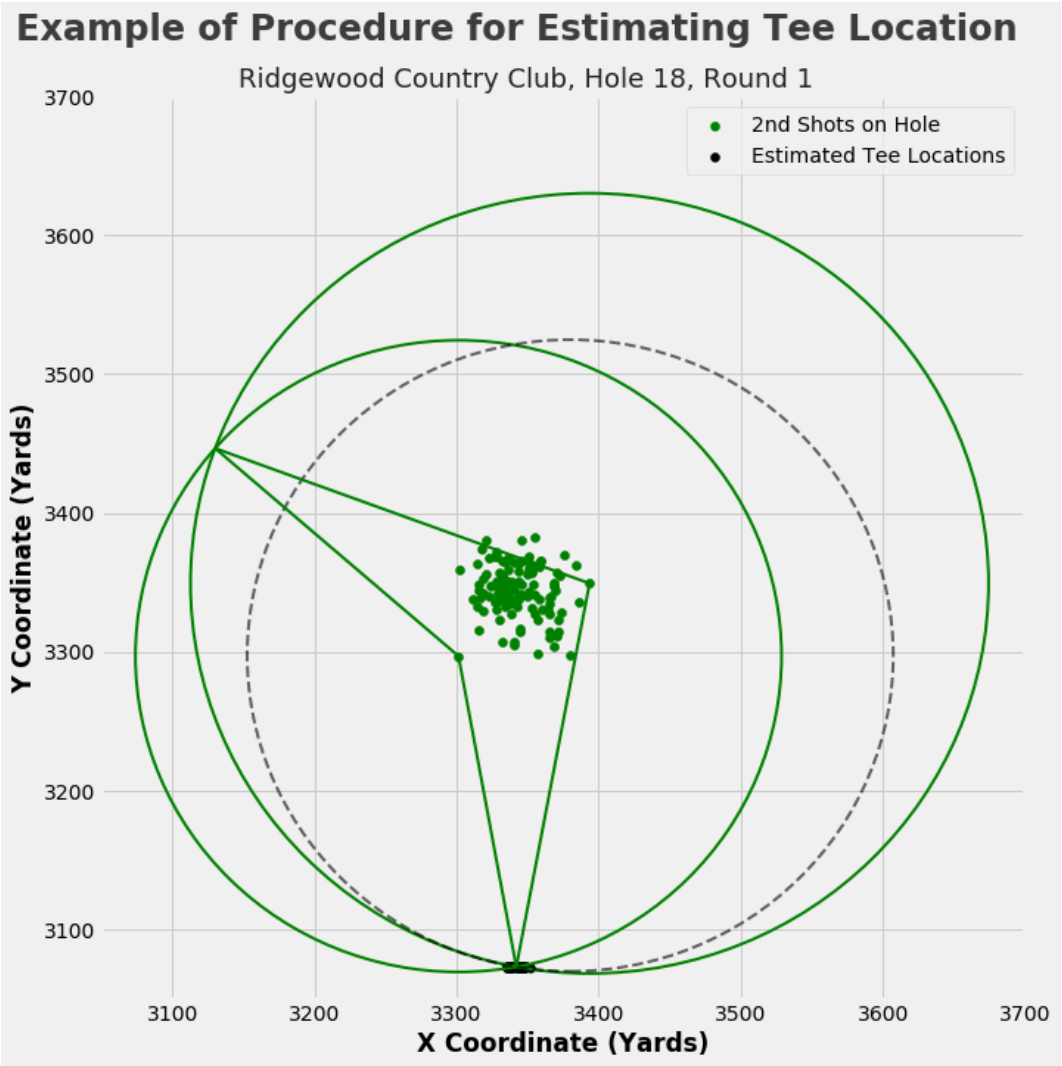

Figure 2.8: A diagram of one iteration of the algorithm created to impute the tee line for a given round. The bottom intersection of the two green circles in the imputed location.

Again, this feature was not directly used in prediction, and `Tee.Location` was used even less frequently for visualization than `Hole.Location`. However, there were some practical applications of this variable for visualization, especially in the section were I measured performance off the tee. Additionally, the direction for the tee shot has large implications for interaction with wind and other obstacles that were beyond the scope of this paper. Due to the sparsity in features for the all of the tee shots in this data, I believe that this algorithm provides general value even though I was not able to make the most of it.

2.3.4 Effective Green

The idea for effective green again came from Levin [Lev17], as well as an interview I conducted over the phone with PGA Tour pro and Harvard undergraduate applied mathematics alumnus Rohan Ramnath. Levin incorporated a metric into his model that measured the space in front of the hole on the green, calling it "Green to Work With." When I asked Ramnath what features he enjoyed seeing when taking an approach shot, he quickly mentioned that he would like to see a wide green so that his ball lands on the green even if he missed wide left or right. From these two insights, I investigated the importance of hole position on the green by engineering four additional features, `Eff.Green.Front`, `Eff.Green.Back`, `Eff.Green.Right`, and `Eff.Green.Left`. The name "effective green" is a reference to the idea that while the green might be large by area, the only important areas of the green for a golfer on an approach shot are the portions that will catch either a horizontal or vertical miss[9].

[9]To the best of my knowledge this is the first formalization of this concept.

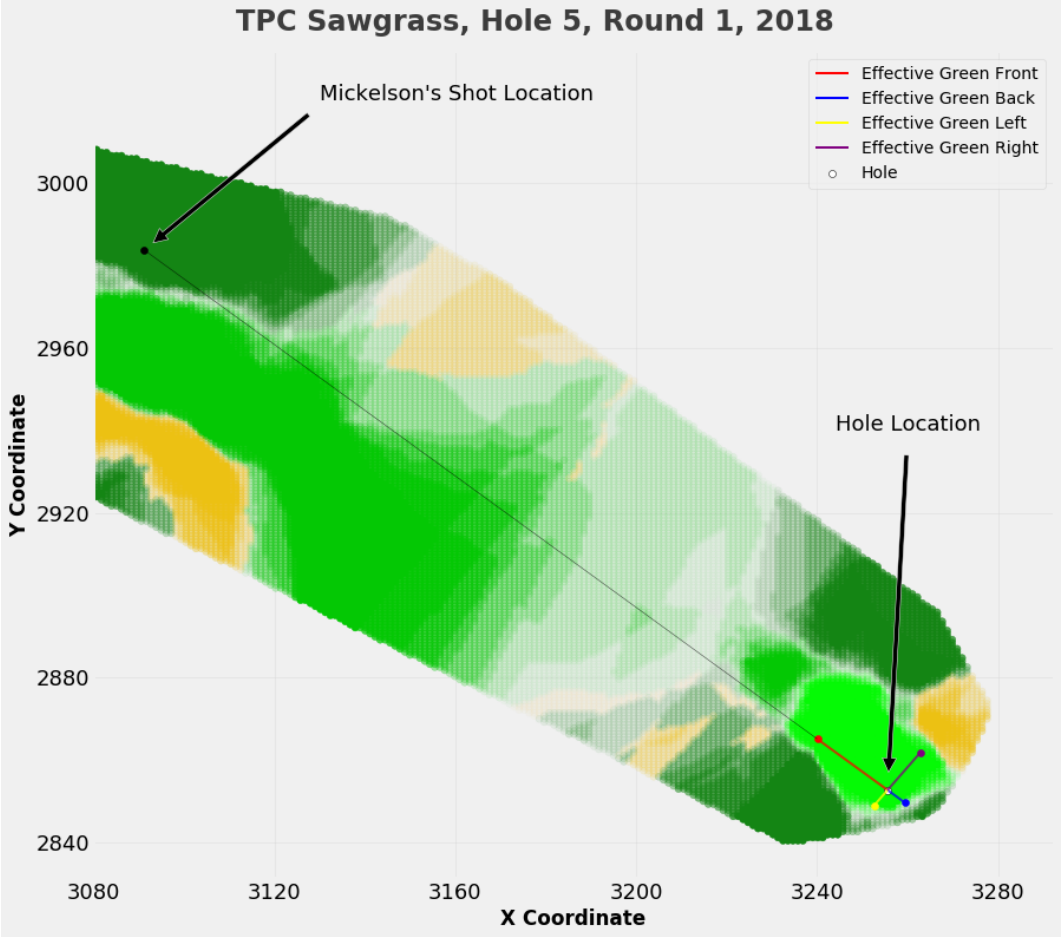

Figure 2.9: A visual representation of `Eff.Green.Front`, `Eff.Green.Back`, `Eff.Green.Left`, and `Eff.Green.Right` for Phil Mickelson's second shot on TPC Sawgrass, Hole 5, Round 1.

To demonstrate this concept visually, recall the approach shot from Phil on TPC Sawgrass hole 5, as shown in Figure 2.9. The lines on the green demonstrate the effective green that Phil has to work with; he does not have much in the way of missing left or long, but he has a lot of space in the front and reasonable amount of space on the right[10].

Looking closely, one might notice that the the lines for `Eff.Green` do not trace all the way to the edge of the green as displayed. This is on purpose. For visualization I was interested in algorithms that would aggressively learn about the space, while for feature engineering I was interested in taking a

[10]Didn't seem to do him much good.

more conservative approach. The way this manifested itself was as follows: For `Eff.Green`, the green was only considered to be the convex hull[11] that encompasses every point on the green. For visualization, points outside the green that do not have much else around will be classified as green by the k nearest neighbors algorithm. The side effect of this is that the first definition of green can be thought of more as "playable" green. Every course has certain areas of the green where balls are unlikely to ever stay put.

2.4 Exploratory Data Analysis

When exploring golf statistics, the key variable in almost every equation was `Distance`. Consider a few examples to illustrate this point. When exploring whether a specific hole has an affect on the Strokes.Remaining, every par three will look like a massively statistically significant indicator relative to a par 5. After controlling for distance however, this effect almost shrinks to 0. The same thing happens for different types of locations. Clearly a ball being placed on the green will be a strong predictor of having fewer`Strokes.Remaining`. After controlling for `Distance`, this is still true but to a much less noticeable degree.

Because of this, most of the EDA in this section was conducted relative to `Distance`. If a systematic effect was observed as `Distance` varied, this was a strong sign that a variable would have some significance in a model later on.

[11] An algorithm that wraps a given step of points with the smallest convex shape.

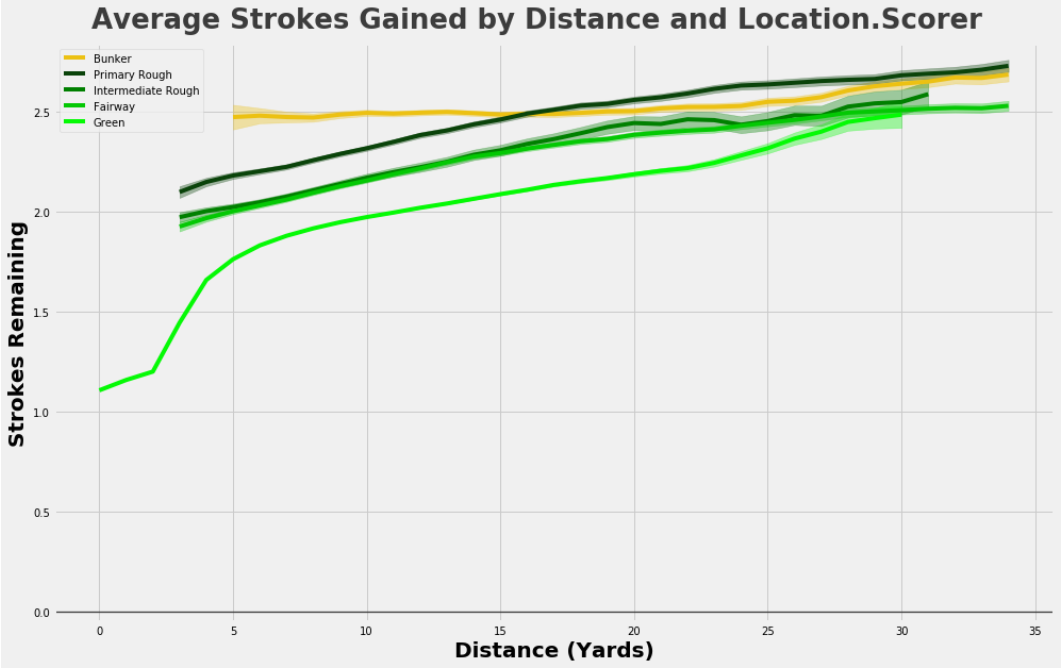

Figure 2.10: Average number of strokes by `Location.Scorer` and `Distance` under than 35 yards. This is plotted using a five yard rolling mean and 95% standard error intervals on the mean estimates.

2.4.1 Location

The first variable I considered relative to `Distance` was `Location.Scorer`. After Broadie [Bro12], almost every golf paper referenced this relationship as the baseline for prediction. To observe this effect, I aggregated data by the number of yards away each event was, and then I applied a rolling mean and standard error calculation over a 5-yard window. The results were then split less than 35 yards as shown in Figure 2.10, and further than 35 yards as shown in Figure 2.11, in order to highlight the effect of the green. Note that sparse location types were excluded from this section including water, fringe, native area, and other. Both of these graphs show obvious directional effects for most of the different location types.

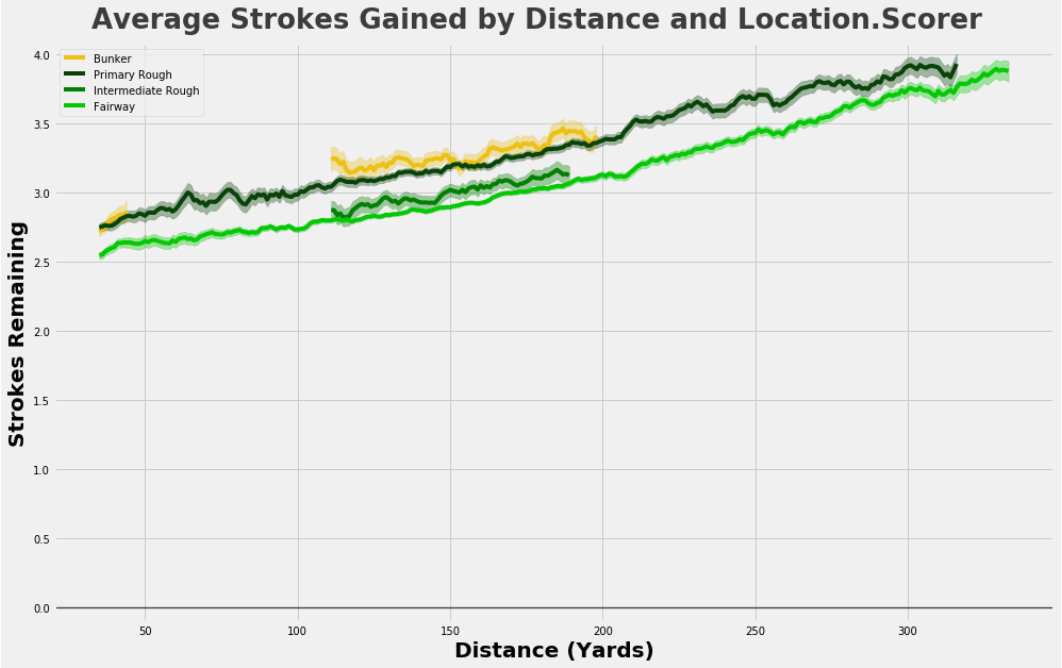

Figure 2.11: Average number of strokes by `Location.Scorer` and `Distance` further than 35 yards. This is plotted using a five yard rolling mean and 95% standard error intervals on the mean estimates.

2.4.2 Distance to Center

The PGA records another interesting spatial statistic that is closely related to `Location.Scorer`: `Distance.to.Center`. Because this variable was highly collinear with `Location.Scorer`, it was useful to graph its relationship with distance over all the data, but also in specific subsections such as the fairway and the rough. Because this is a continuous variable, it was broken up into 4 discrete sections, 0-5 yards, 5-10 yards, 10-20 yards, and 20+ yards, as shown in Figure 2.12. While a global trend was observed here, it was interesting to note that this trend was mostly driven by the primary rough. For players in the primary rough, 20+ yards out was notably worse than 10-20 yards away from the center. This makes intuitive sense because the grass and obstacles can get very difficult far off of the fairway.

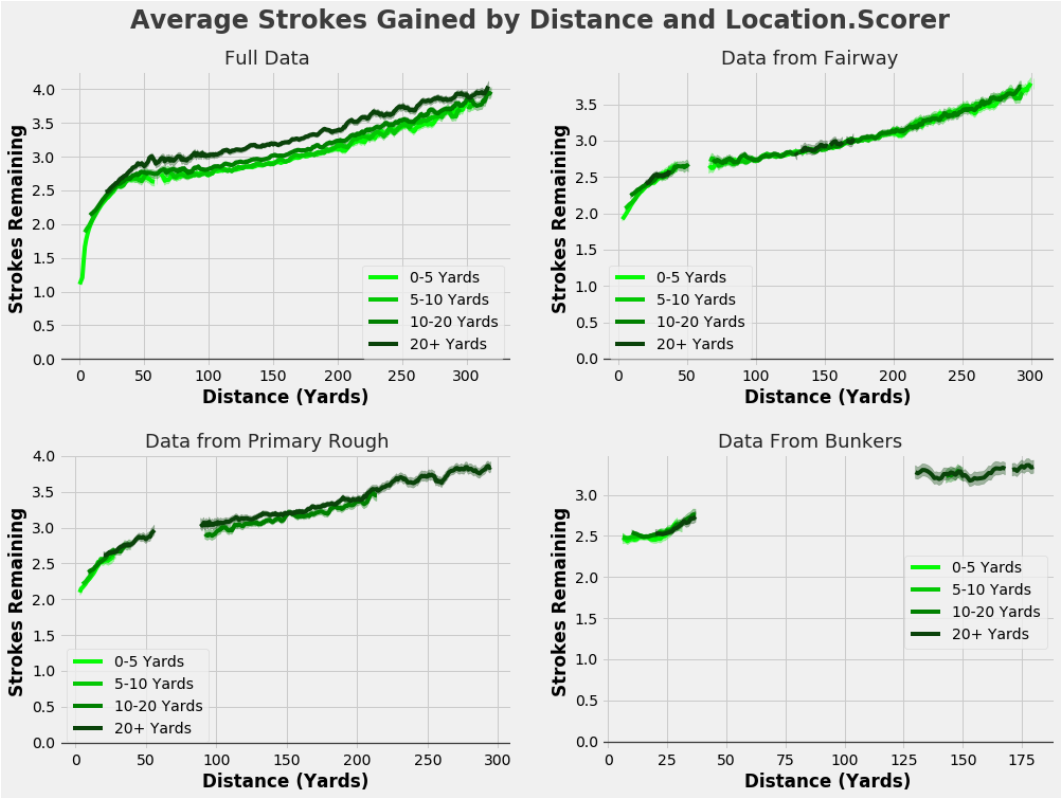

Figure 2.12: Average number of strokes by `Location.Scorer` and `Distance`, varied by `Distance.to.Center` of the shot. The `Distance.to.Center` was sorted in 4 bins, and plotted over the full data, fairway, primary rough, and bunkers.

3

Methods

The computational and visualization tasks in the paper were split between two data science programming languages, `R` and `Python`. The work flow tended to be as follows. Data engineering and cleaning were conducted in `Python` using `pandas` and a variety of other data cleaning tools. The data was then exported to a csv and uploaded to an `AWS ec2` instance[1] where it was used to fit a model in `R`. Once the model was fit to satisfaction, I was then able to export it to a local computer for analysis and interpretation using my local `R` kernel. Once I had a good sense of the model, I embedded it into a `Python` script using an `R` kernel and the `rpy2` package to pass arguments between `Python` and `R` data structures. In `Python`, I decomposed these models to be properly visualized using a variety of visualization tools built in `matplotlib`.

While this transfer back and forth between languages may seem cumbersome, this problem in particular highlighted the relative strengths and weaknesses of each piece of software. `Python` is a more flexible data engineering tool and also allows for an incredibly high level of pixel by pixel customization in visualizations. On the other hand, `R` has a much more sophisticated traditional statistics community and so has access to a much broader array of modeling packages, with a higher level of customization and better documentation. There is no way to fit complex, out of the box GAMs in python without relying on some shaky and poorly maintained infrastructure. The data pipeline was tedious and required working with the underlying data structures of these

[1] This stands for an Amazon Web Services Elastic Cloud Computing Instance. These are industry standard for remote computing.

models, but I believe it is worth it seeing as a large part of this paper's contribution is based around visual inference.

The remainder of this paper will be split up into two main threads. The first is the linear modeling section. In order to obtain a reasonable frame of reference for these models, I fit a series of linear models using both fixed and random effects. While this provided some insight as to which features were significant, these models were sparingly used for inference because they were not as descriptive. The second set of models I employed were generalized additive models and generalized additive mixed models. As will be explained in a later section, these are a form of generalized linear model that allows the predictor to vary based on a nonlinear smooth functions of the features and/or smooth interactions. I also performed some diagnostics comparisons of the two models to analyze the impacts of the additive model structure.

In addition, both the linear and additive modeling sections follow a similar format: a baseline model was fit using the features `Distance` and `Location.Scorer`. These two features have been used in essentially every study on golf stroke data, and if used correctly they account for a large percentage of the deviance in shot difficulty. These two variables are colloquially used to describe shots at all levels of golf (e.g. 200 yards out from the primary rough) and so they provided a nice frame of reference for the rest of the analysis. The other reason for this structure was that due to the computational constraints of this problem, it was not possible to do a broad search over all possible model structures. Instead, different models needed to be built to answer different questions.

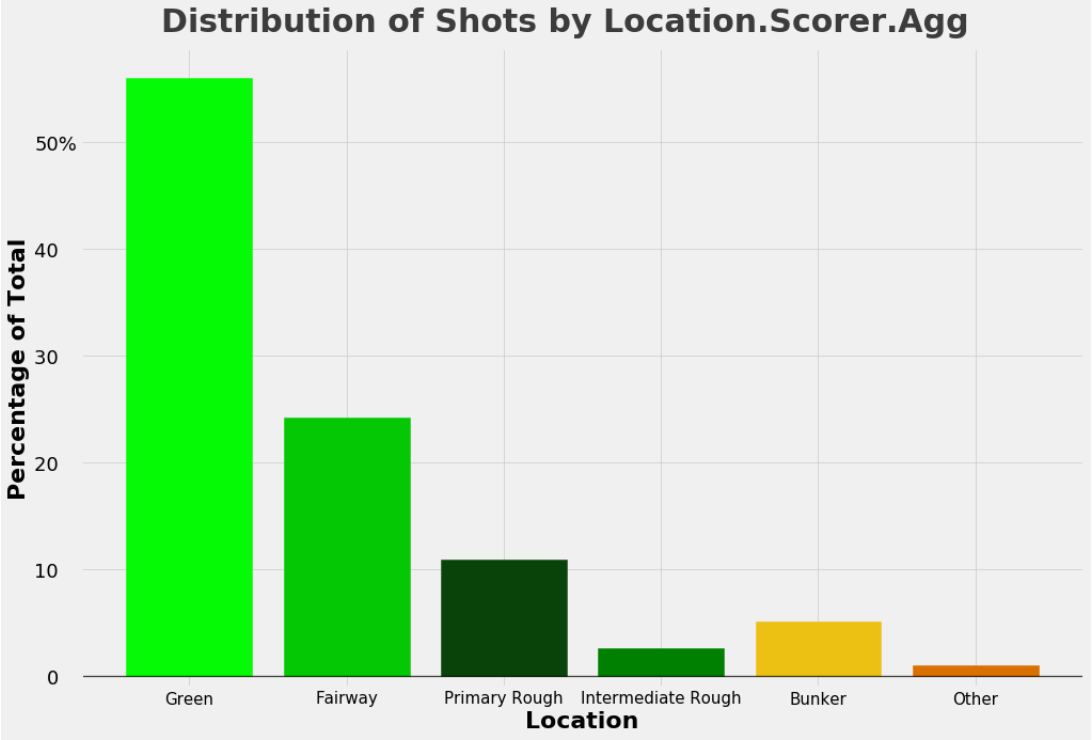

Figure 3.1: Distribution of shots on the PGA Tour by `Location.Scorer.Agg` in 2018. Tee shots have been removed.

3.1 Model Accuracy and Comparisons

The accuracy of every model was scored in the following way. The data was divided up into a 2/3 training set and 1/3 test set, stratified by `Location.Scorer`[2]. The model was subsequently fit on the training data. After this fit, it was scored out of sample on the test data as a whole, as well as each of the following six location groups independently: green, fairway (a combination of fairway and fringe), primary rough, intermediate rough, bunker (a combination of green side bunker and fairway bunker), and other (a combination of native area, water, and other). The distribution of these shot types is as seen in Figure 3.1. Next, the model was tested on the set of strokes that ShotLink had tagged "Recovery Shots" according to the formulation in Broadie [Bro12]. To touch briefly on recovery shots, these are shots that

[2]For some models, additional stratification was needed so that every model was fit with a full coefficient set.

Broadie essentially thought were obstructed, and therefore were much more difficult than his model would suggest. While this was a fine idea in theory, in reality he implemented this approach by looking for shots where the stroke did not make it anywhere near the hole, and then worked backward to infer an obstruction. He used these shots to attempt to tease out obstructed shots in the surrounding area as well. As I will discuss more in the next section, minimizing forward-looking bias should be a goal of any model of this structure. One could easily imagine a situation in which a bad player gets a disproportionately high fraction of recovery shots and a good player gets fewer, making the worse players average shot difficultly look higher than it actually is and vice versa for the strong player. Since this difficulty metric is used as a baseline for Strokes Gained player rankings on the PGA Tour, the weak and strong players will both have skewed performance metrics as a result. Instead of including recovery shots in my feature set, I instead included them as a category in my out of sample evaluation and explored a few strategies for increasing the prediction accuracy on this group of shots without compromising the rest of the coefficient estimates.

Finally, I evaluated these models based on pairwise ANOVA tests with a `Chisq` distribution. While I was able to see if the marginal addition of certain features produced statistically significant coefficient estimates, in marginal cases ANOVA tests evaluate whether the decrease in deviance is worth the increase in degrees of freedom.

3.2 Forward-Looking Bias

An astute reader will notice that a number of the features were engineered using forward-looking information. Take the `Eff.Green` metrics as an exam-

ple. If we assume that the only information available at the time of the shot is the ShotLink data collected before the shot takes place, then for the first shot on a given hole there would be no way to construct an estimation of the size and shape of the green. As the tournament progressed, the `Eff.Green` estimates would get more and more precise until they were a reasonable approximation of the green. To address this concern, I will make a distinction between temporal independence and uncertainty about location attributes. In the case of the green, the size and shape of the green are known at the time of the shot, I just did not have that data available. Adam Levin has experimented with sampling pixels from satellite images[3], and the PGA Tour will inevitably increase the granularity of its data as time goes on. Also, these quantities could be estimated more accurately in a Bayesian manner using shot data from a given hole in previous years. A very precise reader could argue that the feature engineering method does not consider the actual green, but instead the "playable" green[4], to which I will concede that this could present an extremely marginal amount of forward-looking bias. Therefore, it seems sufficient to claim that while all of the information present for this prediction is not easily retrievable at the time of the shot, it is fixed and not forward-looking in the sense that it would cause complications with inference.

The features that suffer from this structural uncertainty are the four measurements for `Eff.Green`, and any out of sample estimation that I made using location imputation[5]. Additionally, the `Hole.Location` and `Tee.Location` were imputed using all of the data from a given `Round`.

[3] He told me this in a phone interview on March 14, 2019.

[4] There may be certain sections of the green that are too slanted for a ball to end there, information which is not immediately obvious but can be inferred from analyzing all the shots throughout the day.

[5] Will be discussed in a later section.

The next source of forward-looking bias comes from the "out of sample" model accuracy measurements. While an argument can be made that the previous features did not include forward-looking temporal information, this model fitting procedure clearly used information collected after the shot takes place. For example, the slope of the relationship between `Strokes.Remaining` and `Distance` is fit using data from all the tournaments in 2018, and then evaluated on different data coming from the same tournaments. This issue has separate implications for inference and prediction. For inference, there is not much reason to think that there is a meaningful relationship between shots taken earlier in the day on a hole and later in the day on a `Hole`. So long as certain assumptions are made such as the categorical contribution of a `Round` or a `Course` being a fixed constant over time, or the global relationship between `Distance` and `Strokes.Remaining` staying fixed over the course of 2018, the inference procedure will not have been tainted.

3.3 Tee Shots

The final note to make before discussing the actual model constructions is with respect to tee shots. Tee shots were unlike every other shot in this database in that they were mostly devoid of location attributes. While I previously discussed a method to impute the rough location of the tee box, it was functionally impossible to estimate the relative location of each player's tee within the box. On top of this, all the tee shots from a given round of a hole essentially had the same location attributes. Features like `Distance.to.Center` and `Distance.to.Edge` had basically no meaning, every shot has the same `Distance`, `Eff.Green`, `X`, `Y`, etc. Because of this, I left the modeling of tee shots to a future researcher.

3.4 Linear Models

3.4.1 Building a Baseline with Fixed Effects

To formulate a reasonable baseline for this problem, one approach I investigated was to model this as a multiple regression but with only linear fixed effects. While the exploratory data analysis section showed that this data at the very least does not vary linearly with distance, linearity is a reasonable approximation within specific regions of a given `Location.Scorer`. This will be shown in more detail in the results section.

Additionally, while it has been long understood that `Strokes.Remaining` do not vary linearly with distance [Bro12; Lev17], there is no accepted method of modeling this non-linearity. Therefore, I started with a linear baseline to gauge future constructions.

In a standard linear regression model, the continuous response variable y is estimated using a linear combination of observations and coefficients plus some error term.

$$y_i = \alpha + x_i^T \beta + \epsilon_i.$$

Linear models can also be adapted to take on non-numeric, "categorical" variables such as `Location.Scorer` in the following way: we can divide the categorical variable into N binary variables where N is the number of categories[6]. When a specific category is selected, this binary indicator is set to 1, and all the other ones are set to 0, with the effect that the contribution to y_i is the coefficient of the given category. Instead of writing this as N separate

[6] In reality, only N-1 variables are used to avoid multicollinearity, meaning that one variable is absorbed into the intercept and activated by setting all of the indicators to 0. However, R masks this process and provides a coefficient for all categories.

terms, I abbreviated the set of terms as $\beta_i \times$ `Categorical.Variable` for sake of brevity.

These models are penalized using ordinary least squares. From this framework, I explored a few possible baseline model formulations. For the remainder of this paper, the three most common variables, `Strokes.Remaining`, `Distance`, and `Location.Scorer`, will be abbreviated `S.R.`, `Dist`, and `Loc.S` in formulas respectively.

Distance-Only Linear Model

In the most crude sense, `Strokes.Remaining` clearly varies with `Distance`. This is the first model that a naive observer would fit, and I have included it to show the importance of including `Location.Scorer` in the next step.

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} \qquad \text{(LM.Dist)}$$

This is a very crude model that understands functionally nothing about the golf course.

Baseline Linear Model

The next obvious improvement was to vary `Distance` by `Location.Scorer` using a categorical variable and an interaction term. Interactions will be represented using a multiplication symbol ($\times$).

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} \qquad \text{(LM.Baseline)}$$

I fit a few different versions of this model. First, it was not clear that both the categorical term and the interaction term were needed, so I checked both reduced versions of this model. Additionally, I investigated whether the out of sample accuracy would be higher by replacing `Location.Scorer` with `Location.Scorer.Agg`. This reduces the descriptiveness of the model, but for some of the sparse terms it was plausible that this would help with overfitting. This model was much closer to a true baseline than the distance-only version, and captured a large amount of the variation in shot difficulty. Every model after this employed a different strategy to improve on this model construction.

Fixed Course Effects

Another intuitive step that has been taken by researchers in the past is to model the `Course` and `Hole` categorical effects. In his papers, Broadie pointed out that the difficulty of fairway and other surfaces will vary course-to-course and hole-to-hole even when controlling for distance, because of things such as grass length, layout, and other obstacles that could be present. In addition to this, I tested the influence of `Round` because this could capture the change in hole location and weather, among other . To evaluate these impacts, I iteratively introduced these three levels of granularity, testing whether the marginal addition was a significant improvement. The three models are shown below

$$\texttt{S.R.} \sim \beta_0 + \beta_1\texttt{Dist} + \beta_2\texttt{Loc.S} + \beta_3\texttt{Loc.S} \times \texttt{Dist} + \beta_4\texttt{Course} \quad \texttt{(LM.Course)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1\texttt{Dist} + \beta_2\texttt{Loc.S} + \beta_3\texttt{Loc.S} \times \texttt{Dist} + \beta_4\texttt{Course} + \beta_5\texttt{Course} \times \texttt{Hole} \quad \texttt{(LM.Hole)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1\texttt{Dist} + \beta_2\texttt{Loc.S} + \beta_3\texttt{Loc.S} \times \texttt{Dist} + \beta_4\texttt{Course} + \beta_5\texttt{Course} \times \texttt{Hole} + \beta_6\texttt{Course} \times \texttt{Hole} \times \texttt{Round} \quad \texttt{(LM.Round)}$$

This construction allows potentially for both an improvement of prediction accuracy and inference on the relative difficulty of courses and holes.

Fixed Player Effects

The last baseline linear model I fit was an incorporation of `Player` effects into the existing final course model. I did this at two levels. First, I fit an additional categorical variable that measured which player was taking the shot. I then tested the impact of adding an interaction between `Player` and `Location.Scorer` with the hypothesis that player effects can often be segregated to certain locations on the course (e.g. Jason Day is a better putter than Tiger Woods, but worse than him on the Fairway). This yielded the following two models

$$\texttt{S.R.} \sim \beta_0 + \beta_1\texttt{Dist} + \beta_2\texttt{Loc.S} + \beta_3\texttt{Loc.S} \times \texttt{Dist} + \beta_4\texttt{Course} + \beta_5\texttt{Course} \times \texttt{Hole} + \beta_6\texttt{Course} \times \texttt{Hole} \times \texttt{Round} + \beta_7\texttt{Player} \quad \texttt{(LM.Player)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1\texttt{Dist} + \beta_2\texttt{Loc.S} + \beta_3\texttt{Loc.S} \times \texttt{Dist} + \beta_4\texttt{Course} + \beta_5\texttt{Course} \times \texttt{Hole} + \beta_6\texttt{Course} \times \texttt{Hole} \times \texttt{Round} + \beta_7\texttt{Player} + \beta_8\texttt{Player} \times \texttt{Loc.S} \quad \texttt{(LM.Player.Loc)}$$

On top of this, I tested the impact of substituting in both `Player.Agg` and `Location.Scorer.Agg` in terms of computational efficiency, statistical significance and out-of sample prediction accuracy. Not only did this have the impact of reducing the feature space by over 50%, importantly it also avoided fitting `Player` fixed effects with small amounts of data.

It is worth noting that throughout this process I treated `Player` effects slightly different from the rest of the features when deciding whether or not to include a given effect. I took the view that it is essentially a mandatory condition that a model produces statistically significant `Player` effects at the very least, and important that it can also distinguish between player skills on different surfaces. In contrast to other features that might be excluded if they do not seem to inform shot difficulty, I took it as given that there is stratification between player abilities in golf. On top of that, a main focus of this thesis and golf analytics in general is determining relative player abilities, and so a model that could not pick this up was of marginal value.

After investigating a few simpler linear models, I looked for alternative strategies for dealing with these sparse categorical variables.

3.4.2 Investigating Random Effects

After taking a closer look at the feature set for this model, a few things jump out about the categorical variables. While some of them make precise distinc-

tions, there are a few such as `Course`, `Course` × `Hole`, `Course` × `Hole` × `Round`, and `Player` that are likely to be important but are also very sparse. Additionally, the `Course/Hole/Round` features have a hierarchical structure to them that can provide some information beyond strictly a series of interactions. Because of these properties, this model is a good candidate for the inclusion of multilevel modeling.

According to Gelman and Hill [GH06], multilevel modeling can be thought of as a standard linear model with an indicator variable interacted with either the intercept or a specific coefficient so that it can take on multiple values. The critical way that it varies from a standard regression model with fixed effects is that the set of possible coefficients for a category is also described by a model that is fit simultaneously. Gelman and Hill [GH06] describe this process as essentially a trade off between *complete pooling* where all the coefficients are the same, and *no pooling* where separate models are fit for all categories. This has two major benefits. First, the coefficients can learn together to some degree, so some information modeled into one coefficient is used to fit the other ones. The result of this is that the coefficients in a multilevel model are normally closer together than a standard fixed effect regression. Second, these models can fit much more quickly than the same model with a fixed effect. This is useful for some of the interaction terms that take on a lot of values.

There are two types of multilevel models, varying intercepts and varying slopes. A varying intercept model only performs this "random effect" on the intercept term, and has no direct impact on the other features. A single term simple varying-intercept model takes on the following form

$$y_i = \alpha_{j[i]} + \beta_i x_i + \epsilon_i$$

with the α_j terms generally sampled from a normal distribution

$$\alpha_j \sim N(\mu_j, \sigma_j^2), \forall j.$$

The μ_j and σ_j^2 are empirically estimated from the data.

The generalization of this is a varying slope model which can fit random effects on both the intercept and any slopes. A simple varying slope model has the following structure

$$y_i = \alpha_{j[i]} + \beta_{j[i]} x_i + \epsilon_i$$

with both the random slopes and intercepts sampled from different empirically determined normal distributions

$$\alpha_j \sim N(\mu_{\alpha,j}, \sigma_{\alpha,j}^2), \forall j \text{ and } \beta_j \sim N(\mu_{\beta,j}, \sigma_{\beta,j}^2).$$

Gelman and Hill [GH06] outlines a procedure for both nested and non nested multilevel models. Nested linear models occur when a random effect belongs to a hierarchical structure of categories. An example of this in golf is Course/Hole/Round. Each stroke belongs to a specific course. Within the course, each stroke also belongs to a specific hole and round. We would expect the most similar strokes to belong to the same course, hole and round, while two strokes on the same course but a different hole could share some similarity but are not fundamentally related. This type of model is fit by adding additional models on top for each new level of the hierarchical model. The other type of construction is a non-nested model. This simply means that each observation can belong to two categories that are not nested within each other. For example, consider a given stroke and the random effects `Player` and `Course`. Each stroke belongs to a given `Course` and also a given `Player`,

but players each play many different courses and many players play on each course. This is a perfect application of the non nested model structure.

This section will explore the application of varying intercept and varying slope models to stroke-level data, fitting both nested and non nested model structures. The notation that will be used is as follows. For a random intercept term, it will be written (1|`Category`), and a random slope will be (`Feature`|`Category`), where the term before the line dictates the slope or intercept, and the term after the line is the category that varies. A hierarchical will be listed with slashes such as `Category.1`/`Category.2`/`Category.3`. This is modeled off of the `lme4` package in R, and is more quickly interpretable than the mathematical formulation.

Varying Intercept over Course

The first two models I fit were translations of the last two fixed effect models into random intercept models. There are a few practical reasons for this. First, some of categories have relatively few number of observations, increasing the probability that a few outlier shots result in a surprisingly high or low coefficient estimate. This random effect procedure penalizes features that stray far outside of the expected probability distribution. Second, this procedure helped to speed up the cumbersome process of fitting the terms with a high number of coefficients. Finally, this model structure allowed me to specify a hierarchical structure for `Course`, `Hole`, and `Round`, allowing holes to inherit course level estimates and rounds to inherit hole level estimates. As with previous models of this structure, I iteratively introduced `Course`, `Hole`, and

Round to control for overfitting. The formulation of these three models is as follows

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} + (\beta_4 | \texttt{Course}) \quad \texttt{(VI.Course)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} + (\beta_4 | \texttt{Course/Hole}) \quad \texttt{(VI.Hole)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} + (\beta_4 | \texttt{Course/Hole/Round}) \quad \texttt{(VI.Round)}$$

Varying Intercept over Player

This next model is a direct parallel to the last fixed effect model, with a nested varying intercept over Player and Location.Scorer added onto random intercept model with the course parameters.

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} + (\beta_4 | \texttt{Course/Hole/Round}) \quad \texttt{(VI.Player)}$$

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + \beta_3 \texttt{Loc.S} \times \texttt{Dist} + (\beta_4 | \texttt{Course/Hole/Round}) + (\beta_5 | \texttt{Player/Loc.S}) \quad \texttt{(VI.Player.Loc)}$$

As with the last player specific model, both `Player.Agg` and `Location.Scorer.Agg` were tested as well.

Varying Slope over Distance

The last random effect I explored was a varying-slope model where I replaced the interaction between `Location.Scorer` and `Distance` with a varying slope term where `Distance` varies by `Location.Scorer`. This model is represented as follows

$$\texttt{S.R.} \sim \beta_0 + \beta_1 \texttt{Dist} + \beta_2 \texttt{Loc.S} + (\beta_3 \texttt{Dist} | \texttt{Loc.S}) \qquad \texttt{(VS.Dist)}$$

While there are many more extensions of this model structure with the other features in this data set, this class of models has a nonlinear extension that allows for a much more flexible model construction.

3.5 Generalized Additive Models

As I alluded to above, there is a class of generalized linear model that effectively targets a lot of the improvements that are needed for this data structure: generalized additive model. In the abstract a generalized additive model loosens the assumption of a linear relationship between the features and the linear predictor. In the simple Gaussian construction, a predictor y_i is modeled as a function of x_i and v_i using the following formulation

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \epsilon_i$$

with the restriction that f_1 and f_2 are "smooth" functions, normally meaning that they are continuous at some high level of derivative. In practice, these smoothing functions are normally fit using either parametric or non parametric splines, specified by some maximum number of degrees of freedom and a few other spline-specific constraints. These models are implemented in `R` to allow the user to vary the splines by a categorical variable, smooth multiple continuous variables together, or fit tensor interactions to allow for a variety of scales on given continuous variables. In practice, these models are an order of magnitude more complex to fit than standard linear models, and involve both a high level of computational intensity and a magnitude of parameters to manipulate. In the next few sections I will outline the basic structure of the GAM models implemented in the `mgcv` package in `R`[7].

Global Parameters

For a given GAM fit, a few parameters must be specified to determine how the model will be fit. The first parameter, `method`, determines the blanket algorithm that will be used to fit the model. The default value is `"GCV.Cp"` which is a generalized cross validation procedure. The other commonly used method is `"REML"` or restricted maximum likelihood. We have selected a variant of `"REML"` called `"fREML"` which is just an optimized and less robust variation of `"REML"`. Related to this, the next flag I modified was to change the `"discrete"` parameter to `"TRUE"`, a process allowed through `"fREML"` to discretize covariates when possible. The final parameter I modified was `gamma` which assigns the penalty for additional degrees of freedom in the smoothing function (this will be discussed in more detail in the next section).

[7] These parameters are actually specific to the `bam` function, an `mgcv` adaptation of `gam` that is optimized to run on multiple cores in parallel

The research on this topics suggests that a `gamma` of 1.4 is optimal to reduce overfitting. The rest of the values were left at default.

The s() Function

The first type of smooth that can be evaluated in the `gam` function is a spline smooth. The `s()` function is built into the gam function in order to specify the parameters over a given spline smooth. The first parameter is the variable to be smoothed, such as Distance. This can also be two or more variables, in which case a smooth surface is fit in multiple dimensions. The underlying assumption of this type of smooth is that both variables are on the same scale. If this assumption is violated, a tensor smooth must be used as described below. The next most relevant parameter is `bs` which specifies the basis function that the spline will use for fitting. The default for this is `tp` which is a thin plated regression spline, which is a non parametric spline fit. I also experimented with `cr` or cubic regression splines which is the standard parametric spline used in `gam`, and `gp` or gaussian process splines which have some intuitive grounding with spatial data such as a smoother over X and Y because they adjust covariance based on distance apart. The final smoother I tested was `ad` or adaptive smooth, which is more intensive to fit but can adapt the "wiggliness" of the spline based on the value of the covariate in cases where there is irregular point density or variance over a given interval of the variable.

Also note that while it is not really a basis function, `gam` allows the user to pass a basis of `re` which treats the covariate as a random effect. This trick allows the user to specify varying intercepts within the framework of an additive model.

From there I am able to specify `k` which is the maximum degrees of freedom on the spline (this has a different mathematical meaning based on the given spline) and m which corresponds to the order of the penalty term, essentially specifying how "smooth" the curve should be. Finally, there is a `by` variable which allows me to stratify the smooths by a categorical variable. This acts as a categorical interaction but some of the global smoothing properties are inherited for each term.

The te() and ti() Functions

The `te()` and `ti()` functions are full tensor product smooths and tensor product interactions respectively. These functions are designed to take N marginal smooths, and instead of fitting a N dimensional spline, it constructs smooths from the tensor products of the bases[8]. TThe distinction between `te()` and `ti()` is that `ti()` only fits the interaction between the two smooths, while `te()` fits both the marginal smooths and the interactions. A useful way to think about it is through this rough identity over two covariates `x` and `z`

$$\texttt{te(x,z)} \approx \texttt{ti(x)} + \texttt{ti(z)} + \texttt{ti(x,z)}.$$

These functions take similar arguments to `s()`: a marginal basis function, specified through `bs`, an order for the penalty term given through `m`, and a stratification over a factor variable given through `by`.

[8] https://www.rdocumentation.org/packages/mgcv/versions/1.8-28/topics/te.

Tuning the Parameters

The `mgcv` package in `R` also has a built in function called `gam.check` which allows the user to determine if the model would fit better with more degrees of freedom to a certain p value level of certainty. These models can also be tested by standard cross validation of ANOVA techniques.

Relevance

The inspiration for a generalized additive model structure was two-fold. First, perhaps the most obvious improvement over existing golf stroke level models is a rigorous method to vary shot difficulty by distance in a non-linear fashion. The PGA Tour still uses the approach championed by Broadie [Bro12] which fits a series of polynomials to the data. While this is not unreasonable, any textbook on generalized additive modeling will explain why strict polynomials are a sub optimal way to achieve this type of smoothing effect. Secondarily, the PGA Tour provides access to precise coordinate data which is very hard to use in the framework of a standard linear model. There are a few model constructions that deal with data of this type reasonably well, such as Gaussian Process Models, K Nearest Neighbors, and Linear Discriminant Analysis, but after further investigation I found that the ability to fit global distance estimates, specify location based covariance, and still preserve interpretability was unmatched by the generalized additive model.

3.5.1 Building a Baseline

As has been mentioned in previous sections, one of the primary incentives for using GAMs was the ability to fit a non-linear distance function that varies by `Location.Scorer`. A general formulation of this model is as follows

$$\begin{aligned} \texttt{S.R} \sim\ & \texttt{s}(\texttt{Dist}, \texttt{k} = \texttt{k}, \texttt{m} = \texttt{m}, \texttt{bs} = \texttt{bs}) + \texttt{Loc.S} \\ & + \texttt{s}(\texttt{Dist}, \texttt{by} = \texttt{Loc.S}, \texttt{k} = \texttt{k}, \texttt{m} = \texttt{m}, \texttt{bs} = \texttt{bs}) \quad \texttt{(GAM.Baseline)} \end{aligned}$$

There are a few quirks of the GAM model construction to note here. First, the authors of the `mgcv` package recommend fitting both a global smooth of a variable and a stratified smooth over a categorical in order to assign as much of the general deviance to a singular spline as possible. This has an effect similar to a varying-slope model where all of the terms are pulled toward the middle. The second point to note is that the categorical effect for `Location.Scorer` must be fit independent of the stratified smoothing term because the smoothing terms are designed to be centered at 0.

Since this model was the baseline for most of the further research, I took special care to consider all of the plausible parameter combinations to ensure that this model is fit optimally given the constraints. The parameters I considered were k, the number of basis splines, m, the degree of the penalty term, and bs, the type of spline. The values I considered for each parameter were as follows

[!htbp]

k	m	bs
10	2	tp
25	3	cr
50	4	ad
100	5	

I then conducted an abbreviated grid search using out-of-sample $\texttt{R}^2$ to narrow down the top handful of parameter combinations, and then considered a few additional factors to select a final model. Once this model was set, the parameters were fixed for each spline going forward, even when more features were added. I was able to do this because the inclusion of additional terms most likely reduces the degrees of freedom for a simple variable like `Distance` except for some anomalous edge cases. Due to nature of the penalty structure, having a `k` value that is too high is no different than having one at exactly the right value.

3.5.2 Player and Course

Varying-Intercept over Course

Once I had a baseline GAM model, I was able to investigate these random effects in a more rigorous way. Using the same random effect structure I outlined in a previous section, I was able to fit a model with the following specification

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by} = \texttt{Loc.S)} + (1|\texttt{Course}) \quad \texttt{(GAM.Course)}$$

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S)} + (1|\texttt{Course/Hole}) \quad \texttt{(GAM.Hole)}$$

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S)} + (1|\texttt{Course/Hole/Round}) \quad \texttt{(GAM.Round)}$$

Note that it was not computationally feasible to investigate these interactions as fixed effects.

Varying-Intercept over Course and Player

My next objective was to generate preliminary player strength estimates. This helped both with understanding the drawbacks of the models and providing a baseline to measure other player estimates against in the future. For this model, I used both `Course` specific random effects, and `Player/Location.Scorer` random effects. This combination highlights one of the main criticisms of golf analysis models, which is that the strength of the field changes, which makes it hard to estimate course difficulty without knowing player strength. Conversely, it is hard to estimate player strength without understanding relative course difficulty. While some papers have proposed iterative approaches to solve this problem, the default optimal way to work out this interaction is to fit all of the coefficients simultaneously over a large data set. The specification of that model is as follows

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S)} + (1|\texttt{Course/Hole/Round}) + (1|\texttt{Player}) \quad \texttt{(GAM.Player)}$$

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by} = \texttt{Loc.S)} + (1|\texttt{Course/Hole/Round}) + (1|\texttt{Player/Loc.S}) \quad \texttt{(GAM.Player.Loc)}$$

During my analysis of player rankings, I noticed that player's abilities were strongly correlated across location type. This was probably due to the hierarchical nature of the final model structure. In reality, player ability by location is probably somewhere between highly correlated and totally uncorrelated. Because of this, I fit the model without the global `Player` coefficients and checked it against the other two models. This model had the following specification

$$\texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by} = \texttt{Loc.S)} + (1|\texttt{Course/Hole/Round}) + (1|\texttt{Player} \times \texttt{Loc.S}) \quad \texttt{(GAM.Loc)}$$

An extension of this research could be to find a method that would allow for more hands-on tuning of the pooling parameters.

Incorporating Time of Day

There is one specific continuous variable that I wanted to investigate at this point because it was plausibly highly correlated to both course and player effects: `Time`. The `Time` variable measures the local time that the shot was taken, and has long been hypothesized to be correlated to shot difficulty because it is a good proxy for moisture levels on the course. There are also some weather effects that may be more concentrated during different times of the day. This variable has a lot of implications for tournament organizers who have to determine a tee off schedule that is the most equitable.

On top of these climate effects, there are also some tournament structure rules that influence shot difficulty over time of day. During the last two days of most golf tournaments, players tee off in an order relative to their position in the tournament. This means that studies looking at the impact of `Time` diagnostically without any controls could easily be tainted. Fortunately, in the previous section I outlined a method to control for most global course and player effects. Fitting a smooth spline over `Time` in conjunction with the varying-intercept model from the previous section was able to give me a much more sophisticated understanding of the impacts of `Time` on a shot in golf. The full model form is as follows

$$\begin{aligned} \texttt{S.R} \sim \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by} = \texttt{Loc.S)} + (1|\texttt{Course/Hole/Round}) \\ + (1|\texttt{Player/Loc.S}) + \texttt{s(Time, k} = \texttt{k)} \quad \texttt{(GAM.Time)} \end{aligned}$$

Throughout the remainder of the paper, I defaulted to inheriting the basis spline and penalty degree [9] from the cross validation of the baseline model, tuning only `k`.

On top of this simple spline over `Time`, I took this temporal analysis a bit further to consider change in difficulty over different values for `Location.Scorer`. When thinking about the implication of dew on a golf shot, it should probably have a larger relative effect in areas with longer grass on the primary and intermediate rough. I also thought it would make sense to see some variation on bunker and green because these surfaces probably interact with water in a much different way than traditional fairway grass. To put this hypothesis

[9] `bs=ad` and `m=5`

to the test, I fit the model above with the addition of a term to varying the `Time` spline by `Location.Scorer.Agg`[10]. The model formula is as follows

$$\begin{aligned} \texttt{S.R} \sim\ & \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S)} \\ & + (1|\texttt{Course/Hole/Round}) + (1|\texttt{Player/Loc.S}) \\ & + \texttt{s(Time, k = k)} + \texttt{s(Dist, by = Loc.S)} \quad \texttt{(GAM.Time.Loc)} \end{aligned}$$

For this model, I inherited the `k` value selected in the previous model for simplicity[11].

The final variation of this model that I considered was `Time` versus `Distance`. While I hoped that most of the temporal variation could be attributed to `Location.Scorer`, I was still curious to see if the moisture levels of the green would have more of an impact on a 200-yard fairway shot than a 100-yard fairway shot[12]. To measure this impact, I fit a tensor interaction between `Time` and `Distance`. This was done because the two variables exist on different scales, and I had already conducted a marginal smooth on each independently. This model has the following formula

$$\begin{aligned} \texttt{S.R} \sim\ & \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S)} \\ & + (1|\texttt{Course/Hole/Round}) + (1|\texttt{Player/Loc.S}) + \texttt{s(Time, k = k)} \\ & + \texttt{s(Dist, by = Loc.S)} + \texttt{ti(Time, Dist)} \quad \texttt{(GAM.Time.Dist)} \end{aligned}$$

[10] The sample for the less frequent locations is too small to fit meaningfully over both time and distance.

[11] These models take an hour or more to fit.

[12] Many golfers talk positively about a wet green because the ball lands more firmly upon impact, making it easier to target a shot from far away. This seems to have the most effect when shooting onto the green.

3.5.3 Smoothing Over Effective Green and Distance

For effective green I engineered four attributes, `Eff.Green.Front`, `Eff.Green.Back`, `Eff.Green.Left`, `Eff.Green.Right`. The intuition behind this is multifaceted. First, it is interesting I had to consider a golfer looking at the green from far away. The golfer would like to see space on either side so that he can miss wide, but also a lot of space in front of the hole to bounce before and space behind in case of an overshot. The next issue to consider was that a golfer is potentially more likely to miss left-right when they are further out, and relatively more likely to miss front-back when they are closer in. This would suggest that these coefficients would vary with distance away from the hole. Third, as observed by Levin [Lev17], golfers are not able to get as much spin when hitting from the rough. This means that in the rough it would be more beneficial to have a lot of space in the front of the tee than in the fairway. Finally, these relationships are unlikely to be linear. It seems that the difference between 3 yards and 8 yards of green in front of the hole is far more important than the difference between 10 and 15 yards.

Armed with these intuitions, I fit three models, each a stepwise improvement on the previous one, to incorporate the first three intuitions about how these covariates might affect shot difficulty. The first model is as follows

$$\begin{aligned} \texttt{S.R} \sim\ & \texttt{s}(\texttt{Dist}) + \texttt{Loc.S} + \texttt{s}(\texttt{Dist}, \texttt{by} = \texttt{Loc.S})) \\ & + \texttt{s}(\texttt{Eff.Green.Front}, \texttt{k} = \texttt{k}) + \texttt{s}(\texttt{Eff.Green.Back}, \texttt{k} = \texttt{k}) \\ & + \texttt{s}(\texttt{Eff.Green.Right}, \texttt{k} = \texttt{k}) + \texttt{s}(\texttt{Eff.Green.Left}, \texttt{k} = \texttt{k}) \quad \texttt{(GAM.EG)} \end{aligned}$$

For sake of simplicity, I decided to inherit the optimized `bs` and `m` from the baseline model. From there, I fit k over the set of 15, 25, 50, and then used that value for the remainder of the effective green terms. Once I had fit this initial model, I refit adding four more terms to dictate the interaction between `Eff.Green` and `Distance`

$$\begin{aligned}
\texttt{S.R} \sim\ & \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S))} \\
& + \texttt{s(Eff.Green.Front, k = k)} + \texttt{s(Eff.Green.Back, k = k)} \\
& + \texttt{s(Eff.Green.Right, k = k)} + \texttt{s(Eff.Green.Left, k = k)} \\
& + \texttt{ti(Eff.Green.Front, Dist, k = k)} + \texttt{ti(Eff.Green.Back, Dist, k = k)} \\
& + \texttt{ti(Eff.Green.Right, Dist, k = k)} \\
& + \texttt{ti(Eff.Green.Left, Dist, k = k)} \quad \texttt{(GAM.EG.Dist)}
\end{aligned}$$

Note that these are tensor interactions because I have already fit the marginal smooths and these terms are on different scales. Finally, I added four more terms to analyze the relationship between `Location.Scorer` and `Eff.Green`

$$\begin{aligned}
\texttt{S.R} \sim\ & \texttt{s(Dist)} + \texttt{Loc.S} + \texttt{s(Dist, by = Loc.S))} \\
& + \texttt{s(Eff.Green.Front, k = k)} + \texttt{s(Eff.Green.Back, k = k)} \\
& + \texttt{s(Eff.Green.Right, k = k)} + \texttt{s(Eff.Green.Left, k = k)} \\
& + \texttt{ti(Eff.Green.Front, Dist, k = k)} + \texttt{ti(Eff.Green.Back, Dist, k = k)} \\
& + \texttt{ti(Eff.Green.Right, Dist, k = k)} + \texttt{ti(Eff.Green.Left, Dist, k = k)} \\
& + \texttt{s(Eff.Green.Front, by = Loc.S, k = k)} \\
& + \texttt{s(Eff.Green.Back, by = Loc.S, k = k)} \\
& + \texttt{s(Eff.Green.Right, by = Loc.S, k = k)} \\
& + \texttt{s(Eff.Green.Left, by = Loc.S, k = k)} \quad \texttt{(GAM.EG.Loc)}
\end{aligned}$$

3.6 Course Visualizations

In a traditional sense, statistical inference is focused on realizing information about the relationship between specific covariates and the dependent value through the fitted parameters of the model and the error on the estimate. One question that may come up is "given a specific golfer, 150 yards from the pin, what can we say about the relative difficulty of the shot on the fairway, intermediate rough, and primary rough?" The answer to this problem however, gets much more complicated when considering spatial coordinates and location data. A golf course is deterministically defined through spatial dependence. For a given `X`, `Y`, (`Z`), coordinate on a hole, there is exactly one `Location.Scorer`, one `Distance`, one set of values for `Eff.Green`, etc. In previous papers, this problem has not been a huge issue because if the domain on distance is restricted, there is a reasonable population of fairway, intermediate rough, and primary rough shots between 100 and 200 yards to make an unbiased estimate of the shot difficulty.

With the addition of features more closely linked to the `X`, `Y` coordinates of the shot, there are a few strategies that can be attempted to make sense of the model results. The first, and simplest approach is to essentially ignore the location based terms. The relationship between different location types, distances, etc, can be analyzed with the spatial information treated as merely a control to make the estimates more accurate.

Assuming we would like to learn a little more about the spatial structure of the course, the second possible method is to analyze only points that are "in sample." This means that only `X`, `Y` coordinates that have been the location of a previous shot on the tour can be estimated. This works because these shots already have the feature set included, i.e. their `Location.Scorer`, `Distance`,

Distance.to.Center, etc. After these estimates are made, a surface could probably be fit to estimate the difficulty of the shots in between. This method leaves a lot to be desired, as it is based on a very elementary understanding of the structure of the course.

The third method, which will be introduced for the first time in this paper, is to try to estimate the features for every playable X, Y on every hole of every golf course. Thanks to the estimation of the Hole.Location, and a K Nearest Neighbors algorithm, this can be done in a relatively accurate fashion for most of the features.

3.6.1 Building the Course Features

This algorithm takes inputs of a specific hole on the PGA Tour, and a round on which the hole is to be played. From there, the model isolates every shot taken on the hole that exists in the data, and runs a convex hull algorithm to wrap the course in a tight polygon. I wanted to limit the drastically out of sample points that it had to predict, without sacrificing any location that a golfer may feasibly visit.

Next, the polygon is filled with a mesh grid of points such that they are evenly spaced within the course. This normally results in 20,000-40,000 points to predict per hole. From here, a K Nearest Neighbors algorithm is fit on the individual location types to the points that have been played on the given hole. This algorithm is then able to predict the Location.Scorer of all of the out of sample points.

Along with predicting the Location.Scorer, the model will also return a confidence on the prediction. While predicting over the convex hull, I wanted

to know where the model had a strong understanding of the course structure, and where the model was under confident. This helped for both smoothing predictions, and creating more accurate visualizations down the line.

Once a map of the location attributes has been fit for a given hole, the distance to the hole must be estimated. For this, the round must be taken into account because each round has a different hole location. Since the hole location has been inferred in the data cleaning section, this is as simple as calculating the pythagorean distance between the given point and the hole.

3.6.2 Visualizing the Hole Layout

Once this data set had been constructed, a few interesting visualizations were possible. First, the playable part of the course that is not the tee box can be visualized purely from the ShotLink data. Two examples can be seen below, from TPC Sawgrass holes 2 and 5. Note in the example that the transparency of the pixel is a function of the confidence of the classification.

Comparing these two projections to the satellite images of the courses in 2019, as done in Figure 3.2 and Figure 3.3, we can see that the model is not perfect, but shows a lot of promise. An accuracy metric for this algorithm based on sampling the pixels in the image would be useful but is beyond the scope of this paper.

3.6.3 Visualizing the Model Predictions

Now that the course is understood to some degree of certainty, I was able to visualize models over the new sample of data. The models using this tool are

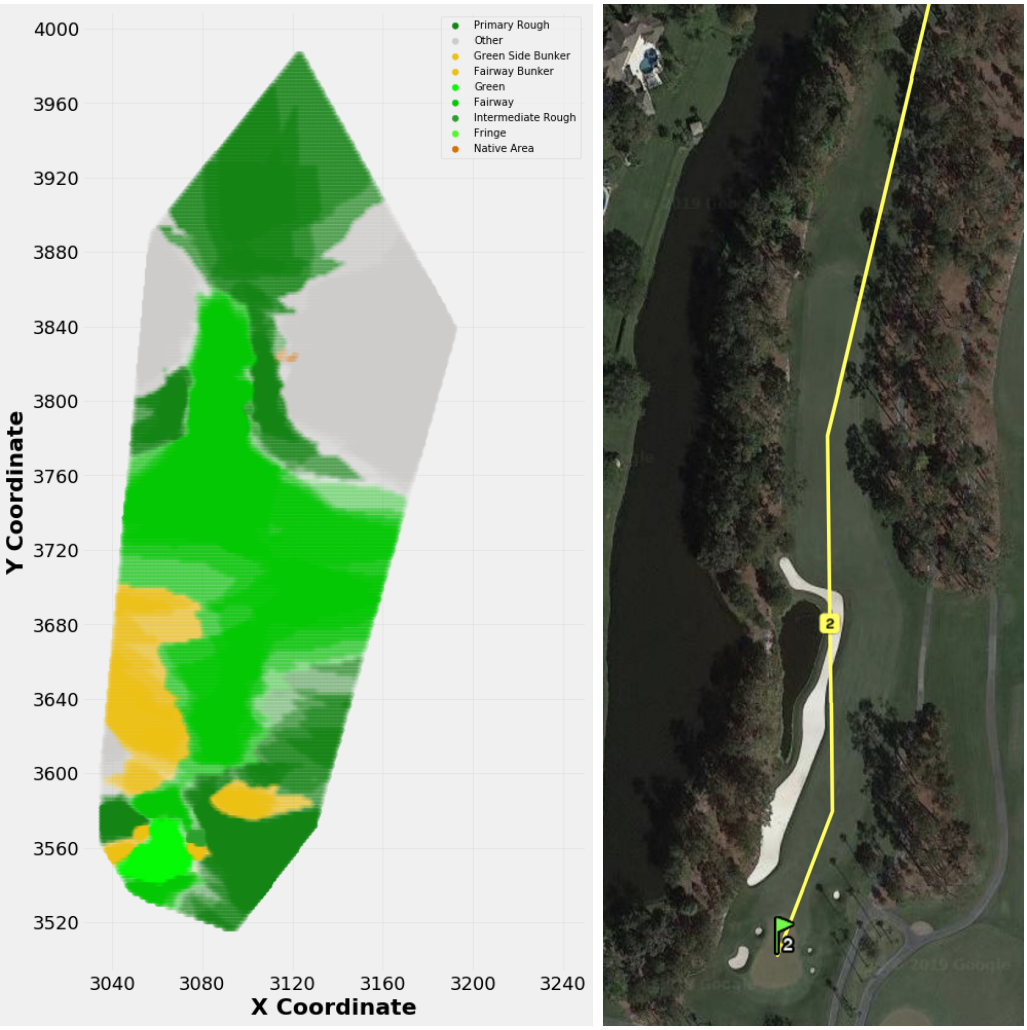

Figure 3.2: The left is an estimate of the location attributes of TPC Sawgrass, Hole 2 using a K Nearest Neighbors algorithm, and shaded based on confidence of the classification. The right is an aerial view of the hole taken from Google Maps.

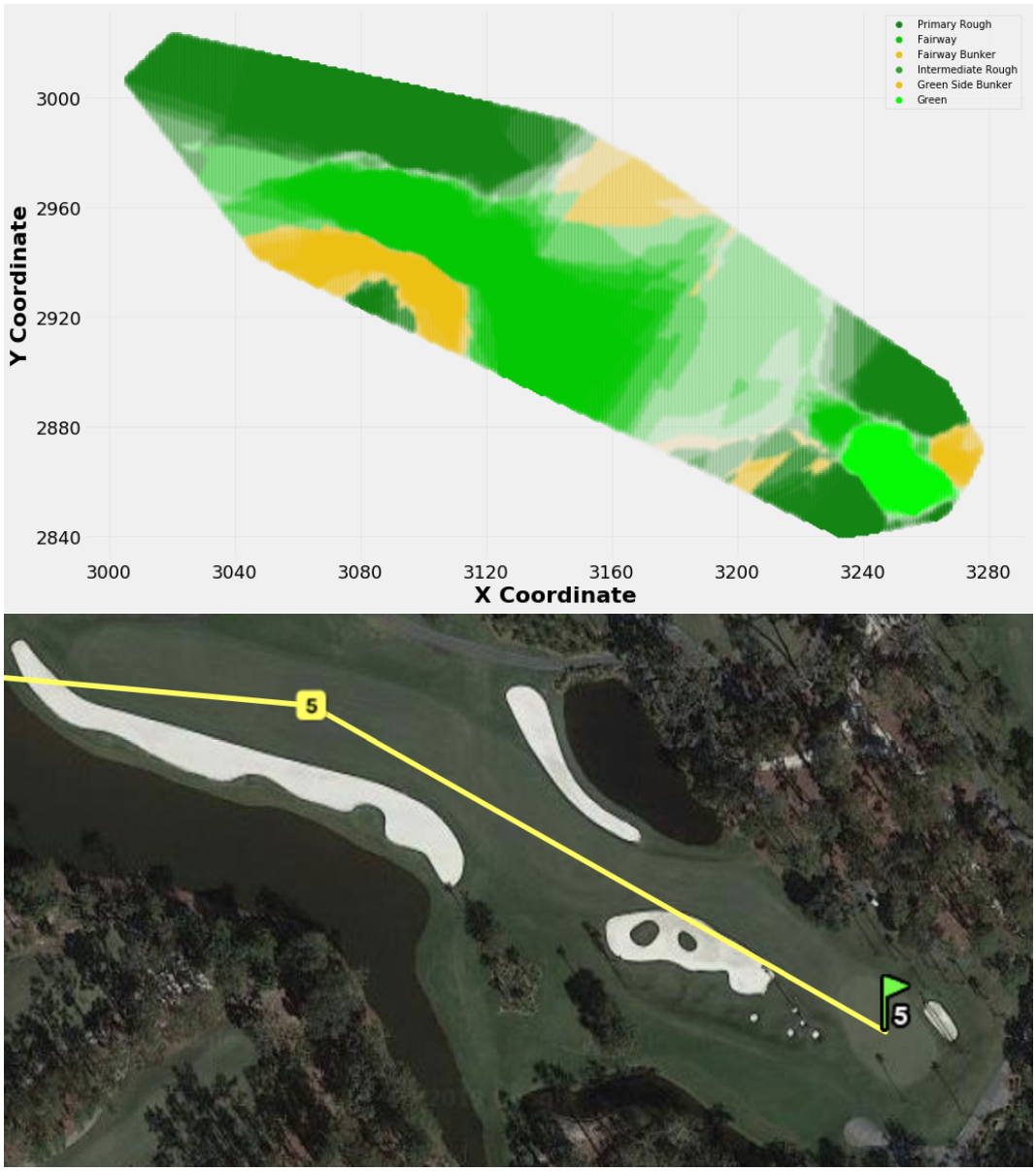

Figure 3.3: The left is an estimate of the location attributes of TPC Sawgrass, Hole 5 using a K Nearest Neighbors algorithm, and shaded based on confidence of the classification. The right is an aerial view of the hole taken from Google Maps.

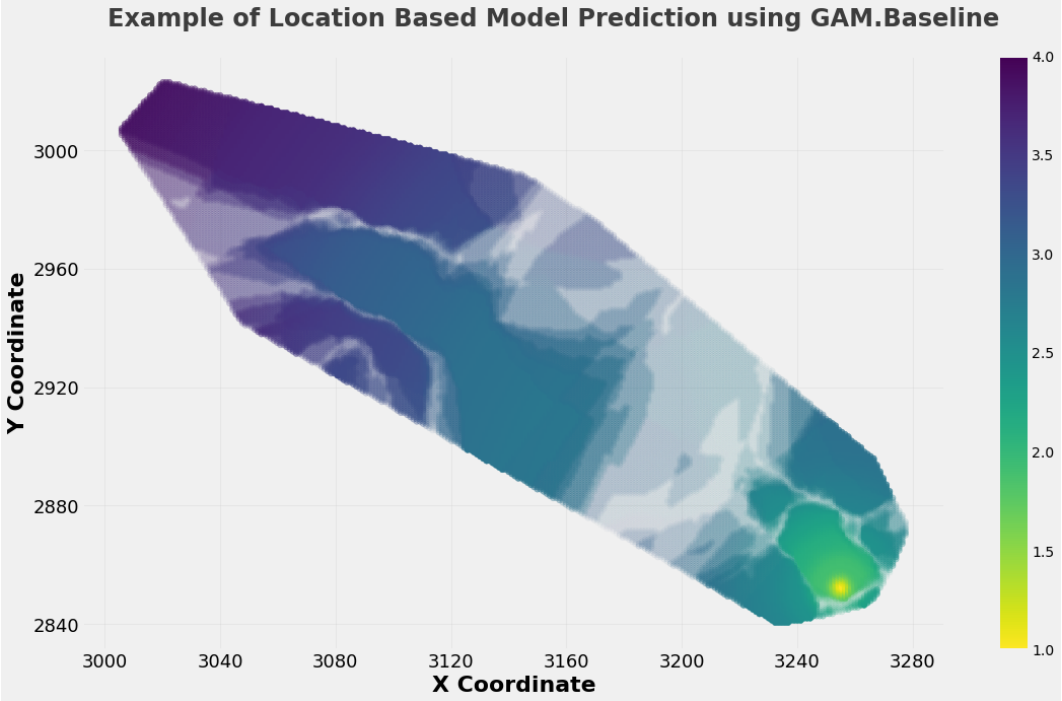

Figure 3.4: An estimate of shot difficulty for every location on TPC Sawgrass Hole 5 for Round 1 in 2018. This difficulty is calculated using `GAM.Baseline` which will be described in a later section.

restricted to location attributes of `Location.Scorer`, `X`, `Y`, `Eff.Green`, and `Distance`. Other, non deterministic features can be added in selectively to visualize specific attributes. For example, this course visualization could be made for a specific player or time of day.

When visualizing the model, the data generated was fed into the model prediction function to get the mean `Strokes.Remaining` estimate. If there was any uncertainty in the previous step as to the specific `Location.Scorer` category at a given point, a weighted average of the possible values was taken and the visualization was faded on the graph relative to the uncertainty. This makes the model fit more smoothly on location boundaries, and allows the viewer to quickly understand where the model is reliable and where it not. An example of this process over the `GAM.Baseline` is shown in Figure 3.4.

As we can see, even with a relatively unsophisticated model, we begin to see a lot of spatial structure in the data using this visualization technique. These ideas will be explored in more detail an on a larger set of models in the discussion section.

3.7 Computational Efficiency

Due to the volume of data collected on the PGA Tour, this problem is computationally complex even with some of the simpler model constructions. Because of this, a large part of the underlying research for this problem was model optimizations for the different classes. The optimizations that I made in this problem fell broadly into three categories. First, I tried to speed up the model optimizers and reduce the dimensionality of the data as much as possible without compromising the model accuracy. Second, I used some parallelization packages and strategies to run these models on multiple cores in order to use the maximum amount of available compute possible. Third, I scaled up my RAM using cloud computing on AWS in order to fit larger vectors and more complex models that exceeded the capacity of my personal computer. The combination of these three improvements allowed me to take a functionally intractable problem and run cross validation in many cases in less than a few hours.

3.7.1 Model Optimizations

Linear Models

The initial set of linear models can reasonably be run on a 16 GB personal computer without much concern.

Linear Mixed Effect Models

The pure mixed effect models were difficult to optimize as there is not an out of the box function that fit optimization more quickly than the standard `nlme` or `lme4` packages. Because of this, I targeted improvements that fit within the already existing model framework. The first of these was to remove the derivative calculation from the model fitting process. This required a lot of memory and did not add value to the predictions I was performing. The second was to use the `nloptwrap`[13] optimizer in order to speed up convergence of the model. Overall, varying-coefficient models do not take prohibitively long to fit, but these improvements were not enough to speed up the feasibility of varying-slope category of model.

Generalized Additive Models

Generalized additive models are a highly customizable model structure in R, and because of this there are many adjustments that can be attempted to speed up model fitting in certain ways. The biggest problem with GAMs

[13] cite this?

is that the degrees of freedom can get quite large from complex splines over large sets of features. As mentioned above, this can be combated by artificially suppressing the k value on some of the splines in order to reduce the degrees of freedom.

Another optimization that I made was switching the optimization method for the splines from `GCV.Cp` to `fREML`, which stands for fast restricted maximum likelihood. I ran a few diagnostics to confirm that there was not a notable loss in accuracy from this switch. This switch alone sped up model fitting by nearly an order of magnitude.

3.7.2 Parallelization

While I was able to speed up the `gam` functions a reasonable amount purely through model optimization, the real speed boost came from the `bam` package designed to fit gam models mostly in parallel. By passing in the full set of cores on a given machine, the `bam` package is able to fit a large part of the model using the full computational power of the hardware[14].

3.7.3 Hardware Upgrades

Once I had exhausted the local optimizations, the final problem I faced was that the GAM function often tried to allocate a vector that exceeded the maximum size allowed. It is possible to edit the maximum amount of RAM allowed by the program up to the size of the memory on a given piece of hardware, but some of the larger models need well over 16 GB vectors to fit properly. This required me to purchase compute power off of AWS. AWS

[14]This package sits on top of the `parallel` package in `R` for multicore analysis

has a set of memory optimized linux instances that can be configured to run `RStudio`. The instances that I relied on for most of the analysis were the `r5d.2xlarge` and `r5d.4xlarge` with 64 and 128 GB of RAM respectively. These ran 8-12 cores and sped up my model fitting by roughly another 2-3x. A possible area for improvement for this process could be to either find hardware that is better suited for this specific model framework, or determine if it is possible to fit a generalized additive model on a GPU.

4 Results

4.1 Linear Models

4.1.1 Simple Fixed Effect Models

Recall from the earlier section that four fixed effect linear models were built to build out a baseline for the simple linear relationships involved in this modeling process. The first was a distance only model (`LM.Dist`), the second was a baseline where distance varied by location type (`LM.Baseline`), and the third considered iteratively included interaction terms to specify `Course` (`LM.Course`), `Hole` (`LM.Hole`), and `Round` (`LM.Round`), and a final that iteratively added player effects (`LM.Player`) and player effects varied by location (`LM.Player.Loc`).

Once these models were fit, I put them through five fold cross validation and measured out of sample across different location types by R^2. I was

Location	LM.Dist	LM.Baseline	LM.Course	LM.Hole	LM.Round
Green	-.037	.485	.485	–	–
Fairway	.063	.308	.310	–	–
Rough	-.035	.391	.392	–	–
Bunker	-.507	.336	.337	–	–
Other	-.548	.188	.190	–	–
Recovery	-1.248	-.573	-.601	–	–
Total	**.565**	**.751**	**.751**	–	–
Fitting Time	*.52 sec*	*1.7 sec*	*5.1 sec*	–	–

Table 4.1: Out of sample R^2 over a series of simple linear models with fixed effects. I was not able to fit a single iteration for two of them.

only able to fit the final two models using `speedglm`, a package that does not natively support prediction. Because of this, I did not proceed to the more computationally complex player models. This fact on its own is enough justification to switch to the faster varying-intercept models.

4.1.2 Mixed Effects Models

The next class of models were a series of models I fit substituting hierarchical random intercepts in for the sparse categorical variables over `Course` (`VI.Course`), `Hole` (`VI.Hole`), `Round` (`VI.Round`), and then `Player` (`VI.Player`) and `Location.Scorer` (`VI.Player.Loc`). I also fit one varying-slope model of `Distance` over `Location.Scorer` (`VS.Dist`). The out of sample R^2 is as follows

Location	LM.Baseline	VI.Course	VI.Hole	VI.Round
Green	485	.485	.486	.487
Fairway	.308	.310	.315	.321
Rough	.391	.392	.396	.401
Bunker	.336	.337	.342	.347
Other	.188	.189	.195	.202
Recovery	-.573	-.598	-.601	-.582
Total	**.751**	**.751**	**.752**	**.754**
Fitting Time	*1.7 sec*	*6 sec*	*19 sec*	*47 sec*

Table 4.2: Out of sample R^2 for mixed effect models over course effects. These models steadily improved in prediction accuracy with increased granularization.

4.2 Generalized Additive Models

Location	LM.Baseline	VI.Player	VI.Player.Loc	VS.Dist
Green	485	.487	.487	.485
Fairway	.308	.321	.324	.309
Rough	.391	.401	.404	.391
Bunker	.336	.347	.351	.336
Other	.188	.201	.208	.188
Recovery	-.573	-.578	-.574	-.571
Total	**.751**	**.754**	**.755**	**.751**
Fitting Time	*1.7 sec*	*18 sec*	*45 sec*	*9 sec*

Table 4.3: Out of sample R^2 for mixed effect models over player effects and distance. The player effects did not contribute to increased prediction accuracy, and the varying slope model was not an improvement over baseline.

4.2.1 Building a Better Baseline

To build a better baseline model using smoothing splines I used the model outlined in the methods section, and conducted a grid search over the parameter set outlined. Since these models are not computationally trivial to fit, I did not test every parameter set but instead conducted a few diagnostic tests to learn more about each parameter. After testing the sensitivity of the `bs`, `k`, and `m` to out of sample R^2[1], I found that the selection of smoothing spline was by far the most consequential, and that an additive smoother had the best performance. This makes sense intuitively because while other splines have semi constant penalty within the feature, adaptive splines allow for this quantity to vary. This means that at distances with high density (on and around the green), the spline can have more flexibility, and deep into the fairway it can vary much less.

Continuing the sensitivity analysis within the remaining two features, I first considered the effect of varying k while leaving m at the default of 5. This result suggests a `k` of 100 as the optimal value. I tested higher than this and

[1] It is worth noting here that each smoothing spline is significant to $***$ in R regardless of the parameter set, so I did not give this much weight

k	m	R^2
10	5	.772
25	5	.773
50	5	.773
100	**5**	**.774**

Table 4.4: Change in prediction accuracy from varying `k` for `GAM.Baseline`.

saw no improvement. This result was reenforced by `gam.check` on k=50 which suggests that a k increase is needed. However, in the process of running `gam.check` I realized that the broad `Distance` spline needed more degrees of freedom than the varying splines by distance. Because of this I settled on `k` = 100 for the single `Distance` spline and `k` = 50 for the varying splines. Next I checked the sensitivity of this result to m. This term seems relatively

k	m	R^2
100/50	3	.774
100/50	5	.774
100/50	7	.774
100	**5**	**.774**

Table 4.5: Change in prediction accuracy from varying `m` for `GAM.Baseline`.

insensitive for an adaptive spline and so the default value of 5 was used.

Here I will note that a grid search in this manner is not robust, and leaves room for odd tail behavior that could generate an absolute maximum that I did not find. Through my experience fitting these models, I found that this progressive order of tuning worked well and that these parameters infrequently had high derivatives relative to fit quality.

With the establishment of this model (`GAM.Baseline`), I was able to compare all of the model structures I had fit considering only `Distance` and `Location.Scorer`

Location	LM.Dist	LM.Baseline	VS.Distance	GAM.Baseline
Green	-.037	.485	.485	.604
Fairway	.063	.308	.309	.318
Rough	-.035	.391	.391	.407
Bunker	-.507	.336	.336	.340
Other	-.548	.188	.188	.199
Recovery	-1.248	-.573	-.571	-.581
Total	**.565**	**.751**	**.751**	**.774**
Fitting Time	*.52 sec*	*1.7 sec*	*9 sec*	*17 sec*

Table 4.6: A comparison of out of sample R^2 across a series of baseline models fit primarily as a function of `Distance`.

This model clearly picks up additional information over the linear model unilaterally across the course. The effect is most concentrated on the green, where a linear effect misses the most additional information.

4.2.2 Course Effects

Next, I refit the random effects over `Course` (`GAM.Course`), `Hole` (`GAM.Hole`), and `Round` (`GAM.Round`), this time with a non-linear function for `Distance`.

Location	GAM.Baseline	GAM.Course	GAM.Hole	GAM.Round
Green	.604	.604	.604	.605
Fairway	.318	.319	.324	.331
Rough	.407	.408	.411	.417
Bunker	.340	.341	.346	.351
Other	.199	.200	.204	.210
Recovery	-.581	-.600	-.598	-.581
Total	**.774**	**.774**	**.775**	**.777**
Fitting Time	*17 sec*	*18 sec*	*43 sec*	*12 min*
Size	*73 Mb*	*77 Mb*	*125 Mb*	*643 Mb*

Table 4.7: A comparison of out of sample R^2 across generalized additive models that incorporated course based random effects into the baseline model. The accuracy increased nearly unilaterally as granularity of the features was increased.

4.2.3 Player Effects

On top of these `Course` effects I fit three additional `Player` models, one considers only `Player` as a random effect (`GAM.Player`), one fitting a hierarchical structure between `Player` and `Location.Scorer`, and one considering only the interaction between `Player` and `Location.Scorer` (`GAM.Loc`)

Location	`GAM.Baseline`	`GAM.Player`	`GAM.Player.Loc`
Green	.604	.606	.606
Fairway	.318	.332	.333
Rough	.407	.418	.419
Bunker	.340	.353	.354
Other	.199	.214	.22
Recovery	-.581	-.573	-.577
Total	**.774**	**.777**	**.778**
Fitting Time	*17 sec*	*11 min*	*34 min*
Size	*73 Mb*	*700 Mb*	*1.1 Gb*

Table 4.8: A comparison of out of sample R^2 across generalized additive models that incorporated player effects into the already existing course effect models.

4.2.4 Time of Day Effects

Finally, I added `Time` effects to the top of these existing `Course` and `Player` models (`GAM.Time`), varying `Time` by both `Location.Scorer` (`GAM.Time.Dist`), and `Distance` (`GAM.Time.Dist`). I did not run cross validation on these models due to the fitting times involved. Instead I ran an ANOVA test on the three models to determine if the iterative models were an improvement. I found that both model iterations starting from `GAM.Time` and going up to `GAM.Time.Loc` were statistically significant to at least `2e-15` in `R`. While random effects can distort these tests, since the model is only using smoothing splines in both of the iterative updates, it should not be a concern.

4.2.5 Impact of Effective Green

The next set of models I fit were more explicitly focused on location structure. I fit three `Eff.Green` models on top of the `GAM.Baseline`, one smoothing over all four `Eff.Green` attributes additively (`GAM.EG`), one varying `Eff.Green` by `Distance` (`GAM.EG.Dist`), and one varying by both `Distance` and `Location.Scorer` (`GAM.EG.Loc`) I found that a `k` value of 15 was suffi-

Location	GAM.Baseline	GAM.EG	GAM.EG.Dist	GAM.EG.Loc
Green	.604	.604	.604	.600
Fairway	.318	.318	.321	.321
Rough	.407	.411	.414	.414
Bunker	.340	.344	.347	.347
Other	.199	.206	.214	.211
Recovery	-.581	-.571	-.559	-0.546
Total	**.774**	**.775**	**.775**	**.774**
Fitting	*17 sec*	*22 sec*	*1.5 min*	*2 h 43 min*
Size	*73 Mb*	*98 Mb*	*152 Mb*	*219 Mb*

Table 4.9: A comparison of out of sample R^2 across models that incorporated varying levels of smoothing effects on the set of `Eff.Green` variables.

cient for all splines to describe the full feature space. Note that `Eff.Green` was not calculated for the green and so I did not expect to see an improvement there.

5 Discussion

5.1 Exploring the Baselines

After establishing the benefits of fitting stroke difficulty against `Distance` and `Location.Scorer` in a nonlinear fashion, I had the tools to further investigate what this relationship looks like. It is hard to see the impact of this change from merely the R^2 because so much of the data is concentrated either extremely close to the hole on the green or far back in the fairway where the relationships are both locally linear. However, plotting these relationships stratified by `Location.Scorer` for both `LM.Baseline` and `GAM.Baseline` highlights the magnitude of this change in assumption, as seen in Figure 5.1 and Figure 5.2. The contrast between these two plots is stark. Figure 5.2 highlights the broad shape of shot difficulty versus distance. The expected strokes soars over the first 10-15 yards, and then levels out to roughly linear as the distance gets beyond 50 yards. There are a few trends to note here. The most striking is a large and uniform penalty for ending a shot in the primary rough, compared to the intermediate rough which seems essentially insignificant. The other curious aspect of this plot is the relative difficulty of the green side bunker, both compared to distance and compared to primary rough. We see a clear advantage to being in the bunker as opposed to the rough between approximately 15 and 30 yards, but as the ball gets closer to the hole the rough gets much easier while the bunker does not. One possible explanation for this is that the rough limits the amount of spin a golfer can get on the ball compared to the sand, which is critical for approach shots. However, bunkers

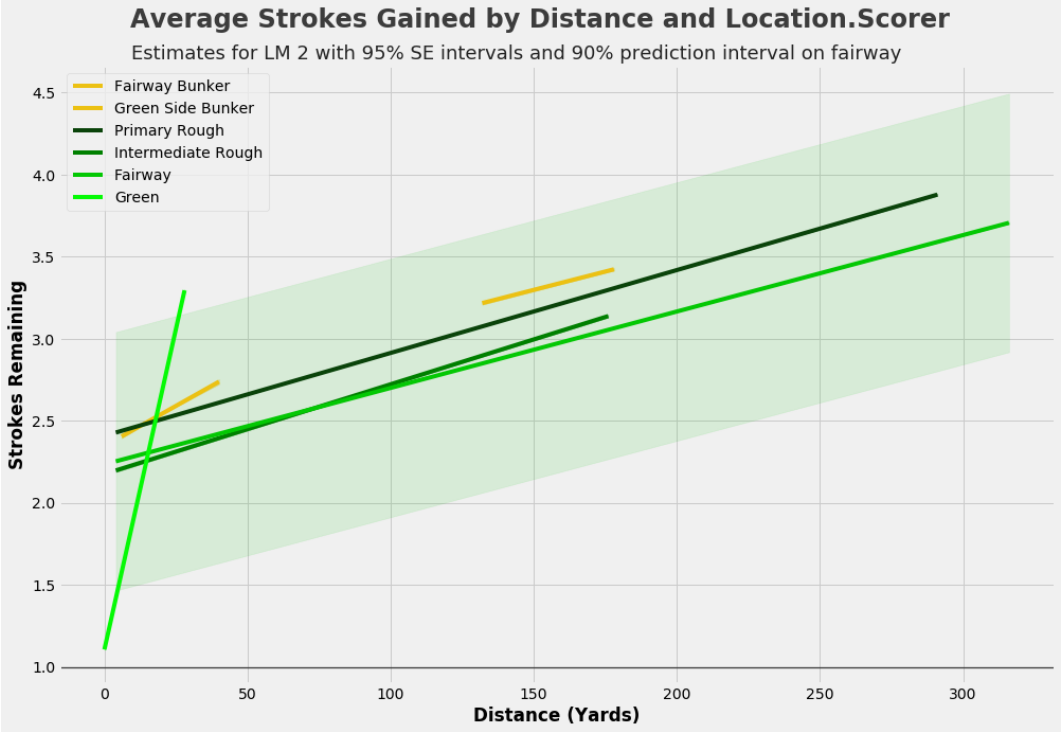

Figure 5.1: A visualization of the linear relationship between `Distance` and shot difficulty in `LM.Baseline`.

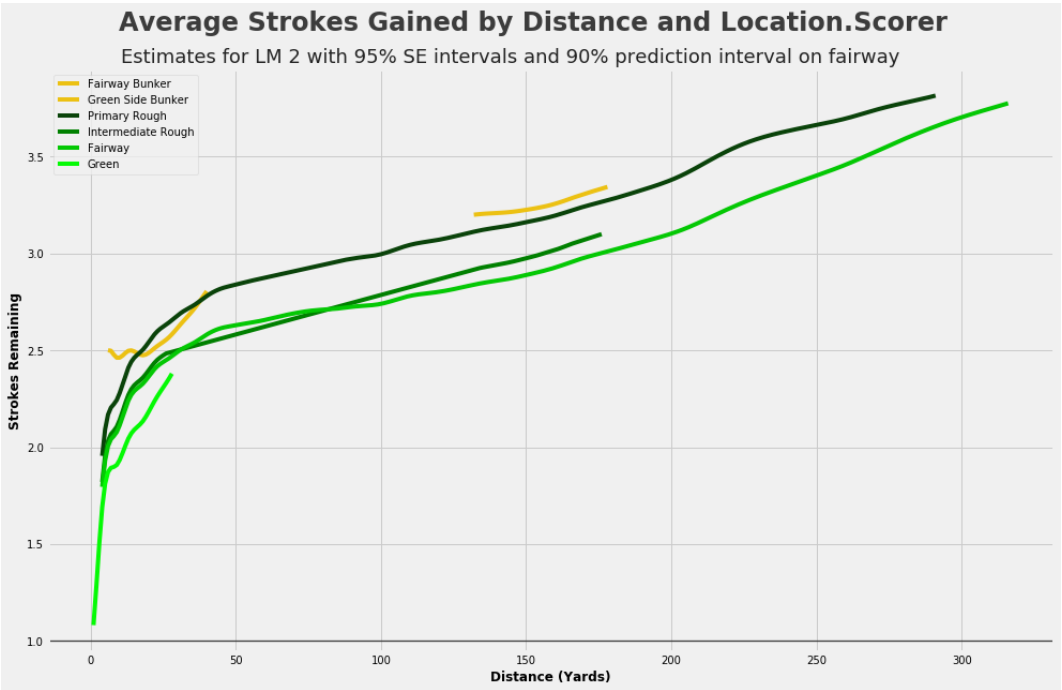

Figure 5.2: A visualization of the smoothed relationship between `Distance` and shot difficulty in `GAM.Baseline`, including 95% SE intervals.

close to the hole often have the characteristic that they are much deeper than the rough, making a close chip shot still extremely difficult.

Finally, it is worth noting that these observed distances do not exist in a vacuum. It may strike the reader as odd that there are slight wiggles in these curves, e.g. the smooth over primary rough between 200 and 250 yards. This, combined with the tiny standard errors, has some odd implications without any obvious answers. I will offer a few plausible explanations. First, there may be certain courses where it is common to end up 230 yards away, and these courses are more difficult than average. With so much structural data, it would not be surprising to see some of these effects. Second, certain clubs may just be more difficult for golfers categorically. Third, there is some location bias in the quality of golfers that shoot from certain distances. Long drivers end up with closer approach shots, and skill across shot type is most certainly correlated. Of these three, `Course` and `Player` effects were partially controlled for in some of the more complex models, while club effects were not possible for me to measure.

5.2 Course Difficulty Rankings

When analyzing the impacts of certain variables through statistical inference, it is common practice to use the most complex model set of features that provide a statistically significant benefit to decreasing the deviance of the model. Through cross validation and the pairwise ANOVA tests in the results section, I settled on `GAM.Time.Dist` because it allowed me to control for player strength and time of day in a rigorous manner when rating course difficulty. One benefit of controlling for player strength is that stronger players are more likely to make the cut and play in the last two rounds of a

standard golf tournament. Because of this, a simpler model is exposed to the bias of rating the later rounds as easier and the earlier rounds as harder. The hierarchical nature of this model should control for this a bit by grouping the round coefficients for a given hole, but nonetheless a naive model would likely confuse these two impacts. To investigate whether my models had solved this problem, I looked at the average contribution of a given round under the `GAM.Round` which has no `Player` effects, and `GAM.Time.Dist` which is the full model controlling for all aspects of the game, as shown in Table 5.1. This shows across the board tiny levels of inter-round bias. If there is anything

	`GAM.Round`	`GAM.Time.Dist`
Round 1	.0005	-.0014
Round 2	.0041	.0028
Round 3	-.0044	-.0033
Round 4	-.0001	.0020

Table 5.1: Bias in shot difficulty estimation by `Round` for `GAM.Round` and `GAM.Time.Dist`.

to glean from this, it is that the the additional covariates reduce the bias by roughly 2-3x, but also that some bias might actually exist. It is not unreasonable to argue that players are rusty and may play slightly worse during Round 1, play harder during Round 2 to make the cut, then relax for Round 3 and try hard again with money on the line during Round 4. This is purely speculation though, and with the obvious biases controlled for it will be left up to future research to investigate this effect on a deeper level.

This model allowed me to look at course difficulty through a few different lenses. First, note that this metric is not attempting to predict which courses have the highest scoring average or anything of that sort. Instead, this coefficient is answering the following question, "all else being equal, how much harder is this shot on one course versus another?" Because of the nature of the model I have fit, "all else being equal" means fixing `Player`,

`Location.Scorer`, `Distance`, and `Time`. With that framework, I looked at the hardest courses on the PGA Tour in 2018, as shown in Table 5.2. Leading

Course	Strokes Added
Pebble Beach GL	.039
Plantation Course at Kapalua	.031
Silverado Resort and Spa North	.028
Quail Hollow Club	.023
Torrey Pines GC (South)	.023
TPC San Antonio - AT&T Oaks	.022
TPC Summerlin	.022
Riviera CC	.018
Muirfield Village GC	.014
TPC Sawgrass	.012

Table 5.2: Courses ranked by expected strokes added per shot. This is a ranking of the courses with the most difficult terrain.

the group is the notorious Pebble Beach. As far as conventional wisdom is concerned, any list of difficult courses from 2018 would include both Pebble Beach and Torrey Pines. There are not many publicly available rigorous course rankings of this style for which I can compare my results.

From here I extended this analysis to the most difficult holes as shown in Table 5.3 Jumping out at the top is the iconic 18th hole of Pebble Beach. If there was any confusion as to why this is a difficult hole, Figure 5.3 shows overhead satellite views of both the 10th at the Riviera and the 18th at Pebble Beach. Both of these holes have large obstacles making the penalty for an inaccurate shot very high. On the Riviera, the hole is essentially fully surrounded by sand traps, while Pebble Beach forces the player to shape the ball on a thin fairway around the ocean and a large sand trap.

While it is possible to use this model to investigate more granular effects such as round specific course difficulties, it is hard to develop a frame of reference for what these mean[1].

[1]The main things might be weather and hole location.

Course	Hole	Strokes Added
Riviera CC	10	0.16
Pebble Beach GL	18	0.14
Pebble Beach GL	14	0.13
TPC San Antonio - AT&T Oaks	1	0.12
Torrey Pines GC (South)	13	0.11
Silverado Resort and Spa North	8	0.10
TPC River Highlands	15	0.10
Pebble Beach GL	11	0.09
TPC Summerlin	11	0.08
Silverado Resort and Spa North	3	0.08

Table 5.3: Holes ranked by expected strokes added per shot. This is a ranking of the courses with the most difficult terrain.

Figure 5.3: An overhead view of the two most difficult holes on the PGA Tour according to the `GAM.Time.Loc` model.

5.3 Player Rankings

Making our way through the model features, the next application is to look at player rankings, both broadly and within specific locations. Also, since this model is fit using the `Player.Agg` feature, the data will contain a baseline estimate for a replacement level player. This means that in addition to scoring players relative to average performance, I was also able to score them relative to replacement level. The player rankings for different surfaces are shown in Table 5.4. There is a lot going on in this chart, starting with the fact

Green		Fairway		Primary Rough		Bunker	
Player	S.A.	Player	S.A.	Player	S.A.	Player	S.A.
J. Day	-.039	T. Woods	-.097	W. Simpson	-.075	J. Day	-.077
W. Simpson	-.035	W. Simpson	-.082	J. Thomas	-.073	W. Simpson	-.077
S. Burns	-.031	W. Bryan	-.074	P. Reed	-.064	K. Na	-.074
G. Chalmers	-.031	K. Na	-.072	T. Hatton	-.056	R. Fowler	-.071
A. Noren	-.029	J. Day	-.072	J. Lovemark	-.055	J. Thomas	-.065
K. Kisner	-.029	J. Rose	-.068	P. Mickelson	-.054	J. Rose	-.063
P. Malnati	-.027	J. Thomas	-.068	J. Rose	-.051	O. Schniederjans	-.061
D. Summerhays	-.026	D. Johnson	-.060	J. Walker	-.049	J. Spieth	-.058
B. Hossler	-.026	P. Mickelson	-.058	J. Dahmen	-.048	P. Reed	-.056
J. Wagner	-.025	T. Fleetwood	-.057	M. Kuchar	-.048	A. Baddeley	-.056
Replacement	.021	Replacement	.05	Replacement	.054	Replacement	.056

Table 5.4: Player rankings by location type according to expected strokes under average, according to `GAM.Time.Dist`. This ranking aggregates players across shot types to a surprising degree.

that that players are able to achieve much higher averages over baseline on longer shots relative to shorter shots. As we can see, Tiger is worth nearly .15 strokes over the average player from the fairway while Day is only worth .06 strokes on the putting green. A complicating factor of this is that the green has far more shots, allowing for the elimination of some of the noise. Still, there is enough data here to suggest that this result will hold up reasonably well with more observations. This is especially supported by the replacement player indicator that has a much higher N but still shows a lot more deviation from the mean on fairway relative to green.

The other thing I will touch on is the apparent correlation across location types among players, stemming from the hierarchical structure of the model. The result of this is that a player like Webb Simpson, who is probably not the best player on the tour, had a series of good rounds in 2018 and now looks like the best player across all dimensions. Because of this, I refit the model relaxing this single hierarchical assumption. The results can be seen in Table 5.5. Finally, I used the visualization technique outlined in methods

Green		Fairway		Primary Rough		Bunker	
Player	S.A.	Player	S.A.	Player	S.A.	Player	S.A.
S. Burns	-0.036	T. Woods	-0.10	J. Thomas	-0.064	K. Na	-0.084
G. Chalmers	-0.034	W. Bryan	-0.078	W. Simpson	-0.063	R. Fowler	-0.075
Jason Day	-0.030	W. Simpson	-0.076	J. Lovemark	-0.061	J. Day	-0.069
D. Summerhays	-0.029	K. Na	-0.075	P. Reed	-0.061	S. Power	-0.066
W. Simpson	-0.028	J. Day	-0.066	J. Dahmen	-0.060	O. Schniederjans	-0.066
K. Kisner	-0.027	J. Rose	-0.064	T. Hatton	-0.058	A. Baddeley	-0.062
B. Hossler	-0.027	T. Fleetwood	-0.061	B. Haas	-0.052	W. Kim	-0.061
P. Malnati	-0.025	J. Thomas	-0.061	P. Mickelson	-0.050	J. Spieth	-0.060
A. Noren	-0.024	D. Johnson	-0.060	D. Bozzelli	-0.050	D. Lee	-0.060
P. Rodgers	-0.022	P. Mickelson	-0.056	M. Kuchar	-0.048	W. Simpson	-0.057

Table 5.5: Player rankings by location type according to expected strokes under average, according to a modified version of `GAM.Time.Dist`. This ranking has much more variation and shows clear outliers.

to chart Tiger Wood's performance over a replacement level player on TPC Sawgrass Hole 5 in Figure 5.4.

5.4 Time of Day Analysis

The last unexplored attribute in `GAM.Time.Loc` model is `Time`. As I mentioned above, this has been fit to vary with `Location.Scorer` and `Distance`. The plot below shows the global change in strokes added in general, and also the specific changes for fairway, intermediate rough, and primary rough. This is plotted to be independent of distance, so it is a summation of the time based splines that do not include distance, as shown in Figure 5.5. This is restricted to the window 8:00 AM to 6:00 PM because outside of these times the shot frequency falls off and the estimates have much higher error. This

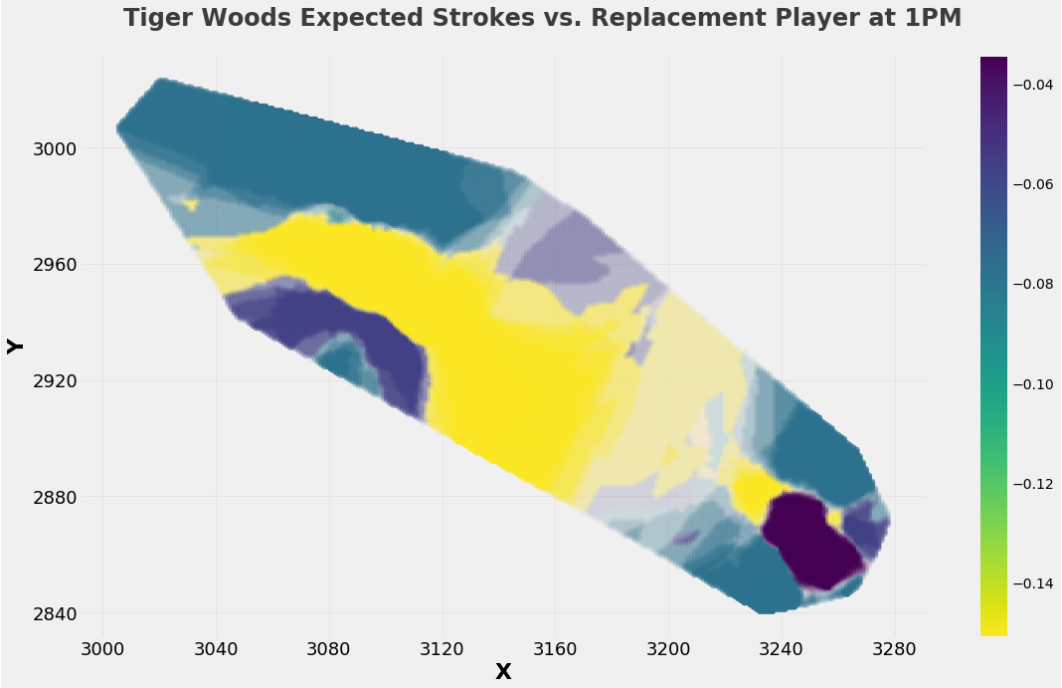

Figure 5.4: A visualization of the difference in expected strokes between Tiger Woods and a replacement level golfer on TPC Sawgrass, Hole 5, Round 1 at 1 PM.

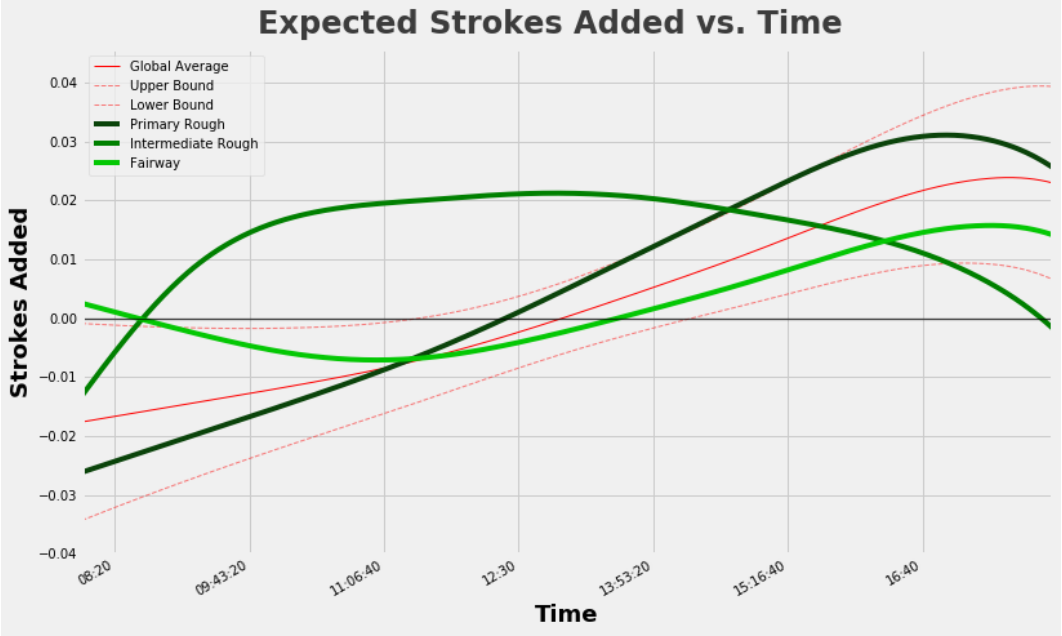

Figure 5.5: Expected strokes added vs. time for primary and intermediate rough, as well as fairway. The red lines show the global average across all shots as well as the 95% standard error on this measurement.

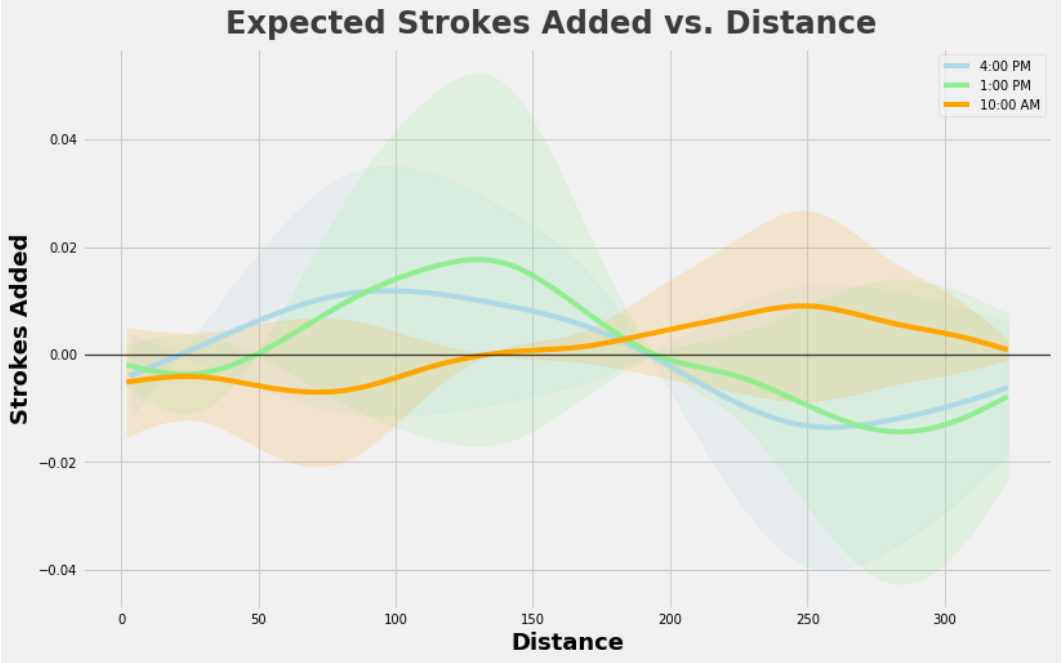

Figure 5.6: Average shot difficulty by `Distance` for three times in the day, 10 AM, 1 PM, and 4 PM, along with 95% standard error measurements.

graph supports the hypothesis that shots are easier in the morning and get progressively more difficult during the day. In terms of locations, there also seems to be evidence that most of the benefit of the grass is accrued by the primary rough relative to the other grasses due to the fact that it is much harder to get spin on the ball from the primary rough, leading to increasing benefit from a damp green.

The next aspect of this worth investigating is the associated impact of `Distance` as it relates to `Time`. This is slightly more difficult to conceptualize because it is a 2 dimensional surface and not a 1 dimensional spline. Instead of generating a heatmap or something similar, I instead opted for a more interpretable version that admittedly loses some of the information. I picked three times of day, 10:00 AM, 1:00 PM, and 4:00 PM, and looked at the change in difficulty relative to distance at all three well spaced out times, as seen in Figure 5.6. This chart reinforces the `Time` effects that have been conventional wisdom. Forced with both extremes, a golfer would rather take

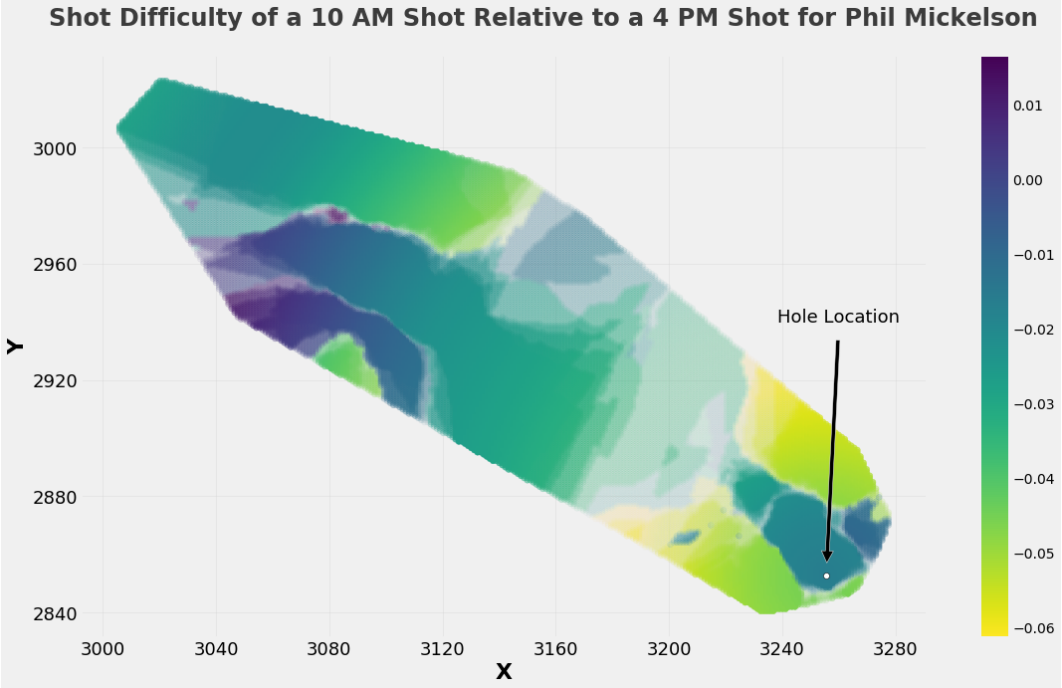

Figure 5.7: Figured generate by taking Phil Mickelson's shot difficulties by location at 10 AM and subtracting the difficulty at 4 PM.

his closer shots with a high probability of hitting the green in the morning, and his further shots in the evening than vice versa. This effect seems to be centered around 200 yards, and anything further out has a comparative advantage in the evening. This chart only shows the interaction though, not the absolute effects. It is worth noting that this is a statistically significant effect, but the standard error measurements on this chart are very large.

Finally, again consider Phil Mickelson's first round on TPC Sawgrass Hole 5. I graphed Phil's shot difficulty at 10 AM relative to 4 PM in Figure 5.7. From this we can see the distance and location effects in action. Phil would prefer his short recovery chip shot in the morning, and a longer shot that needs more bounce in the evening[2].

[2]Unfortunately for Mickelson, his errant shot took place at 3 p.m.

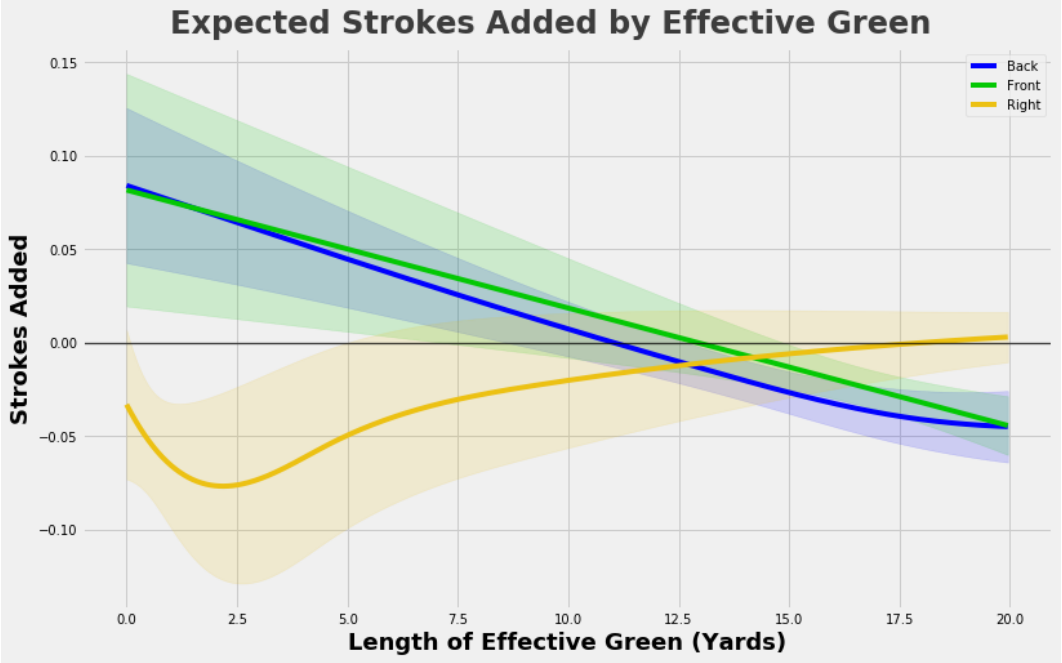

Figure 5.8: `Eff.Green.Front`, `Eff.Green.Back`, and `Eff.Green.Right` plotted against expected strokes added with 95% standard error included.

5.5 Breaking Down Effective Green

The last additive model structure I fit was the models dependent on `Eff.Green`. The obvious question here is the relationship between different types of `Eff.Green` and shot difficulty, which I plotted in Figure 5.8. While front and back seem to vary exactly as one would expect, the green on the right has the inverse relationship. A further investigation should be done to investigate if there is some unforseen spatial dependence that this model is indirectly picking up.

To incorporate more of the `Eff.Green` terms, this model can be visualized on a real course to see how these effects manifest in practice, as shown in Figure 5.9. This structure clearly highlights shots as easier when there is more visible green between the shot location and the hole. There are also some edge effects around the green as the calculation becomes less stable.

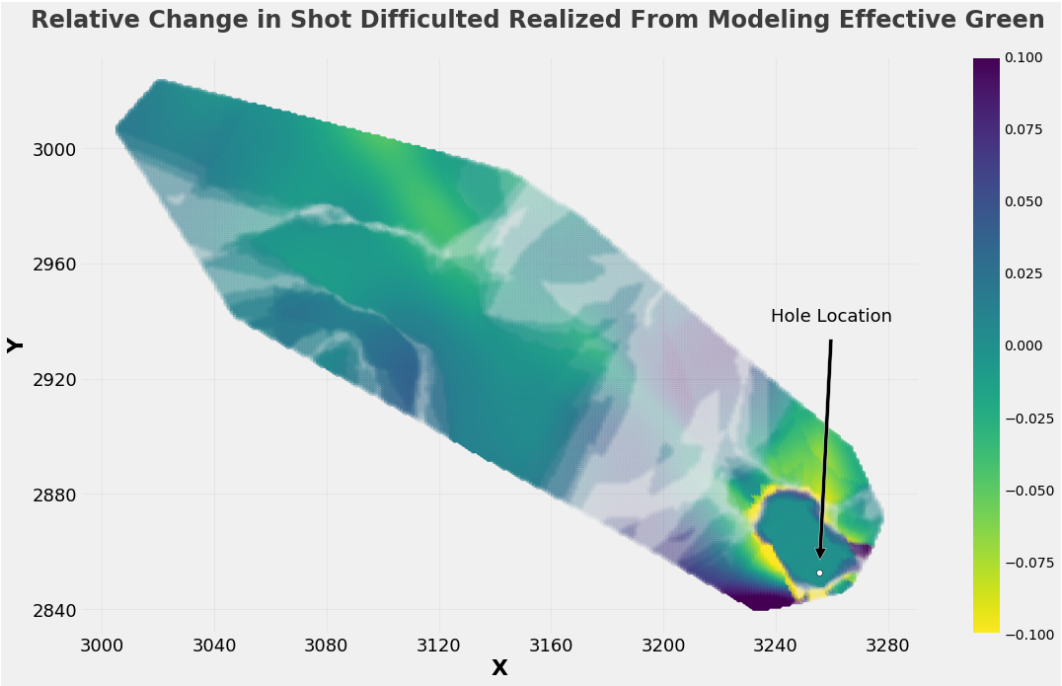

Figure 5.9: Difference between `GAM.EG.Loc` and `GAM.Baseline` on TPC Sawgrass Hole 5 Round 1. This is formed subtracting out the baseline from the `GAM.EG.Loc` predictions.

6

Conclusion

Golf has suffered throughout the years from a problem with data uniqueness. Every course moves around the holes on the green, the PGA Tour cycles through different courses all year, and different players select to play in different tournaments. On the other hand, every tournament possible a hundred shots are taken between 100-yards and 150-yards. In golf analysis there is either a tendency to go too big and say that everything is luck, or go too small and say that nothing matters besides distance and surface. In reality, golf analytics has only scratched the surface of the possible feature set for these models, and there have not been many proposals for how to test the impacts of different attributes in meaningful and robust ways.

The contribution of this paper is on a few fronts. First, I propose a rigorous way to simultaneously fit a nonlinear stroke difficulty function over distance while also controlling for round and player effects. Second, I use this framework to investigate some less talked about golf metrics such as time of day or effective green. And finally, I was able to engineer enough data to build visualizations of course difficulty over essentially every playable area. The impact of this third results cannot be overstated. This is a tool that real people who follow golf would enjoy using. Additionally, no other papers have proposed true out-of-sample prediction in this manner.

This paper puts a large emphasis on interpretability. For a game that obviously has so many different factors at play, there is almost no consensus on how these variables impact play difficulty. It is my hope that through effective

modeling and precise visualization, I can persuade the golf community that these small impacts on stroke quality cumulatively determine most of what we understand about the game of golf.

Bibliography

[Bro08] Mark Broadie. „Assessing golfer performance using golfmetrics“. In: *Science and golf V: Proceedings of the 2008 world scientific congress of golf*. World Scientific Congress of Golf Trust St. Andrews. 2008, pp. 253–262 (cit. on p. 4).

[Bro12] Mark Broadie. „Assessing golfer performance on the PGA TOUR“. In: *Interfaces* 42.2 (2012), pp. 146–165 (cit. on pp. 4, 5, 13, 31, 36, 40, 53).

[Bro14] Mark Broadie. *Every shot counts: Using the revolutionary strokes gained approach to improve your golf performance and strategy*. Avery, 2014 (cit. on p. 4).

[Fea+11] Douglas Fearing, Jason Acimovic, and Stephen C Graves. „How to catch a Tiger: Understanding putting performance on the PGA Tour“. In: *Journal of Quantitative Analysis in Sports* 7.1 (2011) (cit. on p. 3).

[GH06] Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006 (cit. on pp. 45, 46).

[Lev17] ADAM Levin. „Ranking the Skills of Golfers on the PGA TOUR using Gradient Boosting Machines and Network Analysis“. In: MIT Sloan Sports Analytics Conference. 2017 (cit. on pp. 6, 15, 22, 28, 40, 60).

[Stö+11] Michael Stöckl, Peter F Lamb, and Martin Lames. „The ISOPAR method: a new approach to performance analysis in golf“. In: *Journal of Quantitative Analysis in Sports* 7.1 (2011) (cit. on p. 5).

[YS13] Kasra Yousefi and Tim B Swartz. „Advanced putting metrics in golf“. In: *Journal of Quantitative Analysis in Sports* 9.3 (2013), pp. 239–248 (cit. on p. 5).

(cont.)

[PT] INC. PGA TOUR. *ShotLink*. URL: http://www.shotlink.com/about/background (visited on Mar. 31, 2019) (cit. on pp. 3, 4, 7, 10).

List of Figures

List of Tables