Prediction Intervals: The Effects and Identification of Sparse Regions for Nonparametric Regression Methods

Jackson Faires
fairesjc@jacks.sfasu.edu

PREDICTION INTERVALS: THE EFFECTS AND IDENTIFICATION OF SPARSE REGIONS FOR NONPARAMETRIC REGRESSION METHODS

by

Jackson Faires, B.S.

Presented to the Faculty of the Graduate School of

Stephen F. Austin State University

In Partial Fulfillment

of the Requirements

For the Degree of

Master of Science

STEPHEN F. AUSTIN STATE UNIVERSITY

August 2021

PREDICTION INTERVALS: THE EFFECTS AND IDENTIFICATION OF SPARSE REGIONS FOR NONPARAMETERIC REGRESSION METHODS

by

Jackson Faires, B.S.

ABSTRACT

In this work, we provide an overview of different nonparametric methods for prediction interval estimation and investigate how well they perform when making predictions in sparse regions of the predictor space. This sparsity is an extension to the more common concept of extrapolation in linear regression settings. Using simulation studies, we show that coverage probabilities using prediction intervals from quantile k-nearest neighbors and quantile random forest can be biased to low or too high from the nominal level under various situations of sparsity. We also introduce a test that can be used to see if a new data point lies in an area of sparse data so that users may be able to identify problematic situations. Additional simulations results are shown to assess the tests overall performance.

ACKNOWLEDGEMENTS

I would like to thank my family and my friends, who are basically family anyway, for being there to support me. I would like to thank Dr.Turner for revitalising my love for statistics. I would like to thank my committee members for their time and their thoughtful help.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Regression is a practice that models a continuous variable (response) to describe the relationship between the response and another set of numeric and categorical variables (predictors). One goal of a regression model is to provide predictions of future observations based on their known predictor values. To motivate this task, we provide a few basic graphs of the "Advertising" data set referenced in [5]. The goal with this particular data set is to predict the amount of sales revenue using advertising investments in TV, Radio, and Newspaper. Figure 1.1 provides scatter plots of sales versus each predictor. One can see that among TV and Radio, moderate trends exist with sales while there appears to be no association between sales and Newspaper advertising.

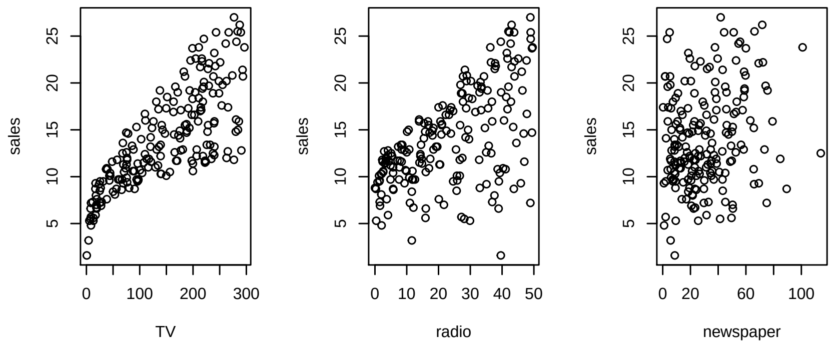


Figure 1.1: Advertising

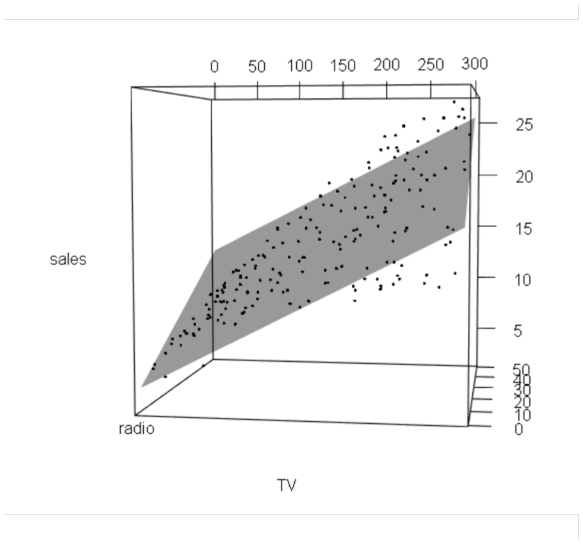
Upon closer examination of the plots one might suspect that a linear trend could be used to express the average of the response as a function of TV, Radio, or both. It is this line of thinking that motivates a general regression model framework. If you have p predictors $X = (X_1, X_2, ..., X_p)$ to help you predict a continuous response Y, then the general form of the relationship can be written as

$$Y = f(X) + \epsilon \tag{1.1}$$

where ϵ is a random error term with mean 0 for each observation. The typical challenge is to estimate f from data, which we will denote $\hat{f}$. The estimation strategy can typically be classified as a parametric or a nonparametric approach. Parametric approaches typically assume a known class of functions which have additional parameters that control the level of complexity the class of functions can offer. They also typically add additional assumptions to the error term including both its distribution and the relationship between the response and the predictors. The most widely used parametric approach is multiple linear regression (MLR). Under this approach, the general form of MLR is expressed as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{1.2}$$

where $\epsilon \sim N(0, \sigma^2)$ and is independent for each observation. If all of our predictors are numeric then our response surface is a hyper-plane. If one or more of our predictors is categorical we can include what are called "dummy" variables. In order to use dummy variables, we consider each category as a separate predictor X where $X = 1$ when that particular category has shown up and $X = 0$ when it does not. Because of this binary coding strategy, creating dummy variables are often referenced as "one-hot encoding". One misleading term in the name multiple linear regression is the word linear because the model does not require the trends we are trying to model to be linear. We can model more complex relationships like polynomial or even interactive trends. For example, specifying $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$ is a perfectly valid linear regression model. The term linear is in reference to the fact that the regression model can be expressed as a linear combination of the regression coefficient parameters. In the case of the Advertising dataset, we can create many different MLR models. To highlight how MLR can be more than just a line Figure 1.3 provides a comparison of a simple MLR model (including TV, Radio versus a more complex) to a more complex model that includes an interaction term between TV and Radio.



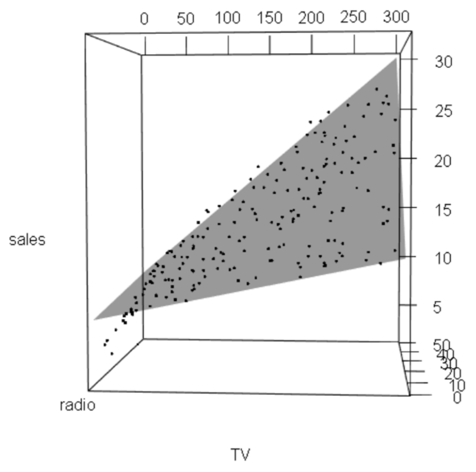


Figure 1.2: Advertising MLR simple additive model vs interaction term added

One disadvantage of multiple linear regression is modeling complicated relationships can be difficult to express by the modeler fitting the data. Nonparametric tools such as regression trees, k-nearest neighbors (KNN), and random forest models are but just a few that take a data driven, “learning approach”, to estimating f with no specific assumptions on the true form of f. Figure 1.3 provides a side by side comparison of a response surface of the advertising data using MLR with Radio, TV, and their interaction term and a $k = 5$ KNN model with Radio and TV as the predictors. The nonparametric fit lacks the “smoothness” exhibited by the regression model as it utilizes predictions using only the observed response values whose predictors correspond to the 5 closest (neighbors) to the new candidate predictor values to which a prediction is needed.

To elaborate on this idea, nonparametric approaches tend to model trends based on location of predictor combinations that are, by some definition, close to the new

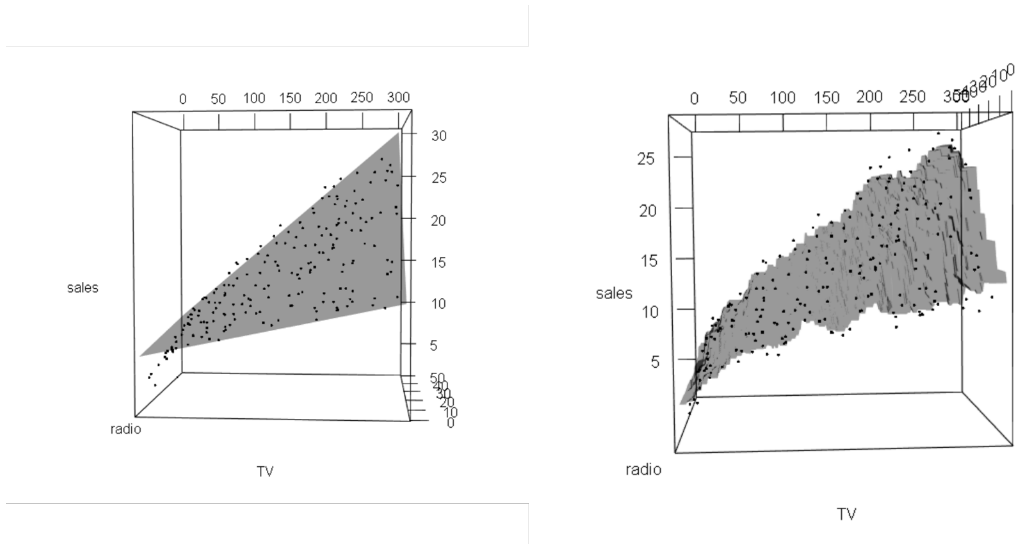


Figure 1.3: Advertising MLR Interaction Model vs KNN model $k = 5$

observation. Treating the observed values of the predictors as a coordinate system is often referred to as the predictor space. For example, when working with two continuous predictors, the predictor space for the observed data points are located in $\mathbb{R}^2$. Figure 1.4 provides the predictor space for the advertising data set when considering Radio and TV alone. The response variable, sales, is color coded to visualize potential trends with the predictors. The warmer shaded points reflect larger values of the response and tend to occur when both TV and Radio advertisement have larger values. This behavior is reflected in both the MLR model and the KNN model in Figure 1.3.

Unlike MLR, which relies on the modeler to recognize trends, nonparametric methods utilize the idea of closeness in the predictor space to make predictions. Relying on closeness has both advantages and disadvantages. The major advantage is that the modeler only has to manipulate the tuning parameter for each method to create a more or less complicated model. The major disadvantage is you need a significant amount of data to make an accurate model that doesn't suffer from modeling random

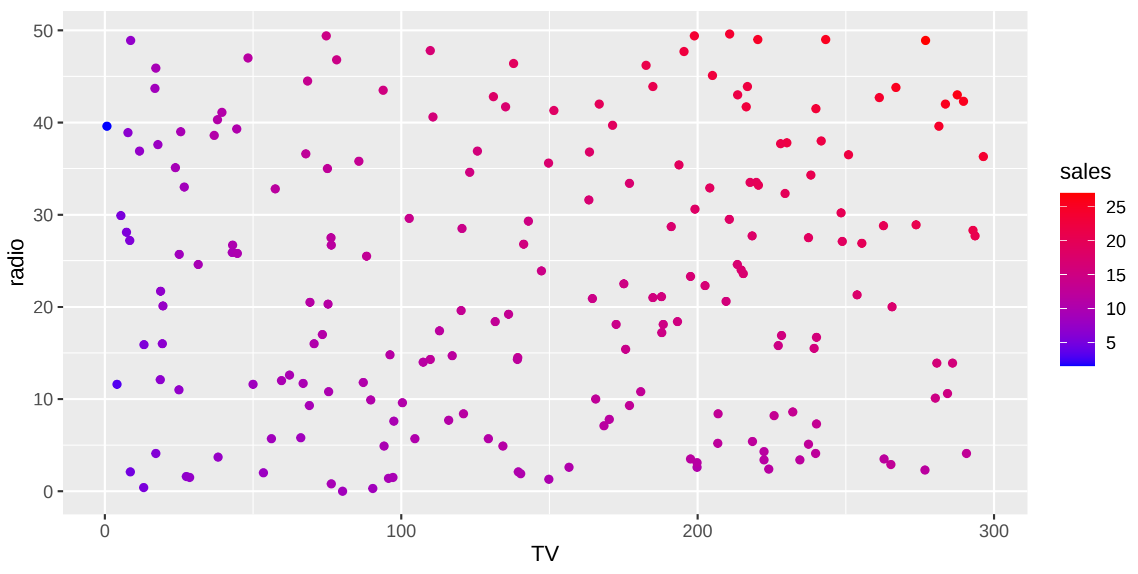


Figure 1.4: Advertising predictor space

variation. A more detailed description of nonparametric methods will be provided later in the manuscript.

The focus of this research is to investigate various aspects of nonparametric regression models which can be broken into two main goals. The first goal is the assessment of nonparametric prediction intervals when the intervals being produced have predictor values that fall in regions of the predictor space where little to no data are available, more specifically, extrapolating when you are within the confines of the predictor space. When the data is observed, some values in the predictor space may be more common than others. In some cases, this phenomena can create regions in our predictor space that are sparsely populated or regions that are void of all data. In Figure 1.5, we removed data from the middle of the advertising data set to illustrate the same predictor space with a hole in the middle. Traditionally we think of extrapolation as trying to predict outside the range of your training data, however in cases like the one depicted in Figure 1.5 trying to predict in the middle of this data can be just as problematic as trying to predict outside of the predictor space.

This particular research questions allows for a few major discussions to take place.

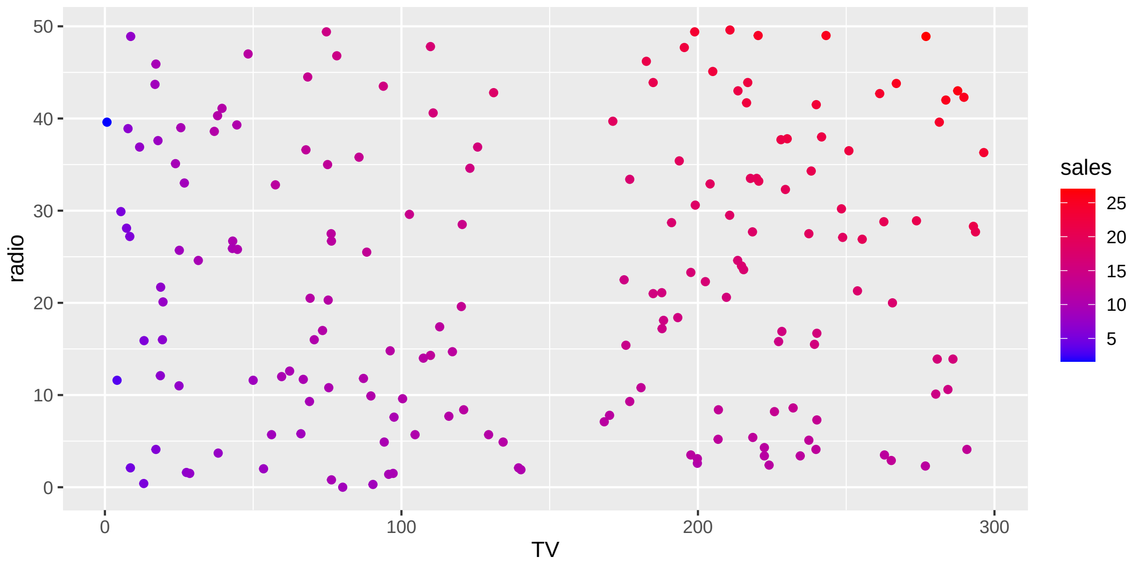


Figure 1.5: Advertising predictor space with data removed

The first is to introduce, through literature review, how some nonparametric prediction intervals are constructed. The main focus for nonparametric regression models have been based on point predictions. However, as predictive modeling has grown in popularity, providing prediction intervals is a more common request and various methods have been introduced to meet the need. Secondly, we wanted to assess the performance of these prediction intervals under various conditions including making predictions in sparse regions. To our knowledge, we are not aware of any research that has done so to date. Finally, we have made an observations that most authors argue their respective prediction interval methods are working as intended by producing prediction intervals for the data that the model was trained on and then computing the coverage rate of their prediction intervals. It is our hypothesis that this is a flawed approach when examining if prediction intervals have control of their true specified coverage levels as it ignores issues like the sparsity. We performed simulation studies to show prediction interval coverage rates from the training data set do not necessarily reflect good prediction coverage in some cases and can be overly optimistic in others depending on the location of the new point you are trying to predict relative to the training data used to fit the model. We examined three different prediction interval methods used for multiple linear regression, k-nearest neighbors, and random forests.

The second major goal of this project was generated after review of our simulation study results. It became apparent to us that there is a need to be able to identify if a new observation presented for prediction is indeed consistent with the training data or not. We investigated two possible procedures that could potentially help detect whether a new observation is close to data used to build the model or not. If verified, we can be more comfortable about the prediction we will make for the new observation. Our first investigation was to see if prediction interval widths could be used to detect areas of sparse data. Our thinking was that since we are using nonparametric methods to build the intervals, maybe the intervals would become wider in areas of little or no data. For our second investigation, we developed a straight forward diagnostic test to determine if a new observation is in a region of sparse data. We also performed simulation studies to assess the performance of our diagnostic tool.

A brief summary of the following chapters is provided. Chapter 2 consists of a more in depth discussion of the nonparametric prediction models along with prediction interval strategies for linear regression, k-nearest neighbors, and random forests. We also introduce the methodological framework and intuition of our proposed statistical test for detecting sparse points. Chapter 3 provides a series of simulation results illustrating the issues that these prediction intervals can have under certain situations along with an investigation of our diagnostic test's ability to detect these problems on future values. Chapter 4 discusses our findings, considerations, and possible future work.

2 Methods

To begin, we will briefly introduce a small amount of theory to give insight to three regression models we considered following the discussion of James, Hastie, Tibsharani, and Friedman [5]. We will briefly discuss how a point prediction is made for each method along with strategies for making prediction intervals for each method.

Building regression models comes down to a proper selection of $f(X)$, the relationship between the continuous response, Y, and an input vector X. Lets assume that that the predictor space for X is in general $\mathbb{R}^p$. To determine f, a loss function is introduced for penalizing the prediction errors made for a given candidate model. The most common of these is the squared error loss function $L(Y, f(X)) = (Y - f(X))^2$. The typical criterion for choosing f is to minimize the expected loss,

$$E[L(Y, f(X))] = \int (y - f(x))^2 P(x, y) dx dy \tag{2.1}$$

where $P(x, y)$ is the joint distribution of the explanatory variables and the response. Using conditional probability, minimizing the expected square error loss can be achieved, by conditioning on X, and minimizing it pointwise. The solution can be shown to reduce to

$$f(X) = E(Y|X = x), \tag{2.2}$$

the conditional expectation of Y given a single observed point x in $\mathbb{R}^p$. Estimating f using data, denoted as $\hat{f}$, requires that numerous observations have been observed for each fixed value x. This is typically not the case in practice. As we will see in the upcoming sections, either additional assumptions are made on the conditional expectation or the conditional statement is somewhat relaxed. When making a prediction for a new value of Y at a new given point x_0, a simple point prediction is to evaluate the estimated function at $\hat{f}(x_0)$.

The conditional mean provides just one piece of summary information on the conditional distribution of Y. This led to the development of quantile regression [1]. The conditional distribution function $F(y|X = x)$ is given by the probability statement

$$F(y|X = x) = P(Y \leq y|X = x). \tag{2.3}$$

The α-quantile of a continuous conditional distribution $Q_\alpha(x)$ is then defined as the value satisfying $P(Y \leq Q_\alpha(x)|X = x) = \alpha$. Quantile regression tries to estimate conditional quantiles and can be viewed in the same light as the optimization problem for general regression just using a different loss function. If we let $g(x)$ represent the functional form for the α-quantile of the conditional distribution $Y|X = x$, define the loss function L_α as

$$L_\alpha(y, g(x)) = \begin{cases} \alpha|y - g(x)| & y > g(x) \\ (1-\alpha)|y - g(x)| & y \leq g(x) \end{cases} \tag{2.4}$$

Minimizing the expected loss $E(L_\alpha)$ yields the solution $g(x) = Q_\alpha(x)$. So while the conditional mean minimizes the squared error loss, conditional quantiles minimize a loss function defined on absolute deviations that are weighted by α, the quantile of interest. Similar to standard regression tools, we must estimate conditional quantiles, $Q_\alpha(x)$ from observed data. Additional assumptions are typically made on the functional form of $g(x)$ or the conditional statement is relaxed slightly.

Once an estimate of the quantile function $\hat{Q}_\alpha(x)$ can be obtained, quantile regression can be easily used to produce prediction intervals for a new value of Y at a new given point x_0. Suppose one computes two quantile regression fits for the 2.5 and 97.5% quantiles. A 95% prediction interval for the value of Y given a new observed

x_0 is given by

$$I(x) = (\hat{Q}_{.025}(x_0), \hat{Q}_{.975}(x_0)). \tag{2.5}$$

2.1 k-nearest neighbors

k-nearest neighbors (KNN) is a nonparametric regression tool that attempts to estimate the conditional mean for a new observation, x_0, by identifying the k points of observed data that are closest to the new observation for which a prediction is needed. The response values for these nearest observations are then averaged together. The k-nearest neighbor predictions are more formally computed by using the following equation:

$$\hat{Y}(x_0) = \frac{1}{k} \sum_{x_i \in N_k(x_0)} y_i \tag{2.6}$$

where $N_k(x_0)$ is the neighborhood of x_0 defined by the k closest points x_i in the training data. Since most observed data will likely have just one or no observations at a candidate x_0, the observed response values for the closest neighbors serve as an approximate for the conditional distribution $Y|x = x_0$. Thus, averaging across these observed values is an estimate of the conditional mean at x_0.

An important decision a statistical modeler must make when using KNN is to decide what value of k minimizes the squared error loss. The choice comes down to the ever present bias-variance trade-off. A smaller choice of k leads to a more flexible but also more variable model, whereas a larger choice of k will lead to a smoother, less variable model since the predictions are an average of more points. The smoothness of the higher k models might lend themselves to a higher bias since the averaging can lead to masking some of the structure of $f(X)$. In Figure 2.1 you can see the advertising data graphed twice along with two grey surfaces laid over the data. The grey surfaces represent the predictions made by building two different KNN models using the Advertising data. The model of the left was built using $k = 5$ where as the model on the right was built with $k = 20$.

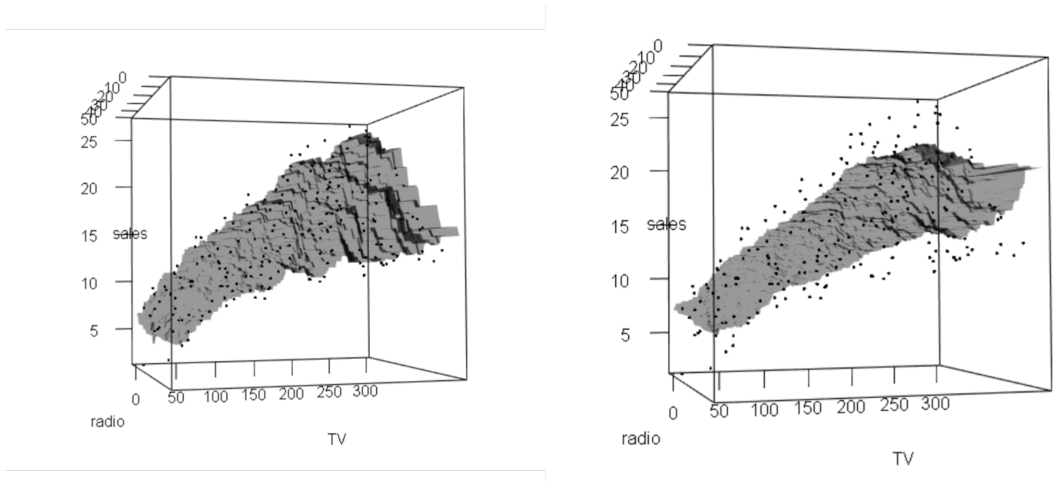


Figure 2.1: KNN 5 model vs KNN 20 model

2.2 Prediction Intervals for KNN

k-nearest neighbor quantile regression (QKNN) has been discussed in numerous articles dealing with various properties of the method. Both [2] and [4] provide good introductory definitions along with exploring various asymptotic properties of the estimator. The procedure is quite straightforward based on our previous discussions. Since the response values at k closest neighbors to x_0 serve as an approximate for the conditional distribution $Y|x = x_0$, minimizing the loss function in equation (2.4) for a given α corresponds to computing the α^{th} quantile from the k observed response values.

There are numerous estimates of quantiles for given set of data. Generally speaking, they tend to be biased estimators of the true quantile but are asymptotically unbiased. The quantile function in the statistical package R provides 9 numerical estimates from estimating quantiles. The differences in methodology tends to boil down to the choice of how two ordered statistics are averaged together to estimate the quantile of interest. For this manuscript, we utilized the "type-7" procedure which is a commonly used default setting in software such as SPSS.

It was only just recently, that [7] proposed a method of choosing an appropriate value of k for QKNN. Similar to the standard KNN procedure for a point prediction, the choice of k effects the complexity at which the quantiles are being predicted. The choice of k can also be different depending on the quantile level, α, one wishes to estimate. While a procedure to handle the process automatically is useful in practice, for our investigation we will explore the choice of k through our simulations results provided later.

2.3 Random Forests

In order to understand random forests as a regression tool, we must first discuss regression trees. A regression tree, like KNN, is a nonparametric prediction method that approximates the conditional mean by using available data close in proximity to the point one wishes to predict. For continuous predictors, regression trees split the predictor space into high dimensional rectangles rather than using neighbors. When predicting a new point, x_0, one simply averages all the response variables located in the hyperrectangle that x_0 belongs to. This effectively approximates the conditional mean $E(Y|X = x_0)$.

The partitioning of the predictor space through these high dimensional rectangles can be expressed using a tree. Using the advertising data set as an example, Figure 2.2 provides a simple regression tree fit that partitions the two dimensional predictor space into three regions. Its corresponding tree is also provided. At each branch of the tree, the value you are trying to predict is compared with the number at that branch of the tree; if the value you are trying to predict is less than you take the left branch, greater than you take the right branch. You continue in this fashion until you reach one of the bottom branches of the tree. At the bottom of each branch is the predicted value for that rectangle. The complexity of a regression tree depends on the

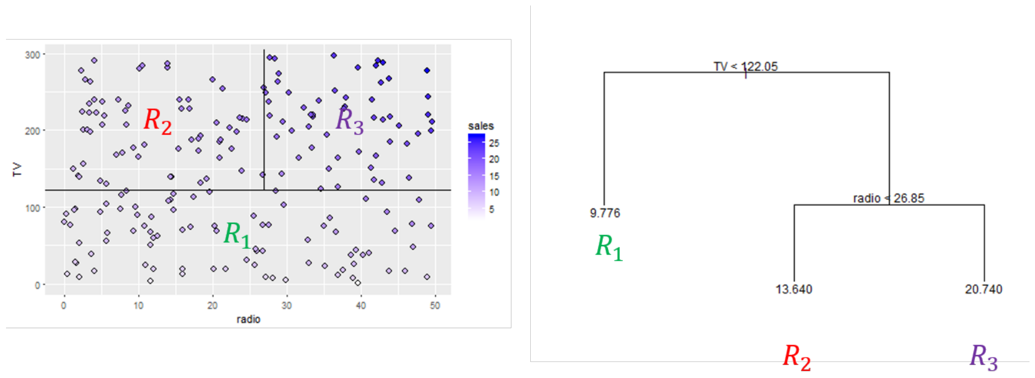


Figure 2.2: Simple tree model

number of partitions defined by the tree. The previous example with only two splits can only provide three unique prediction values for any new candidate observation, x_0. This provides a very coarse approximation to $f(X)$. Figure 2.3 provides response surfaces to the advertising data set using the previous regression tree (left) versus a regression tree with more partitions (right). The splits in our predictor space are chosen using a technique known as recursive binary splitting [5] which seeks to find hyperrectangles $R_1, R_2, ..., R_j$ that minimize the RSS given by

$$\sum_{j=1}^{J} \sum_{i \epsilon R_j} (y_i - \hat{y}_{R_j})^2 \tag{2.7}$$

where J is the total number of rectangles, R_j is the jth rectangle, y_i is the i^{th} response, and $\hat{y}_{R_j}$ is the average of the response values associated with the j^{th} rectangle. Since considering every possible combination of splits would be computationally intensive, recursive binary splitting is a top-down greedy approach to select each split of the regression tree in a computationally efficient way.

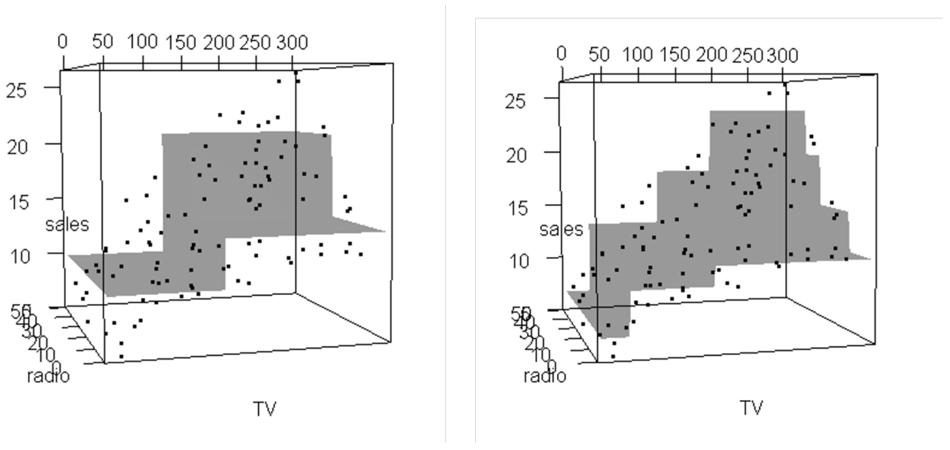


Figure 2.3: Simple tree model vs Complex Tree model

In summary, we start with assuming that there is only one region, the entire predictor space. Recursive binary splitting determines the best possible binary partition of the predictor space using each individual predictor one at a time. "Best" here is defined by minimizing the RSS. The predictor with its corresponding partition that has the smallest RSS is then chosen to represent the first split of the tree diagram. Moving forward, the same splitting routine is conducted but applied to each of the two regions of data rather than the entire data set, and the best split is found. This would create three regions. The process continues until some stopping criterion is met, such as, no region should contain fewer than five observations or reduction of RSS threshold is not met.

Recursive binary splitting is "top down", since the tree grows downward visually as additional regions are created if reduction in RSS is found through additional binary splitting. The algorithm is "greedy" since we do not consider how each split prior affects the tree later down the line in terms of overall RSS reduction. It is quite possible that there exists a candidate split that does not produce the smallest possible reduction in RSS at a given step, but leads to much larger global reduction in RSS via the partitions made deeper down the tree. Recursive binary splitting cannot achieve these potentially lower minimums since they only favor the best partition split at each step.

As Figure 2.3 shows, trees with deep splitting produce more complex estimates of $f(X)$. Like with KNN, choosing the most appropriate tree fit comes back to the bias-variance trade off. The logistics of choosing the best tree can be found in [5], however for random forests this is a non-issue as it overcomes fitting overly complex issues in a very specific way.

While regression trees are very intuitive and provides a graphic that is highly interpretive to understand the relationship between the response variable and predictors, regression trees tend to suffer in prediction accuracy as most relationships observed in practice are relatively smooth. The random forest regression model attempts to overcome this issue by using a modified version of bagging.

Bootstrap aggregation, or bagging, is an ensembling method that takes advantage of averaging predictions from numerous models. Before providing details of the bagging procedure, it is important to briefly introduce the bootstrap procedure. The bootstrap, originally developed by Bradley Efron [9], is a resampling procedure that is typically used to approximate the sampling distribution of statistics. The procedure is very useful when working with statistics whose sampling distributions are hard to obtain theoretically, like the median.

If one had access to an extremely large number of data sets, one could obtain the sampling distribution of a statistic numerically by calculating the statistic of interest on each one of the data sets and recording them for study. Visualizing the statistics in a histogram allows for the investigation of the sampling distributions shape while computing means and variance offer insight to its measure of location and variability. Assessing variability of the statistic is key as it directly relays the level of accuracy or reproduciblity the statistic has.

When building a predictive model, the estimate $\hat{f}(X)$ is also a statistic derived from data. Therefore, if we examine this function estimate for a given X pointwise, it also has a sampling distribution. The variability of this sampling distribution gives us a sense of how reproducible our estimate of $f(X)$ really is at a given point. Models with high levels of complexity that tend to model the the overall trend well tend to suffer from having too high of variance. Having multiple samples available to obtain numerous versions of $\hat{f}(X)$ would allow for us to gauge and quantify the level of variability present.

In reality, we only have one data set to work with so obtaining sampling distributions numerically as previously described is wishful thinking. However, the bootstrap procedure offers a way to mimic taking multiple samples by using just the one observed sample available. In essence, the single observed sample is picking itself up by its own bootstraps to still obtain a sampling distribution for a statistic of interest. Hence, the name bootstrap. The bootstrap sampling method can be applied to a data set with n observations using 4 easy steps.

1. Determine how many samples you want to generate, B.

2. Generate a bootstrapped data set by sampling n observations from your original data set but doing so with replacement. Compute the statistic of interest and store it.

3. Repeat the process B times, recording the statistics each time.

4. Use the stored statistics to assess the sampling distribution.

Bagging uses bootstrap sampling to assess the variability that a predictive model has and harnesses the power of averaging to produce a more stable prediction. Any regression model can be "bagged" using the following procedure:

1. Generate B boostrap samples from the original data set.

2. For each sample, fit the specified regression model you wish to bag and store the models. Denote these models as $\hat{f}_b(X)$ for $b = 1, 2, ..., B$

3. Predict a future value, x_0, by averaging the predictions over the B fitted models. $\hat{f}_{bag}(x_0) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}_i(x_0)$

The true power of using a bagged model can be summarized in the following way. Complex models suffer from being too variable from data set to data set. Creating multiple bootstrap samples allows us to fit the same complex model. Therefore, its variability can be assessed directly. Averaging the predictions produced by these models allows for their variance to be dampened. Simple mathematical statistics can be used to illustrate the improvement. For a given point x_0, suppose the variance of our estimating procedure $\hat{f}(x_0)$ is σ^2. If we suppose that all $\hat{f}_i(x_0)$ are mutually independent, $Var(\hat{f}_{bag}(x_0)) = \frac{1}{B^2}\sum_{i=1}^{B} Var(\hat{f}_i(x_0)) = \frac{\sigma^2}{B}$. So the variability of a bagged prediction model is lower than just using the predictive modeling tool alone by a scale of $\frac{1}{B}$. Unfortunately the bootstrap samples are not independent samples, so the reduction in variance is typically not as strong as derived here assuming independence.

A random forest model is a predictive model that applies the bagging principle to regression trees. As the name suggest, multiple trees with a large number of splits are built using the bootstrap resamples creating a "forest". To predict a future value, one simply creates the prediction using each tree and then averaging over all of them to produce the final prediction. So why aren't random forest models simply called bagged trees? As alluded to previously, the trees from the bootstrap samples are not indepdendent but rather quite similar to each other. This correlation impacts how much the variance can be dampened by using the bagging approach. To circumvent this issue, [3] proposed only using a subset of the available predictors when building each tree. The subset of predictors used are randomly selected. By using different subsets of predictors, the trees are effectively "decorrelated" and averaging over the predictions dampens the variance to a stronger degree. So the random selection of predictions used to generate $\hat{f}(X)$ for each bootstrap sample is unique to the random forest algorithm and separates itself from bagging.

2.4 Prediction Intervals for RFs

The random forest model is a highly valuable and applied nonparametric form of regression. The trees provide a natural way to automatically approximate $f(X)$ without doing a lot of thinking about what the true from of $f(X)$ looks like. Its bagging nature lends itself to better prediction accuracies than a regression tree and also allows for categorical predictors to be incorporated where other nonparameteric tools, like KNN, do not. The original work provided by [3] only discussed making a single point prediction. But as the popularity of nonparametric regression methods have soared due to the emergence of big data science fields, interest in the developement of prediction interval estimation under nonparameteric regression methods has also began to rise.

Quantile random forests (QRF) was developed by Nicolai Meinshausen [8]. The technical details are provided by expressing the predictions made by a random forest model as a weighted average of the training data,

$$\hat{f}(y|x_0) = \sum_{i=1}^{n} w_i(x_0) y_i.$$

Expressions for the weights $w_i(x)$ can be found in [8]. Briefly, for each tree in the RF model, the observation is weighted either as 0, if x_0 does not belong to the region x_i belongs to, or $\frac{1}{n_{R_j}}$ if it does. If one denotes these weights across all B trees as $w_i(x, b)$ for $b = 1, 2, ..., B$, the final weights are simply averages of the B individual tree weights,

$$w_i(x_0) = \frac{1}{B} \sum_{k=1}^{B} w_i(x, k).$$

Recall that random forest models try to minimize expected squared error loss, and thus are approximating the conditional expectation, $E(Y|x_0)$. Quantile random forest utilizes this fact, to approximate the conditional distribution function(cdf), $F(y|x_0)$. A common statistical theory result states that the conditional distribution function can be expressed via an expectation and is given by,

$$F(y|x) = E(1_{\{Y \leq y\}}|x)$$

where $1_{\{Y \leq y\}}$ is a Bernoulli random variable with "success" defined as $Y \leq y$. Since the random forest model already obtains a mechanism to estimate a conditional expectation for Y, an estimate for the cdf can be obtained by

$$\hat{F}(y|x_0) = \sum_{i=1}^{n} w_i(x_0) 1_{\{Y \leq y\}} \tag{2.8}$$

Quantile regression forest, as an algorithm, can now be summarized:

1. Generate B regression trees as in RF. For each region, take note of all observations rather than simply record their average as in RF.

2. For a given x_0, compute the weights for each observation, $w_i(x_0)$ for $i = 1, 2, ..., n$.

3. Estimate $F(y|x_0)$ by making the calculation in (2.8) for all $Y \in \mathbb{R}$

The value $\hat{Q}_\alpha(x_0)$ is determined numerically by solving for $\hat{F}(\hat{Q}_\alpha(x_0)|x_0) = \alpha$ and prediction intervals can be obtained via the method introduced at the beginning of the chapter. Full implementation of quantile random forests is provided in the **qrf** package within R statistical software.

2.5 Quantile Multiple Linear Regression

Recall from Chapter 1 the general model for multiple linear regression (MLR) is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{2.9}$$

where $\epsilon \sim N(0, \sigma^2)$ and is independent for each observation. Under this framework, estimating $f(X)$ from training data simplifies to estimating the regression coefficients $\beta_0, \beta_1, ..., \beta_p$. For notation purposes, we will express the MLR model as a function of both the predictors and the regression coefficients, $f(X, \beta)$. Using the squared error loss function in (2.1), the regression coefficients are estimated by minimizing the observed residual sums of squares,

$$\min_{\beta} \sum_{i=1}^{n} (y_i - f(x_i, \beta))^2. \tag{2.10}$$

The solution to this minimization problem has a closed form and is commonly referred to as the ordinary least squares (OLS) estimates. Quantile linear regression operates under a very similar framework as OLS. Since minimizing the loss function defined in (2.4) produces conditional quantiles as solutions, quantile linear regression seeks to estimate the regression coefficients by minimizing

$$\min_{\beta} \sum_{i=1}^{n} L_\alpha(y, f(x_i, \beta)). \tag{2.11}$$

This minimization problem is typically handled through linear programming using constrained optimization [1]. The quantile regression package in R utilizes the Barrodale and Roberts simplex algorithm or a Frisch-Newton algorithm depending on the size of the data set. For additional technical details on quantile regression see [6]. Prediction intervals are easily obtained by fitting two seperate quantile regression models, denoted $\hat{f}_{\alpha/2}(x)$ and $\hat{f}_{1-\alpha/2}(x)$. A $(1-\alpha)\%$ prediction interval at x_0 can be expressed as $(\hat{f}_{\alpha/2}(x_0), \hat{f}_{1-\alpha/2}(x_0))$.

2.6 A test for new observations in sparse regions

Predicting a new observation, x_0, in a sparse region of the predictor space where little to no training data exists is a form of extrapolation. This extrapolation can impact prediction performance for nonparametric models as they rely on approximating conditional expectations using training data that are by some definition "close" to the new observation one wishes to predict. Identifying when a new observation is in a sparse region would be helpful knowledge when making the prediction and how much confidence one should have in it.

We developed a diagnostic test to determine if a new observation is in a sparse region of the predictor space relative to the training data that is being used to fit a nonparametric model such as trees, random forests, and KNN. The test is a blending of traditional hypothesis testing and the core concept of KNN. The procedure is as follows.

1. For each training observation ($i = 1, 2, ..., n$), determine the k-nearest neighbors and record their distances denoted d_{ij} for $j = 1, ..., k$.

2. Compute the average distance to each neighbor, $d_i = \frac{1}{k} \sum_{j=1}^{k} d_{ij}$.

3. Compute the test statistic d^* which is the average distance of the k-nearest neighbors to x_0, our new candidate point.

4. Letting D^* be the random variable defined as the average distance of k nearest neighbors among training data, compute the p-value $P(D^* \geq d^*)$ using the d_i's as an empirical distribution for D^*.

5. If the p-value is smaller than some threshold, we flag the observation as potentially in a sparse region.

3 Simulation Studies

The first primary goal of this project was to assess the performance of prediction intervals obtained from nonparametric regression methods when a new prediction scenario, x_0, is given, and it falls outside the range of the original data the model was trained on. Additionally, we wanted to illustrate that providing estimates of prediction interval coverage derived directly from the training set is not a good indicator of how point prediction coverages will behave, especially in sparse regions.

After illustrating through simulations that identifying problematic observations is critical to avoid poor prediction interval performance, our second objective was to investigate the performance of our statistical test provided in Chapter 2 in regards to its ability to identify problematic points. This chapter will be presented in two main sections, one for each objective. Each section will provide the details of our simulation procedure and scenarios, performance metrics, and a discussion of the results we obtained.

3.1 Prediction Interval Performance in Sparse Regions

Recall from the discussions in Chapter 1, the general regression model takes the form

$$Y = f(X) + \epsilon. \tag{3.1}$$

The general simulation workflow is as follows. We developed six scenarios that highlight different aspects of the general regression model which includes, defining $f(X)$, sampling values from the predictor space, and creating sparse regions within the predictor space. Under these scenarios, we simulated 10,000 data sets of 1000 observations each and produced 80% prediction intervals using quantile linear regression, QKNN, and QRF on numerous values of x_0 located in regions of the predictor space varying from within the training data to extremely sparse regions. We then determined if the prediction intervals contained new observed response value at that location. The percentage of prediction intervals that encompassed a newly generated data point was recorded for each method as well as the average widths of the intervals. Additionally, we computed how well the method's prediction intervals covered the observed training data. Pseudocode for our R script is listed below:

```
1. For $i$ in $1 : 10000$
      a. Create new set of training data.
      b. Build the quantile regression, QKNN and QRF models.
      c. Make prediction intervals for the series of test points.
      d. Create a new random response at the given test points and check if the
   new value falls in the prediction interval and store result as a count.
      e. Store the widths of each interval.
      f. Store the training coverage.
2. End Loop
3. Calculate the overall coverage rate, average widths, average training coverage
```

For our six scenarios, we used two main functions to generate our response variable. Both of them utilized two numeric predictors. The first was of a common MLR type model that includes an interaction term between the two predictors. The response surface is a curved plane and is $f(x_1, x_2) = 6.75 + 10x_1 + 18.8x_2 + 41x_1x_2 + \epsilon$. The second function,$f(x_1, x_2) = \sin(10x_1) + x_2^2 + \epsilon$, was used to generate a more complex response surface to see if that had any impact with the nonparametric methods.

In both functions the error terms were randomly generated from independently and identically distributed normal distributions with a mean of 0 but the standard deviation for each function are a little different. For the plane, the standard deviation is 3, while the complex surface had a standard deviation of 0.5. It should be noted for quantile regression it is assumed that the true $f(x)$ is known. This served as a control for our simulation study.

The two predictors were simulated using one of two scenarios as well, both of which fall on the unit square. The first strategy was more ambitious. Under this setting, the two predictor variables were randomly sampled from 4 bivariate normal distributions (250 observations each). The four mean vectors are respectively: $(.4, .4), (.4, .6), (.6, .4), (.6, .6)$. The variance-covariance for all four distributions were the same and is simply $.01I$ where I is a 2×2 identity matrix. Observations that fell outside of the unit square, were deleted and re-sampled. The point of this particular scenario was to generate a predictor space that has much denser regions in some parts than others. An example of this situation can be see in Figure 3.1. For the second scenario, the two predictor variables were randomly sampled from independent uniform random variables on (0,1). Unlike the normal distribution approach, the density of points is uniform across the whole unit square as seen in Figure 3.2.

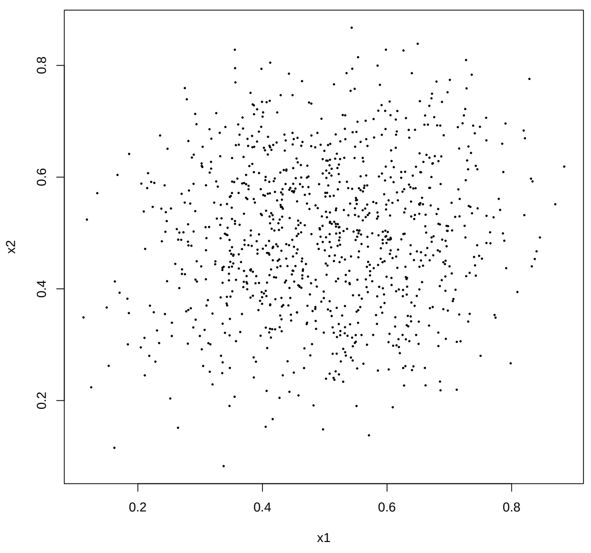


Figure 3.1: Predictor Space with Normal Generation

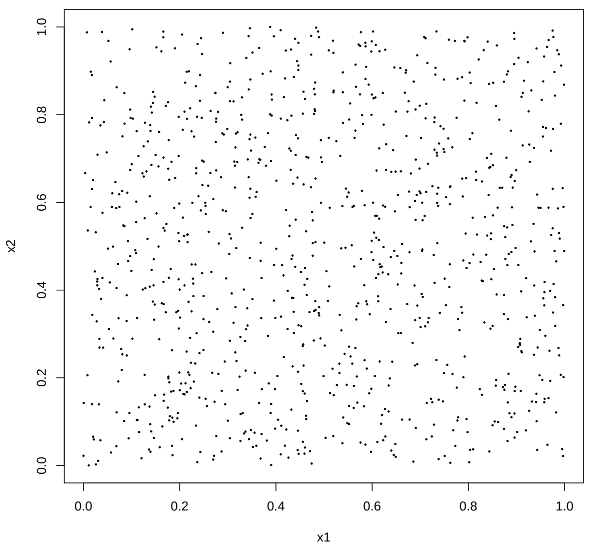


Figure 3.2: Predictor Space with Uniform Generation

When the predictors are sampled, there really are no definitive sparse regions. We generated sparsity in the predictor space by checking if any of the observed coordinates fell inside of a predefined circle. If they did, then they were deleted, creating a sparse region where no data exists. In the plane scenario we eliminated all of the data in a circle whose center was $(.5, .5)$ with a radius of .1. In the complex scenario we used a center to $(.8, .8)$ still with a radius of .1.

Using the above methodology our 6 scenarios considered are:

I Complex response, normal predictors, hole at (.8,.8)

II Complex response, uniform predictors, hole at (.8,.8)

III Plane response, normal predictors, hole at (.5,.5)

IV Plane response, uniform predictors, hole at (.5,.5)

V Complex response, normal predictors, hole at (.5,.5)

VI Complex response, uniform predictors, hole at (.5,.5)

The candidate points used to generate prediction intervals when the center of the hole was located at (.5,.5) were $(.5, .5), (.55, .5), (.59, .5), (.6, .5),$ and $(.61, .5)$. The candidate points used in the scenarios with the hole centered at $(.8, .8)$ were $(.8, .8), (.7646, .7646), (.7363, .7363), (.7292, .7292),$ and $(.7222, .7222)$. In both cases, the points were chosen to assess the progression of the prediction interval performance. The first point is located at the center, while the additional points migrate outwards getting closer and closer to the training data and eventually coinciding with the training data. Both Figure 3.3 and Figure 3.4 plot the points under consideration against a sample of training data from a simulation run for illustration.

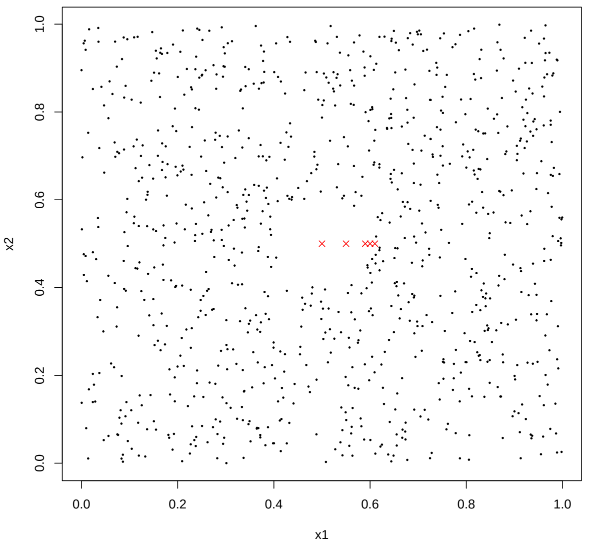


Figure 3.3: Simulated training data from the uniform predictor space with hole at (.5,.5) along with the candidate test points (highlighted with red crosses) to assess prediction interval performance.

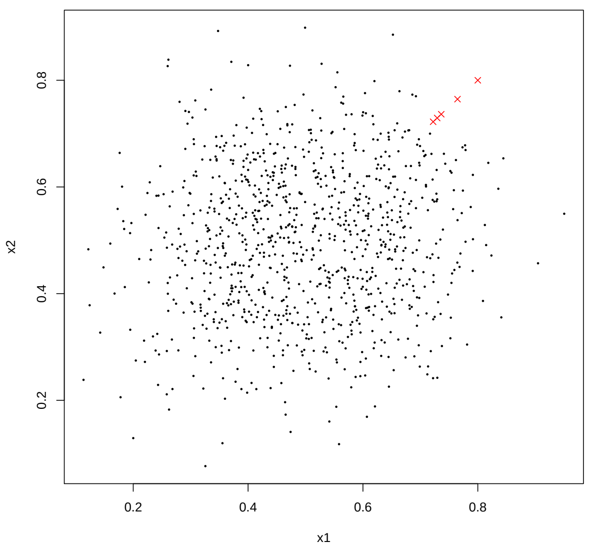


Figure 3.4: Simulated training data from the normal predictor space with hole at (.8,.8) along with the candidate test points (highlighted with red crosses) to assess prediction interval performance.

3.1.1 Sparse Simulation Results

The coverage rates for Scenario I, which was generated under the complex surface with the predictors generated using the normal distribution with the hole located at (.8,.8), is provided in Figure 3.5. Each bar represents the percentage of new random responses that were covered by the prediction intervals generated by each prediction interval method and at each of the candidate points. Each point in the graph is one of the candidate points from the scenario, Point 1 refers to (.8,.8), the point in the middle of the sparse region, Point 2 refers to (.7646,.7646) and so on. The dotted line is the nominal, expected coverage percentage of 80%.

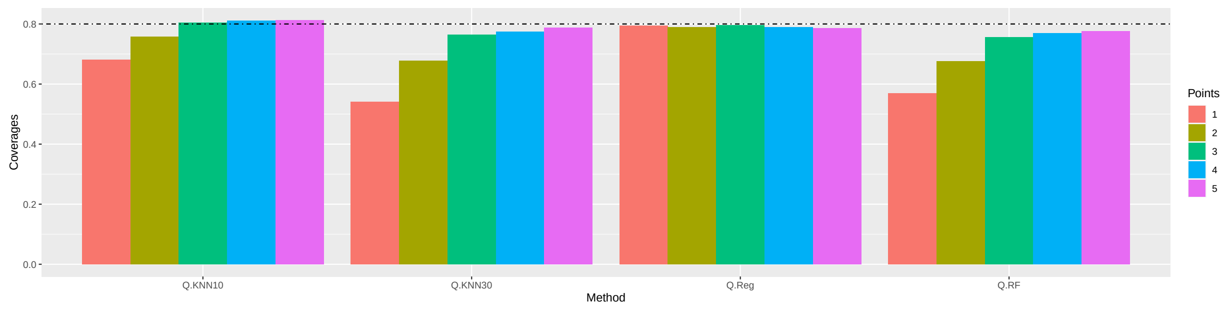


Figure 3.5: Prediction coverages of QKNN, Quantile Regression, and QRF for Scenario I. Point 1 corresponds to the center of the hole at $(.8, .8)$, Point 2 at $(.7646, .7646)$, and so on to Point 5 at $(.7222, .7222)$ within the training data. The nominal threshold is 80% (dotted line).

Upon examining Figure 3.5, it is clear that all three nonparametric approaches have reduced coverage performance, some as much as 10% below the nominal threshold of 80%, when making predictions inside the sparse region. The coverage improves the closer the candidate point gets to the training data. As expected, quantile regression performs consistently across the board as we specified the correct form of f when performing the simulations. A similar pattern emerges when examining the result of Scenario II, which is identical to Scenario I except for the predictors are sampled uniformly. Figure 3.6 shows the nonparametric methods tend to have poorer coverage when in the sparse region except for one caveat involving QKNN. When the candidate point was directly in the center of the sparse region (hole), the coverage probabilities were slightly above nominal. This behavior only happens for QKNN and was not observed for QRF.

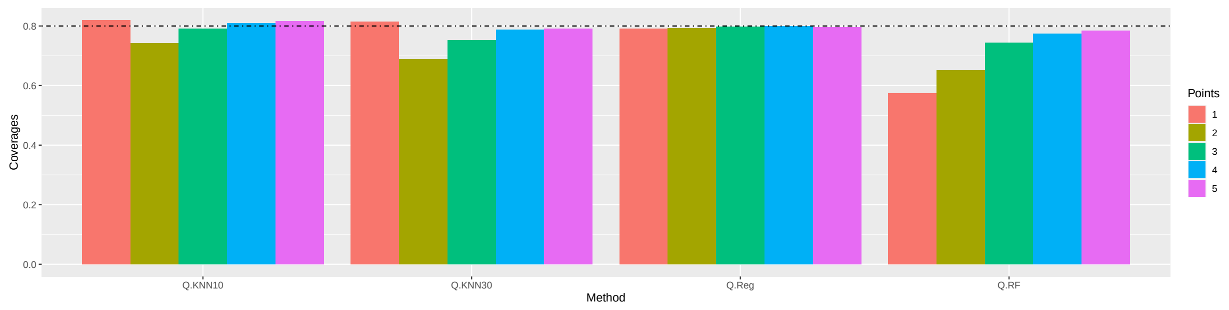


Figure 3.6: Prediction coverages of QKNN, Quantile Regression, and QRF for Scenario II. Point 1 corresponds to the center of the hole at $(.8, .8)$, Point 2 at $(.7646, .7646)$, and so on to Point 5 at $(.7222, .7222)$ within the training data. The nominal threshold is 80% (dotted line).

Our best explanation for this behavior is that under Scenario I, there is effectively no training data to work within the area of the predictor space above and to the right of the sparse circle centered at $(.8, .8)$. This creates a more traditional extrapolation setting and all the nonparametric approaches suffer. Under the the uniform setting of Scenario II, there is more available data in this upper most quadrant of the unit square. This allows for QKNN to utilize data not just on the bottom left side of the hole close to $(.8, .8)$. We believe the trees being built up through random forest partitions the predictor space in such a way that always keeps the regions separated. This hypothesis was motivated by examining the average prediction interval widths as seen in Figure 3.6. The interval widths are globally consistent, regardless of which point is being used for prediction, except for the center which has a much wider average width for QKNN. This higher variance, we believe is coming from utilizing training data in the upper most quadrant of the unit square. In the end, it is our opinion that this is more of a fluke situation in which a perfect storm has happened in which performance looks good, but in general, the problem still persists as we can see by simply moving the candidate point just off of center and the prediction coverage worsens.

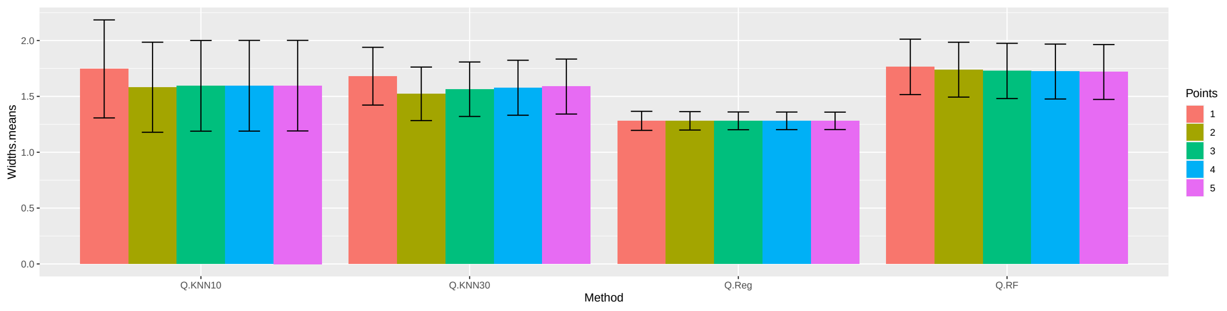


Figure 3.7: Prediction Interval Widths from Scenario II. Point 1 corresponds to the center of the hole at $(.8, .8)$, Point 2 at $(.7646, .7646)$, and so on to Point 5 at $(.7222, .7222)$ within the training data.

Scenario III offers some insight into how predicting into sparse regions behave when the level of complexity of f is much simpler. The coverage estimates of this simulation scenario can be seen in Figure 3.8. The coverage probabilities are much closer to nominal or slightly conservative and are overly conservative at the center of the sparse region, $(.5, .5)$ for this scenario. The QRF method tends to be more conservative even as the candidate points move closer to available training data.

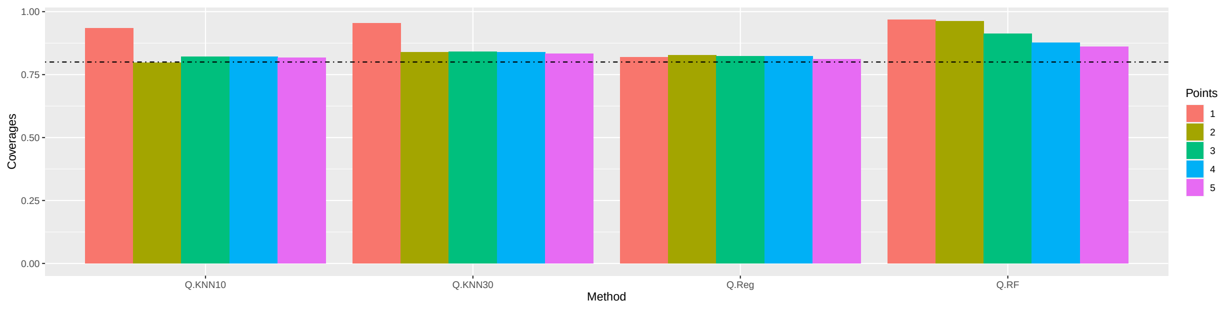


Figure 3.8: Prediction coverages of QKNN, Quantile Regression, and QRF for Scenario III. Point 1 corresponds to the center of the hole at $(.5, .5)$, Point 2 at $(.55, .5)$, and so on to Point 5 at $(.61, .5)$ within the training data. The nominal threshold is 80% (dotted line).

The remaining three scenarios offer similar conclusions and results as the ones previously discussed so we did not include them in the report. In summary, our simulations provide evidence that prediction interval performance can greatly depend on the location of the point you are trying to predict with respect to the training data set being used. This is an important discussion and illustration due to the fact that many nonparametric prediction interval developments are largely justified by the claim that they work based on computing coverage probabilities of the training data. There is no discussion of the limitations of these approaches when applying them to new observations that are in sparse regions of the predictor space. Perhaps it is assumed by the authors that people are aware of the issue, but it is our experience that there is no real thought given to this issue and there is relatively no investigation of the predictor space in practice, outside of traditional parametric techniques like leverage values and standardized residuals.

To illustrate this problem, for each of our scenarios we also produced coverage estimates using the training data set for each simulation run. The average coverage probabilities are listed below in Table 3.1. It is important to understand that training coverage probability is obtained across the whole predictor space and not just an examination of one point. The prediction intervals being made will always be within the training data so the sparsity issue is avoided. When examining the training coverages, all methods perform close to nominal or more conservatively which contradicts many of the simulation coverages we explored previously at specific points. While using training coverage probabilities to verify that a method is working "as advertised", this should come with a more vocalized disclaimer that the method will behave well when predicting new observations that are very similar to those found in the training data.

Method/Scenario	I	II	III
QR	0.7984	0.7985	0.7985
QRF	0.7821	0.7836	0.8841
KNN30	0.8007	0.8020	0.8314
KNN10	0.8002	0.8005	0.8123

Table 3.1: Overall training coverage for each method in Scenarios I, II, and III

3.2 Detecting points in sparse regions

An examination of prediction interval widths, such as those found in Figure 3.6, offer a segue way into our next simulation discussion. For additional evidence, Figure 3.9 below, provides the interval widths for Scenario III. Our hope, after examining the coverage simulation results, that the widths of the prediction intervals themselves could offer insight as to when a new observation is potentially in a sparse region of the predictor space. This proved to be a poor strategy for the following reasons. Outside of the wider intervals at the center of sparse region as seen in Figures 3.6 and 3.9, the interval widths are relatively consistent across the other candidate points regardless of their proximity to the training data. Additionally, nonparametric interval estimation is affected by non-constant variance of the error terms. In other words, the interval widths can change drastically if the variability in the response depends on the location of x_0 in the predictor space. This is true even when predicting in areas where the training data is dense. A good visual example of this issue can be found in Figure 3 of [8], which produces a graphical visualization of prediction intervals from a famous Boston housing market data set often used in the development of machine learning tools. The intervals were derived using points within the training data set, yet have interval widths that vary quite drastically.

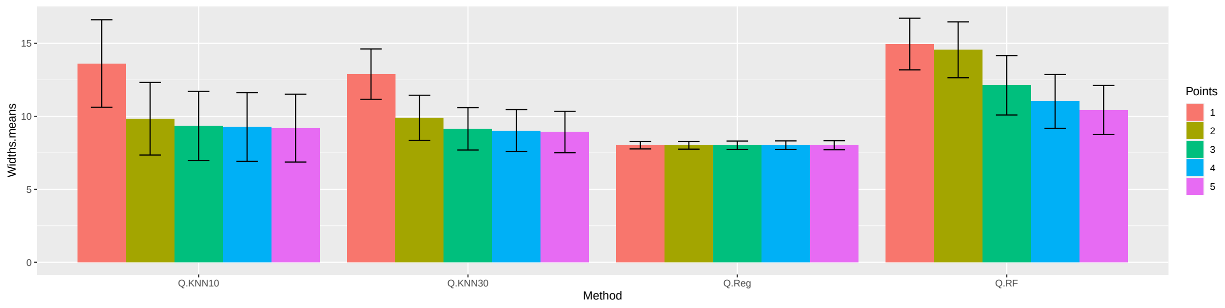


Figure 3.9: Prediction Interval Widths from Scenario III. Point 1 corresponds to the center of the hole at $(.5, .5)$, Point 2 at $(.55, .5)$, and so on to Point 5 at $(.61, .5)$ within the training data.

Since we could find no other useful information from the prediction intervals themselves, we developed a diagnostic test to determine if a new observation is indeed in a sparse region of the predictor space as discussed in Chapter 2. To investigate the performance of this test, we performed "power" simulations under various predictor spaces. The first two situations follow the predictor space scenarios of III and IV. Either the two predictors were generated using uniform or normal distributions with a hole centered at $(0.5, 0.5)$. A third scenario was also generated using 2 independent Beta$(1/3, 3)$ with a hole centered at $(0.3, 0.3)$. Figure 3.10 provides simulated data sets under the three scenarios along with candidate points in which we will investigate how well the test performs. Under scenarios where the hole is at $(0.5, 0.5)$, candidate x_0 values were sampled along the horizontal line from $(0.4, 0.5)$ to $(0.5, 0.5)$ as depicted by the red crosses in the left and middle plots of Figure 3.10. Under the Beta distribution scenario, the candidate points were sampled along the 45 degree line starting at $(0.23, 0.23)$ and ending at the center of $(0.3, 0.3)$. The motivation for the points selected was similar to the previous simulation work. We expect that our test should not flag points that are within, or very close to available training data, but the test should be able to flag points that are much closer to the the center of the circle at a much higher frequency.

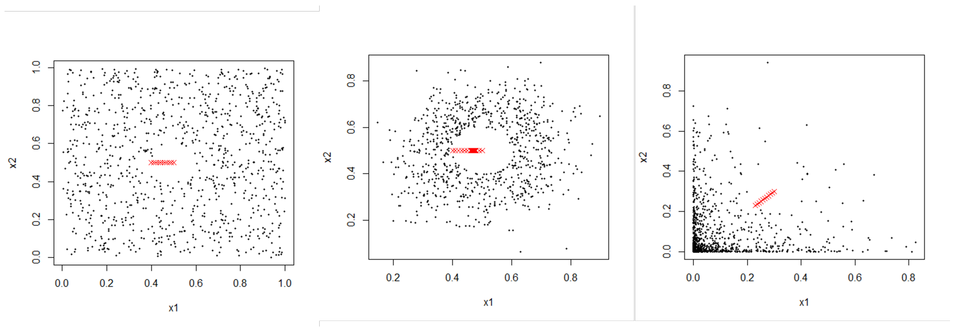


Figure 3.10: The predictor spaces for each of the scenarios with the test points highlighted.

The power simulations were conducted as follows. For each scenario, a random sample of 1000 observations were drawn and our testing procedure was applied to each of the candidate points using $k = 30$ neighbors and a significance testing threshold of 0.05. It is then observed if each test was rejected and the point was concluded to be in a sparse region or not. The process was then repeated 10,000 times and the proportion of rejected tests (power) at each candidate coordinate was recorded and plotted to examine the test's performance. Figure 3.11 provides a plot of the power versus the x coordinate of the candidate point being testing. For example, the power for the candidate point $(.471, .5)$ can be found in Figure 3.11 by examining the power value associated with the x-axis at 0.471 which is roughly 0.5. So this candidate point was determined to be in a sparse region in 50% of the tests.

Upon looking at Figure 3.11 as a whole from left to right, one can see that for points relatively close to the training data, the power of the test is relatively low as expected. Once the candidate points venture farther out toward the center, the power increases and the test is able to detect the points in the sparse region more frequently. For the normal distribution, there is clear demarcation where the test performs poorly and then ramps up to great power values ranging above 0.8. Naturally, for any test, we would like to see the increase in power happen as the soon as the candidate points are in the sparse space. For this scenario, it takes the points going over half of the radius of the circle into the sparse region before good power is obtained, a property that could potentially be improved through future research.

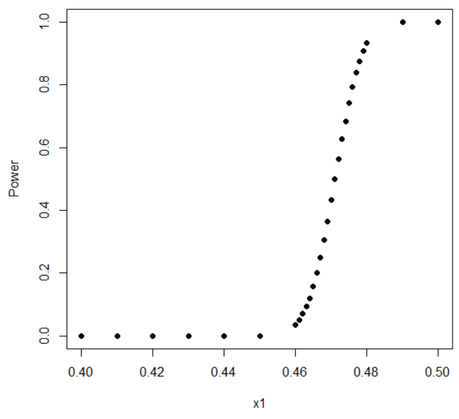


Figure 3.11: The power at each of the highlighted test points for the Normal scenario

Figure 3.12 provides the power results for the uniform scenario with the hole again centered at $(0.5, 0.5)$, which has some unique differences from the previous normal case. Rather than a steep increase in power once the points are extended far enough away from the training data, the power gradually increases over the whole range of candidate points. This results in the test having better power at detecting points that are closer to the training data but are still in the sparse region. For example, the power at $(0.46, 0.5)$ is just below 0.8 under the uniform setting while the power is less than .10 under the normal setting.

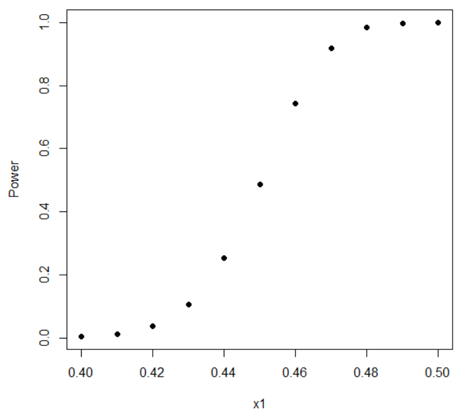


Figure 3.12: The power at each of the highlighted test points for the uniform scenario

Figure 3.13 provides the power results for the beta scenario with the hole at $(0.3, 0.3)$. The power behaves very similarly to the other scenarios where good power values above 0.8 tend to occur once the candidate points are roughly half way into the circle. The reason for this additional scenario was for two reasons. One was our interests in looking at a simulation that behaves like numerous genetic study data that use gene transcripts counts to predict patient outcomes such as cancer status or serology measurements. The second was to offer a scenario that is closer to traditional extrapolation.

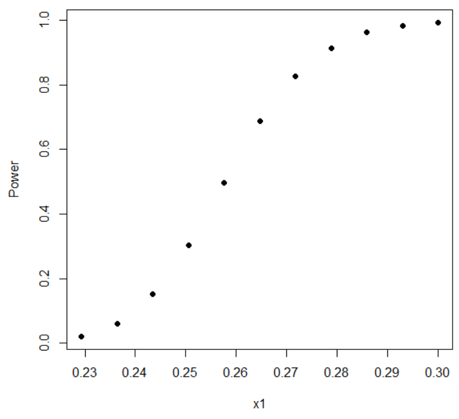


Figure 3.13: The power at each of the highlighted test points for the beta scenario

In summary, the power simulations show that our candidate test is performing as expected on the limited number of scenarios we considered. The performance of the power of the test seems to depend on the distribution of the points in the predictor space in which the sampling distribution of the test statistic is derived. For the normal distribution, the sampling distribution of the average distances are heavier tailed requiring a larger distance before a rejection of the test occurs. Additional discussion on improvements, pitfalls, and practical issues involving the test are discussed in the next chapter.

4 Final Remarks and Future Work

The investigations and method development of this manuscript is somewhat broad but not very deep in certain aspects. It is important to note various pitfalls, considerations, and improvements to gain more insight and to answer more specific questions in the future. To discuss these issues, we have broken the discussion down into two sections. The first is in regards to point and prediction interval methods as a whole when dealing with sparse observations and the second covers various aspects of our developed test.

4.1 Prediction Intervals in Sparse Regions

While our simulations provide some good examples of how prediction intervals obtained on candidate x_0 values in sparse regions can impact performance, an investigation to examine point predictions and the observed mean squared error loss could help establish and articulate the problem more clearly. Our focus on prediction intervals was in hopes of using information from the prediction intervals to identify problematic points which the results of this manuscript have suggested that they do not provide any meaningful information on sparsity.

Our simulations of prediction interval coverages included 3 methods across 6 different scenarios involving the choice of $f(X)$, the predictor space, and the sparsity. A deeper investigation of each prediction interval method could yield better insight into the cause of the performance change for better or for worse in more settings than we considered. For example, what if the sparse regions were rectangular, rather than circular? How do these issues scale when extending to a larger, higher dimensional predictor space? How does sample size play a role? These type of questions can be examined more closely for each individual prediction interval method to determine a more precise summary of the pros and cons of each method.

For QKNN, while additional methods exist to choose an appropriate k, the assignment of neighbors can potentially be affected by including irrelevant predictors in addition to relevant predictors into the model. This issue, also known as "the curse of dimensionality" is widely known to cause problems in point predictions. Quantifying this issue on prediction interval performance would be another interesting investigation to pursue.

4.2 Limitations of Our Statistical Test and Future Improvements

There are a few concerns and logistical issues that need to be further investigated and understood before recommending our test in real applications. In some situations, new data sets in which prediction intervals are needed suffer from global shifts due to instrumentation. For example, suppose the predictor space on the training data set involving 2 predictors covers the unit square, but the new data being generated has suffered a constant shift of one in a single predictor variable. Under this setting, the new observed values will all be outside the range of the original predictor space and our test would rightfully flag many of the candidate points as problematic. This may not necessarily be needed if a simple normalization step can place the new candidate points back in the original predictor space. These type of issues are commonly seen and corrected for in genomics studies using technologies such as microarrays, metabolomics, and RNA sequencing where appropriate controls are incorporated to allow for the normalization to occur. The test should be applied when the new observations are truly on the same scale as the original training data.

Our preliminary simulations suggest that our test is working as expected, rejecting much more frequently when a point is clearly in a sparse region. However, the rejection rate is much lower than the nominal value of 0.05 when the test points were in the regions that were not sparse at all. Further investigation and adjustments here could help increase power. Additionally, modifications could be made to handle harder situations when the point is still in sparse regions but close enough to training data to not be detected by the test. One adjustment to the test that could potentially address this is to incorporate the direction of the closest neighbors. These directions could then be used as weights when computing the test statistic and could give more extreme values to points whose neighbors are all coming from similar directions.

It seems natural to suspect that the performance of our test will depend on the choice of k. We did not investigate this outright but the affects of this choice and a procedure to potentially choose the best k should be developed and investigated. In a similar fashion, the "curse of dimensionality" should also be investigated to see how it can impact the performance of our test. Finally, the test is only valid for problems in which the predictors are numeric. Providing additional methodology to tackle predictor sets that also include categorical variables would also be extremely beneficial as it would capture a much larger set of real world problems.

4.3 Summary

The main goals for this thesis were to provide a brief summary of current prediction interval methods for common nonparametric tools and investigate their performance when a user requests a prediction interval for a new candidate observation that falls in sparse regions of the predictor space. Our simulations were designed to provide evidence, in a controlled setting, that nonparametric prediction intervals do not offer any particular protection when trying to extrapolate in sparse regions. We also feel that it is important that researchers developing new nonparametric prediction interval methods should consider examining prediction interval coverages using the simulated framework described in this manuscript versus using simple training coverage as a means to illustrate that the method is performing as intended.

In addition to these investigations, our brief literature review yielded no real discussion on these issues or how to identify problematic candidate points, which resulted in the development of a new statistical test to determine if a new point falls in a spares region or not. In a limited set of simulations, we have shown that our test behaves in a reasonable and expected way when examining its statistical power, and we look forward to extending the method to determine an optimal choice of k and developing a set of best practices when using the test.

BIBLIOGRAPHY

[1] Koenker Bassett, *Regression quantiles*, Econometrica **46** (1978), no. 1, 33–50.

[2] P K. Bhattacharya and A K. Gangopadhyay, *Kernal and nearest-neighbor estimation of a conditional quantile*, Annals of Statistics **18** (1990), no. 3, 1400–1415.

[3] Breiman, *Random Forests*, Machine Learning **43** (2001), no. 45, 5–32.

[4] K F. Cheng, *Nonparametric estimatros for percentile regression functions*, Commun. Stat.-Theor. M. **12** (1983), no. 6, 681–692.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, Springer, 2013.

[6] R Koenker, *Quantile regression*, Cambridge University Press, 2005.

[7] X. Ma and X. Shi, *A variant of K nearest neighbor quantile regression*, Journal of Applied Statistics **43** (2016), no. 3, 526–537.

[8] Meinshausen, *Quantile regression forests*, Journal of Machine Learning Research **7** (2006), 983–999.

[9] Efron Tibshirani, *An introduction to bootstrap*, Chapman and Hall/CRC, 1994.

VITA

Jackson Faires received a Bachelor's degree in Mathematics from Stephen F. Austin State University in 2016. She began her work towards a Master's degree in Mathematics at Stephen F. Austin State University in the Fall of 2019 and is expected to graduate in August 2021.

Permanent Address:	PO Box 13040 SFA Station
	Nacogdoches, TX 75962

The style manual used in this thesis is A Manual For Authors of Mathematical Papers published by the American Mathematical Society.

This thesis was prepared by Jackson Faires using LaTeX.