AN ABSTRACT OF THE DISSERTATION OF

Faraz Niyaghi for the degree of Doctor of Philosophy in Statistics presented on February 7, 2019

Title: Localized Variable Selection with Random Forest

Abstract approved: _____ _____
Sarah C. Emerson Sharmodeep Bhattacharyya

Due to recent advances in computer technology, the cost of collecting and storing data has dropped drastically. This makes it feasible to collect large amounts of information for each data point. This increasing trend in feature dimensionality justifies the need for research on variable selection. Random forest (RF) has demonstrated the ability to select important variables and model complex data. However, simulations confirm that it fails in detecting less influential features in presence of variables with large impacts in some cases. In this dissertation, we propose two algorithms for localized variable selection: clustering based feature selection (CBFS) and locally adjusted feature importance (LAFI). Both methods aim to find regions where the effects of weaker features can be isolated and measured. CBFS combines RF variable selection with a two-stage clustering method to detect variables where their effect can be detected only in certain regions. LAFI, on the other hand, uses a binary tree approach to split data into bins based on response variable rankings, and implements RF to find important variables in each bin. Larger LAFI is assigned to

variables that get selected in more bins. Simulations and real datasets are used to evaluate these variable selection methods. Finally, we also propose an extension to CBFS for localized prediction.

Localized Variable Selection with Random Forest

by
Faraz Niyaghi

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented February 7, 2019
Commencement June 2019

Doctor of Philosophy dissertation of Faraz Niyaghi presented on February 7, 2019.

APPROVED:

_____

Co-Major Professor, representing Statistics

_____

Co-Major Professor, representing Statistics

_____

Chair of the Department of Statistics

_____

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

_____

Faraz Niyaghi, Author

ACKNOWLEDGEMENTS

Without any hesitations, my biggest appreciation goes my parents, Ali Niyaghi and Manijeh Asadiniyazi. I wouldn't be where I am today if it wasn't for their unconditional support. They were the ones who motivated me to start my graduate studies in the first place.

My sincere gratitude goes to Dr. Sarah C. Emerson who is more than an advisor to me. I would like to think of her as my role model in life. I would also sincerely thank Dr. Sharmodeep Bhattacharyya who granted me the opportunity to work with him as my co-advisor. This research wouldn't be possible without all the support from my advisors.

I would like to thank my committee members, Dr. Charlotte Wickham, Dr. Katherine Mclaughlin, and Dr. Bogdan Strimbu. My special acknowledgment goes to Dr. Charlotte Wickham who assisted me as my co-advisor at the beginning of this Journey. I am eternally thankful that I had the honor to meet and work with Charlotte.

Finally, I would like to cordially thank my lovely partner, Sogol Sadat Haddadi. I have no idea how she managed to bear with me during this challenging journey. I can't thank her enough for her support and care.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Table Page

LIST OF TABLES (Continued)

Table Page

LIST OF ALGORITHMS

DEDICATION

This thesis is dedicated to my parents, Ali Niyaghi and Manijeh Asadiniyazi, who never hesitated to sacrifice anything for my success.

CHAPTER 1: INTRODUCTION AND BACKGROUND

1.1 Introduction

Variable selection is a statistical procedure with applications in many problems, and its importance has grown in tandem with the increasing size of datasets. It can be loosely characterized as any method for identifying a good subset of original input variables. These methods could be aiming for different things: low prediction error, parsimonious feature space, and so on. Not all variable selection procedures aim to achieve the same goal. Mitchell and Beauchamp provided four main reasons for variables selection: 1) simplifying the relationship between response and explanatory variables; 2) identifying the influential and negligible set of variables; 3) reducing computation cost for predictions; and 4) improving the accuracy of predictions and estimations [1]. One application of variable selection is in genomics where the number of features (genes expressions) are much larger than the number of under study patients. Research by Guyon et al. on identifying key genes in cancer diagnosis is an example of such studies [2].

Variable selection approaches heavily depend on the associated statistical problem. So, procedures of variable selection in presence of response, that is supervised learning, can be quite different from procedures of variable selection in datasets without a response variable, that is unsupervised learning. For example, principal component analysis (PCA) is a variable selection technique primarily for unsupervised cases since PCA does not use the response variable information in any way for variable selection [3]. Partial least squares

(PLS), on the other hand, is a variable selection method which uses the information of response variables and is suitable for variable selection in regression-based problems only [4], [5]. There are certain variable or model selection techniques which are generalizable to both supervised and unsupervised problems, such as Akaike information criterion (AIC), Bayesian information criterion (BIC), minimum description length (MDL) and so on [6]–[8]. These procedures depend on the information associated with the data and which model best uses that information. Thus, these procedures can be used in both supervised and unsupervised problems.

There are several techniques for variable selection including Least Absolute Shrinkage and Selection Operator (LASSO), PCA, and Random Forest (RF) [9], [10]. This chapter provides an overview of common variable selection methods with emphasis on RF, which is extensively used in this research. Most variable selection methods, like the ones considered in this chapter, focus on finding important features on a global level. This can result in ignoring locally important features. This is illustrated as a motivational example in the last section of this chapter.

In this dissertation, we propose two algorithms for localized variable selection: clustering-based feature selection (CBFS) and locally adjusted feature importance (LAFI). CBFS, which is introduced in chapter 2, combines RF variable selection with a two-stage clustering method to detect variables whose effect can be isolated only in certain regions. Chapter 3 discusses the challenges of using this algorithm for local prediction purposes. Chapter 4 presents LAFI, which uses a binary tree approach to split data into bins based on response variable rankings. Next, it implements RF to find important variables in each bin.

Larger LAFI scores are assigned to variables that get selected in more bins. This dissertation concludes with a summary of our findings and suggestions for future research directions.

1.2 Background on Variable Selection Methods

Variable selection methods can be broadly classified into supervised and unsupervised categories. This section serves as a literature review for widely used variable selection methods in these categories.

1.2.1 Supervised Variable Selection

Supervised variable selection problems start with considering a set of n data points $S = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ where each X_i is a p-dimensional feature vector and Y_i is the response variable which is either categorical or numerical. These methods aim to find a subset of features or some combinations of features to either explain the variability in the response variable (inferential purposes) or to make better predictions for unseen data. Sequential variable selection techniques, Linear Discriminant Analysis (LDA), and regularized regression methods are introduced in this section.

1.2.1.1 Random Forest

RF is a nonparametric method which is described in section 1.3. It is a great candidate for variable selection in presence of linear and nonlinear patterns in response surfaces because

it is not constrained by a linear framework. In addition, it can be used in both classification and regression problems. This section briefly mentions some of the previous studies on variable selection with RF.

Hapfelmeier and Ulm divide RF variable selection methods into two major classes: performance-based and test-based approaches [11]. Performance-based methods fit several RF models with different input feature sets to find the best model in terms of prediction accuracy. The features included in the best performing model are the selected features. Test-based methods, on the other hand, use hypothesis testing to evaluate the significance of feature importance (FI) for variables.

Algorithms presented by Svetnik et al., Jiang et al., Díaz-Uriarte and Alvarez de Andrés, and Genuer et al. are examples of performance-based methods [10], [12]–[14]. The algorithm proposed by Genuer et al. is used in chapter 4 of this dissertation. First, this algorithm sorts features based on their mean FI obtained from fitting multiple RFs. Next, it fits nested models starting with the model including only the most important variable, and iteratively adds the next most important variables to the input feature space. In general, features in the model with the highest prediction performance are selected. Further details on this algorithm are provided in the following chapters.

The main idea of test-based methods is to permute feature values to obtain an empirical distribution of FIs, and use that distribution to calculate the corresponding *p*-value. Features with *p*-values smaller than a threshold, usually 0.05, are selected. Altmann et al., Rodenburg et al., Wang et al., and Tang et al. present similar algorithms in their articles

[15]–[18]. For an extensive literature review of RF variable selection methods see *A new variable selection approach using Random Forests* by Hapfelmeier and Ulm [11].

1.2.1.2 Sequential Variable Selection

Sequential variable selection was introduced in the regression setting by Efroymson in 1966 [19]. Backward, forward, and bidirectional selections are examples of this method. Backward selection starts with a saturated model including all features. At each step, it drops a variable according to some criteria such as AIC, BIC, or *p*-value, e.g. it might drop the variable with the largest p-value. The selection process stops when all variables remaining in the model have *p*-values smaller than a user-defined threshold, which is also known as significance level or type 1 error level. Forward selection does this process in reverse order. It starts with the model including only an intercept. Then, iteratively, the algorithm adds variables with the smallest *p*-values until no remaining candidate variable results in a *p*-value smaller than the predefined threshold. Bidirectional selection algorithms consider both adding and dropping of variables at each step.

Sequential variable selection methods have been used in a variety of fields ranging from genomics to chemistry [20], [21]. They are a simple, computationally efficient, and intuitive collection of algorithms. However, they are also greedy and can result in a local optimal set of features because they do not consider all possible combinations of features.

1.2.1.3 Linear Discriminant Analysis

One of the earliest applications of LDA comes from Fisher's paper on classification of Iris versicolor and Iris setosa species [22]. LDA is a classifier which can be also used as a dimension reduction technique. A typical application of LDA is in speech recognition tasks, where the number of features are much larger than the number of classes [23]–[25]. LDA aims to find linear combinations of features which result in the best separation of classes. Intuitively, separation increases as the distance between center points of classes gets larger and within class variations get smaller. Let us define $W_i = \sum_P \alpha_p x_{ip}$ as a linear combination of features. Subsequently, $\bar{W}_c$ is the center of the c^{th} class. Then, in a two class problem, the separation is defined as following [26]:

$$Separation = \frac{|\bar{W}_1 - \bar{W}_2|}{S_W} \tag{1}$$

Where S_W is the pooled standard deviation:

$$S_W = \frac{\sum_{i=1}^{n_1}(W_{1i} - \bar{W}_1) + \sum_{i=1}^{n_2}(W_{2i} - \bar{W}_2)}{n_1 + n_2 - 2} \tag{2}$$

Note that the separation varies with different weights of features, i.e. the α_p. LDA finds the best set of weights that maximizes this separation. This optimal combination can be thought of as a dimensionally reduced presentation of feature space from p to 1. Rao introduced an extension of LDA for more than two classes [27]. Refer to Johnson and Wichern's *Multivariate Analysis* for more details on LDA and its extensions [26].

1.2.1.4 Regularized Regression

Regularized regression models such as LASSO, elastic net, and variants (e.g. group LASSO and fused LASSO), smoothly clipped absolute deviation (SCAD), minimax concave penalty (MCP) are considered modern variable selection methods [28]–[32]. These methods are essentially constrained optimization techniques: they find coefficient estimates that minimize an objective function (typically either the negative log-likelihood of the data given the parameters, error sums of squares, or a pseudo-likelihood) within a subspace of the parameter space, where the subspace is defined by a bound on some penalty function such as the L_1 or L_2 norm of the vector of parameter values.

LASSO is probably the most widely used regularized regression method. It has been used in several fields of study including genetics and survival analysis [33], [34]. In a regression setting, the LASSO estimator is calculated by:

$$\hat{\beta} = argmin_{\beta}\{(Y - X\beta)^T(Y - X\beta) + \lambda\|\beta\|_1\} \quad (3)$$

LASSO coefficients vary for different values of λ. Larger values of λ result in smaller coefficients and a sparser feature space. The value of λ is usually obtained through cross validation.

1.2.2 Unsupervised Variable Selection

Unsupervised variable selection problems only consider features and do not have a labeled response variable. The main objective here is to reduce feature dimension while maintaining most of the information in the original features.

1.2.2.1 Principal Component Analysis (PCA)

PCA was originally introduced by Hotelling in 1933 [35]. The main idea is to find linear combinations of features that account for most of the variability in the original data. Principal components are eigenvectors of the covariance matrix of the features. These are linear combinations of features that are uncorrelated and sorted in descending order based on the amount of variability they explain, i.e., the first principle component has the highest variability, the second principal component has the next highest variability, and so on. The total number of principal components is the same as the number of original features. The general practice is to pick top q principal components based on some strategy. One of the simplest examples of such strategies is to set a threshold on the percentage of variability explained by the principal components, and then to choose the smallest number of principal components that account for that amount of variability. See Jolliffe's *Principle Component Analysis* book for a comprehensive literature review on PCA and how to decide the number of principal components to keep [36].

1.2.2.2 Autoencoders (AE)

Baldi and Hornik introduced AE, which are a special case of neural networks (NN), in 1989 [37]. The simplest AE consist of an input layer, an output layer, and one hidden layer. The idea is to first compress (encode) the original features to a smaller latent space, i.e. from input layer to hidden layer, and then reconstruct (decode) the original feature using latent space. In training, AE use the original p features as both input and output layers. The values in the hidden layer are combinations of features. These combinations can be

considered as new features or compressed versions of the original features. The main difference between AE and PCA is that AE can produce non-linear combinations of features, depending on the activation function used in its structure. See *Representation Learning: A Review and New Perspectives* by Bengio et al. for more details on AE [38].

1.3 A Closer Look at Random Forests

Random Forest (RF), which was introduced by Brieman [39], is a popular machine learning method. It can be used for feature selection and prediction in both classification and regression problems [11], [12], [40], [41]. To fully understand the underlying algorithm of RF, we need to introduce classification and regression trees (CART), which was developed in the late nineties [42]. CART models consist of nodes that recursively partition the feature space such that observations with similar response variable values fall into the same regions. CART models are simple estimators with low computation cost, but they usually have a high variance. RF has the same bias as CART, but RF significantly reduces the variance by fitting several trees and averaging their predicted values. In order to achieve a small variance, RF generates uncorrelated trees in two steps: using a bootstrap sample from the data for each tree, and considering only a random sample of features at each node of the trees. The bootstrapping step leaves out some observations from the original sample in the construction of each tree. These are called out of bag (OOB) cases and have a role in FI calculation and test set error estimation.

FI is a RF attribute that is used for variable selection and ranking of variables. There are several different methods for computing feature importance; two that we will focus on here are a permutation method and a relative rank based method. Brieman proposed the permutation-based feature importance measure that is implemented in randomForest Package in R [39], [41]. It starts by calculating prediction accuracy (R^2 in regression) of a CART when fitted to its OOB observations. Next, it permutes the values of a feature in OOB cases and fits the CART to them to recalculate prediction accuracy. Large differences in prediction accuracy indicate higher importance for a feature. The relative rank importance, on the other hand, is implemented in sklearn.ensemble module in Python [43]. The definition of the relative rank importance by authors of this module is provided here: "The relative rank (i.e. depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the target variable. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features" [44]. Both of these methods are used to find salient features in our motivational example.

1.4 Motivational Example

As a motivational example, let us evaluate performances of the permutation and relative rank based feature importance measures when applied to a data with unbalanced feature effect magnitudes. Each RF used in this section consists of 100 trees, and one-third of

features are considered as candidates at each split. A sample of size 1000 is simulated using this formula:

$$Response = e^{|x_1|} + 10\,sin(x_2) + \varepsilon \qquad (4)$$

The main features, x_1 and x_2, are generated from a uniform(-5,5) distribution, and the error term has standard normal distribution. Due to the presence of x_1 in the exponential term, the effect of x_2 is masked in most of the response surface except the region where both features are between zero and two. This is evident in Figure 1.1 which plots noise-free signal as a function of the main features.

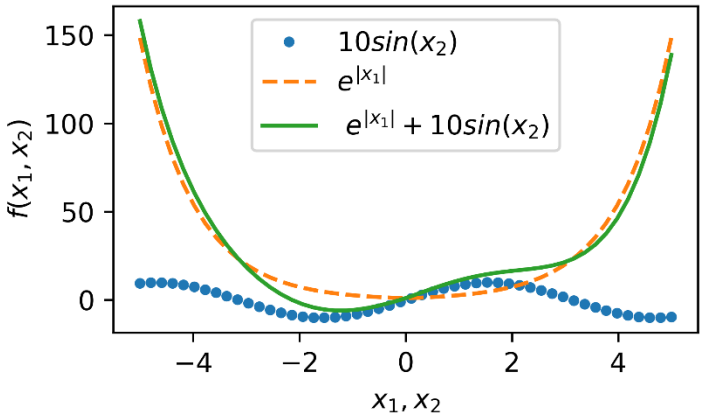

Figure 1.1. Effect of main features on response surface

To be able to evaluate the variable selection performance of these methods, eight independent noise variables ($N_1, \ldots, N_8$) with standard normal distribution are added to the dataset. Ideally, x_1 and x_2 would have a distinguishable FI from the noise variables. Figure 1.2 shows boxplots of FI scores obtained from fitting 50 different RFs (by varying random state argument) to one realization of formula in Equation 4. Note that dots represent outliers

in these boxplots. Features are sorted based on mean FI on x-axis. It is also notable that results from both methods are scaled to sum up to one for the sake of fair comparison.

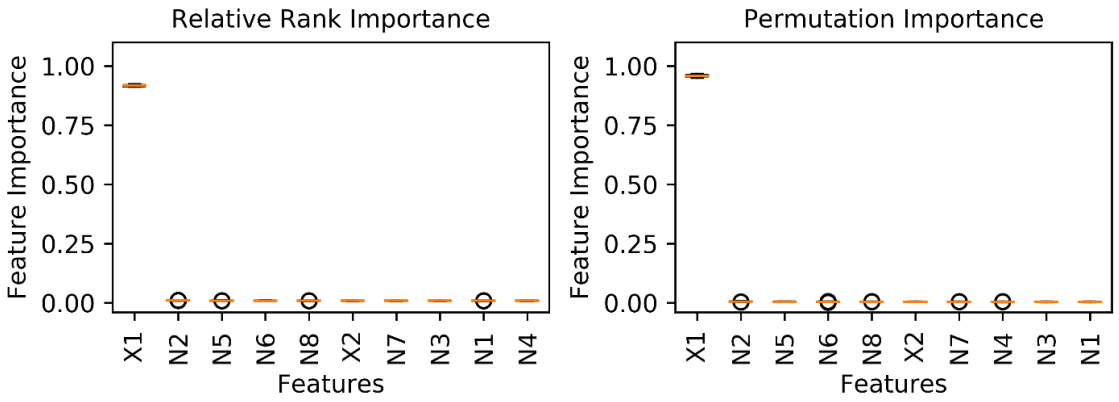

Figure 1.2. Feature importance comparison in 50 random forests fitted to a simulated sample

In both cases, x_2 is picked as the sixth important feature. In this example, RF fails in detecting the effect of a small impact variable in presence of a dominant feature.

One might argue that the results from one realization of this simulation function are not generalizable. To address this point, we generated 50 samples from the formula in Equation 4, and fitted a RF to each sample to calculate FIs. Figure 1.3 displays the results of this simulation. Again, features are sorted based on mean FI on the x-axis.

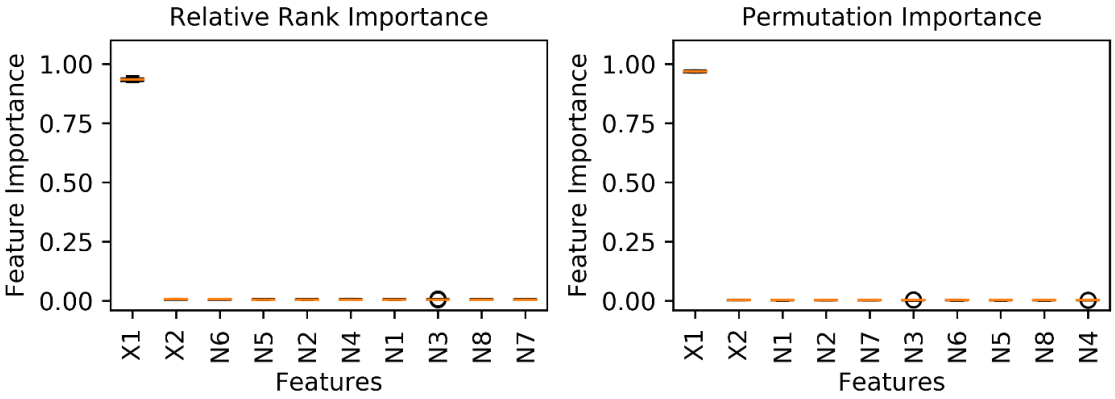

Figure 1.3. Feature importance comparison in 50 simulated samples

This time x_2 is selected as the second most important variable but its FI is indistinguishable from the noise variables. This result is not surprising because these 50 samples provide much larger information than one realization of the simulation function. Intuitively, RF should be able to detect small effects if large enough data is provided.

In both scenarios, x_2 does not stand out as a salient variable among the noise variables. RF's permutation-based and relative rank-based FIs fail in detecting the effect of x_2 in presence of the dominant feature[1]. Following chapters of this dissertation introduce two algorithms to discover locally important features such as x_2.

[1] Due to convenient parallelization, we used the relative rank based importance implemented in RandomForestRegressor from scikit-learn library in Python to perform our calculations in this study.

1.5 Local Variable Selection

Despite the large body of research on global variable selection methods, limited attention has been given to methods of finding locally important variables. Bai et al. propose an approach to find salient features in the neighborhood of a point of interest [45]. Within this neighborhood, variables with near zero partial derivatives are considered insignificant. This same team of authors published another article to extend their original method by relaxing some of its assumptions [46]. Winkel et al. introduce a Bayesian local variable importance measure to find important features in the vicinity of the global optimizer [47]. They report that use of locally important variables results in better estimation of global optima in a smaller number of steps in their simulations.

In this dissertation, we propose two algorithms for localized variable selection: CBFS and LAFI. Both methods aim to find regions where the effects of weaker features can be isolated and measured. CBFS combines RF variable selection with a two-stage clustering method to detect variables that their effect can be detected only in certain regions. LAFI, on the other hand, uses a binary tree approach to split data into bins based on response variable rankings, and implements RF to find important variables in each bin. Larger LAFI is assigned to variables that are selected in more bins. Simulations show great potential for these variable selection methods.

To summarize, this research has three main contributions: 1) LAFI and CBFS look for local important features across the whole response surface while previous studies pick salient features in a particular region of response surface; 2) to the best of our knowledge, this is

the first use of RF for local variable selection; 3) the challenges of using locally fitted RFs for prediction purposes are discussed.

CHAPTER 2: CLUSTERING-BASED FEATURE SELECTION

Random forest (RF) has demonstrated the ability to select globally important variables and model complex data. However, as illustrated in section 1.4, it fails to detect locally salient features in some cases. This chapter introduces clustering-based feature selection (CBFS) which aims to address this issue. The general strategy of CBFS is to segment data into clusters and use RF to select features within each cluster. We propose average and intersection methods to aggregate the variable selection results from these clusters.

2.1 CBFS Algorithm

We use a toy example in this section to illustrate the steps of CBFS. Let us consider an example with two input features, x_1 and x_2, and a non-linear response surface. Figure 2.1 presents the heat map of this response surface. The response variable has larger values in red regions and smaller values in white and yellow regions.

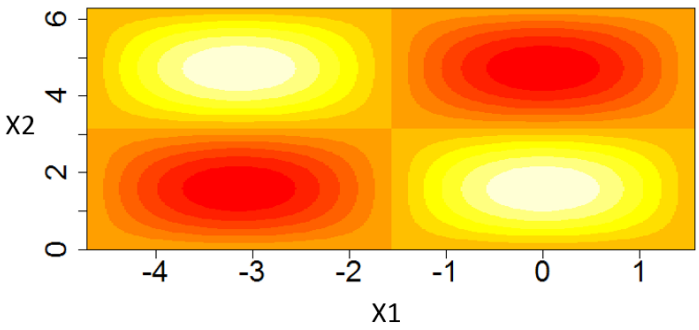

Figure 2.1. Heat map of the toy example response surface

CBFS starts with splitting data into same-sized groups based on response value rankings. This step aims to find important features in different regions of the response surface. The number of groups is a user-defined parameter. Note that a very large number of groups results in a small number of observations in each group. This can destabilize the variable selection results in the next steps of CBFS. On the other hand, choosing a very small number of groups drives CBFS towards global variable selection instead of local variable selection. In our example, we decided to divide the data into two groups as shown in Figure 2.2.

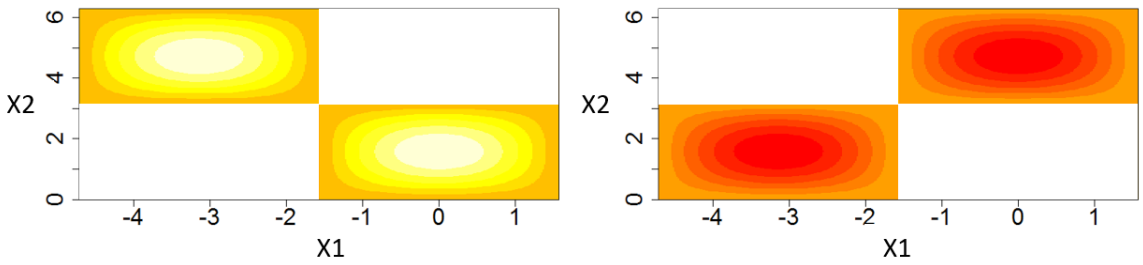

Figure 2.1. Dividing data in low (left plot) and high (right plot) response value groups

Then, within each response group, features are used to form clusters, and a RF is fitted to each cluster. Note that we scale each feature to have zero mean and unit variance before clustering. K-means clustering is used to make the clusters, and the Silhouette score is used to determine the optimal number of clusters [48], [49]. Note that this can result in different number of feature-based clusters in different response groups. We use only features in making these clusters to be able to assign new observations to these clusters (based on proximity of features or other techniques discussed later) and use the corresponding RF for the assigned cluster to make local predictions. The results and challenges of local

prediction are discussed in the next chapter. This chapter focuses on the local variable selection aspect of CBFS. Figure 2.3 shows the feature-based clusters in the toy example.

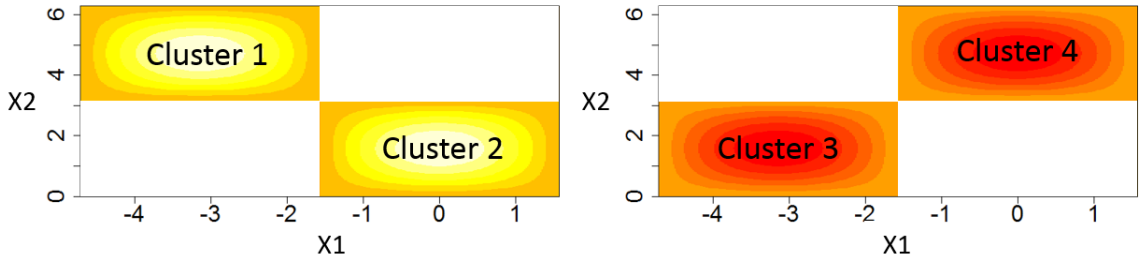

Figure 2.3. Feature-based clusters of data for the toy example

Finally, Feature Importance (FI) scores are obtained from the fitted RFs in each cluster. We propose average and intersection methods to aggregate the variable selection results from these clusters. The average method ranks variables based on their weighted mean FI, where cluster sizes are used as weights, obtained from local RFs fitted in feature-based clusters. This assigns larger weights to FI scores obtained from bigger clusters. The average method steps are summarized in Algorithm 2.1.

Algorithm 2.1: Average Method
Input: Number of response bins M; Maximum number of feature-based clusters in each response bin K; Number of features to output N.
Output: Top N important features.

1. Split data into M same-sized bins based on response value rankings.
2. Scale each feature to have zero mean and unit variance.
3. Within each response bin, use only scaled features to make clusters and find optimal number of clusters in range 2 to K using Silhouette score. This may result in different number of clusters in each response bin.
4. Fit separate local RFs to data points in clusters in step 3 and obtain FI from each model.
5. Calculate weighted average FI for each feature. Use cluster sizes as weights.
6. Rank features based on their mean FI.

Table 2.1 contains the hypothetical FI scores obtained from the four RFs fitted to the clusters in the toy example. We assume feature-based clusters are same-sized in this example. The weighted average column in this table shows the weighted mean FI score for each variable. The average method ranks variables based on this column.

Table 2.1. Feature importance scores for toy example

	RF in Cluster 1	**RF in Cluster 2**	**RF in Cluster 3**	**RF in Cluster 4**	**Weighted Average**
x_1	0.50	0.30	0.70	0.30	0.4500
x_2	0.30	0.40	0.10	0.50	0.3250
Noise 1	0.05	0.20	0.15	0.03	0.1075
Noise 2	0.15	0.10	0.05	0.17	0.1175

The intersection method uses ranking of variables based on their FI scores from the fitted local RFs. It iteratively selects variables that are in common among top-ranked variables of these models. The steps of this method are shown in Algorithm 2.2.

Algorithm 2.2: Intersection Method
Input: Number of response bins M; Maximum number of feature-based clusters in each response bin K; Number of features to output N.
Output: Top N important features.

1. Split data into M same-sized bins based on response value rankings.
2. Scale each feature to have zero mean and unit variance.
3. Within each response bin, use only scaled features to make clusters and find optimal number of clusters in range 2 to K using Silhouette score. This may result in different number of clusters in each response bin.
4. Fit separate local RFs to data points in clusters in step 3 and obtain ranking of features based on their relative FI for each model.
5. Iteratively, select variables that are in common among top ranked variables in step 4.
6. In case of a tie, use weighted mean FI to break the tie. Weights are cluster sizes.

Table 2.2 shows the ranking of variables in each of the four fitted RFs. Let us start with the rank one variables, i.e. x_1, x_2, x_1, and x_2 in clusters one through four, respectively. Obviously, the intersection of these features is an empty set. Next, we consider sets of top two variables from each RF. Now, x_1 is picked because it is either the first or the second-ranked variable in all RFs. By looking at the top 3 ranked features in all clusters, we can pick x_2 as the next in common variable. Finally, Noise 1 and Noise 2 variables are selected at the same time in the last round of iteration. We use the average FI score from Table 2.1 to break the tie between these two variables. In conclusion, the iteration method ranks the toy example features in this order: x_1, x_2, Noise 2, and Noise 1.

Table 2.2. Ranking of variables in the toy example

Rank	RF in Cluster 1	RF in Cluster 2	RF in Cluster 3	RF in Cluster 4
1	x_1	x_2	x_1	x_2
2	x_2	x_1	Noise 1	x_1
3	Noise 2	Noise 1	x_2	Noise 1
4	Noise 1	Noise 2	Noise 2	Noise 2

2.2 Results

This section presents real-world and generated datasets to compare the variable selection performance of regular RF and CBFS. We use simulation functions from section 2.2 to generate datasets. Regarding real-world datasets, we use wine quality and bike sharing datasets from University of California Irvine machine learning repository [50]–[52].

In each dataset, CBFS splits data points into 5 same-sized bins based on response value rankings. Top selected variables from CBFS are compared to those picked by a regular RF fitted to the whole data. The KMeans and silhouette_score functions with their default parameters from the scikit-learn library in Python are used to form feature-based clusters and determine the optimal number of clusters, respectively [53], [54]. In each response bin, we limit the maximum number of feature-based clusters considered by silhouette score to eight to reduce computation time and avoid forming very small clusters. The RandomForestRegressor function from the same Python library is used to fit RFs [55]. Each RF in CBFS considers one-third of the features at each node, and uses 100 trees with maximum depth of ten. The rest of the RF parameters are set to RandomForestRegressor's default values. Finally, for the sake of fairness in comparing CBFS and regular RF, the number of trees in regular RF is set to the same total number of trees as is used by CBFS. For example, if four RFs with 100 trees each are used in CBFS, then a regular RF with 400 trees is used for comparison. All the other parameters of the regular RF and the RFs in CBFS are identical.

We use the above-mentioned setting as our default for all experiments in this chapter. However, depending on experiment outcomes, we may repeat some experiments with another set of parameters in some cases. The results for default and updated settings are presented in such cases.

2.2.1 Simulations

2.2.1.1 Simulation 1

We start with the motivational example introduced in section 1.4. Here is the simulation function:

$$Response = e^{|x_1|} + 10\,sin(x_2) + \varepsilon \qquad (4)$$

10000 data points are generated with x_1 and x_2 independently following a uniform(-5,5) distribution, and the error term having a standard normal distribution. Ten independent noise variables, each with a standard normal distribution, are added to the data set. Ideally, x_1 and x_2 would be picked over the noise variables.

Table 2.3 compares the variable selection performance of CBFS and regular RF in 50 replication of this simulation. It is notable that each replicate of this experiment takes less than a minute to run on 20 CPUs with 2.1 GHz processing speed. All RF and clustering functions are run in parallel using these CPUs.

Table 2.3. Frequency of selecting $x_1$and x_2 among top N = two variables in 50 replication of simulation 1 with 10 noise variables

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x_1	50	50	50
x_2	50	25	4

CBFS outperforms regular RF in this example. This is hypothesized to be due to the masking effect of x_1 in the exponential term as described in section 1.4. In addition, the results indicate the superiority of average method over intersection method in this example.

As an extension, we repeat this simulation with 50 noise variables instead of ten. Table 2.4 presents the results of this simulation. As expected, adding more noise variables makes it harder to detect the true signal variables. However, CBFS with the average method still correctly selects x_1 and x_2 as the top features in every replicate. The addition of noise variables has negatively impacted the performance of regular RF and CBFS with intersection method.

Table 2.4. Frequency of selecting x_1 and x_2 among top N = two variables in 50 replication of simulation 1 with 50 noise variables

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x_1	50	50	50
x_2	50	13	1

2.2.1.2 Simulation 2

A highly nonlinear response surface with interaction terms is simulated in this section to compare CBFS and regular RF. A sample of 10000 data points is generated using the formula in the equation below:

$$Response = x + y + z^2 + w + q + sin(xy^2) + e^{zwq^2} + \varepsilon \qquad (5)$$

The features and error term each independently follow the standard normal distribution. In addition to $x, y, z, w,$ and $q, 50$ independent noise variables with the standard normal distribution are added to the data set. Table 2.5 shows the variable selection results from 50 replications of this simulation.

Table 2.5. Frequency of selecting main features among top N = five variables in 50 replication of Simulation 2

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x	50	36	0
y	50	21	1
w	50	36	37
z	10	20	40
q	50	38	50

Regular RF fails to detect x and y in presence of the features in the exponential term. However, CBFS does not miss these less influential variables in most replicates. Note that while $x, y, w,$ and q are selected among top five features in every iteration of CBFS with average method, z is only selected in ten iterations. To investigate the reason for this result, we present the frequency of selecting main features among top seven variables in Table 2.6. This table shows that feature z is selected in 29 of iterations. In overall, CBFS with average method seems to be the best variable selection method in this simulation.

Table 2.6. Frequency of selecting main features among top N = seven variables in 50 replication of Simulation 2

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x	50	34	2
y	50	22	4
w	50	38	43
z	29	27	44
q	50	40	50

2.2.1.3 Simulation 3

The simulation function in Equation 6 presents a low signal-to-noise ratio response surface with underlying interaction terms. Note that presence of 10ε in this formula makes this response surface very noisy. 10000 data points are generated using this equation.

$$Response = xy + \text{wzq} + 10\varepsilon \qquad (6)$$

The main features and error term all independently follow the standard normal distribution. 50 independent noise variables with standard normal distribution are added to the data set.

Table 2.7 shows that regular RF does a better job than CBFS with average method in selecting x and y but CBFS detects z and w more often. These methods show almost similar performances for q.

Table 2.7. Frequency of selecting main features among top N = five variables in 50 replication of Simulation 3

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x	6	8	18
y	9	7	13
w	9	3	8
z	10	7	6
q	7	4	7

To explore this result in more depth, we repeat this experiment without multiplying the error term by ten. This simplifies the variable selection task by boosting signal-to-noise ratio. Table below shows the new variable selection results. This time both regular RF and CBFS with average method are able to detect features in interaction terms in every iteration.

Table 2.8. Frequency of selecting main features among top N = five variables in 50 replication of Simulation 3 with improved signal-to-noise ratio

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
x	50	23	50
y	50	19	50
w	50	19	50
z	50	16	50
q	50	15	50

2.2.2 Wine Quality Data

The wine quality data has been used in several data mining studies [50], [56]. The response variable in this dataset is wine quality score, which ranges from one to ten, based on physicochemical tests. It contains about 4900 instances with 11 features: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. Five independent noise variables with standard normal distribution are added to the dataset to compare variable selection performances of CBFS and regular RF. We replicate this experiment 50 times by changing the random_state argument of RandomForestRegressor function which is used in regular RFs and RFs in CBFS [55]. Table 2.9 compares the variable selection results of the three methods.

Table 2.9. Frequency of selecting main features among top N = 11 variables in 50 replications for wine quality data

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Fixed acidity	42	33	50
Volatile acidity	50	50	50
Citric acid	1	24	50
Residual sugar	41	49	50
Chlorides	23	30	50
Free sulfur dioxide	50	47	50
Total sulfur dioxide	46	47	50
Density	46	43	50
pH	7	39	50
Sulphates	48	48	50
Alcohol	50	26	50

Regular RF outperforms the other methods. We hypothesize that these results are due to the formation of small clusters in the CBFS clustering step. Having a small number of observations in clusters can result in overfitting of RFs and selection of noise variable. In an attempt to evaluate our hypothesis, we repeat this experiment with a few tweaks in our settings. We fix both the number of response bins and the number of clusters in each response bin to two (M and K parameters in the CBFS algorithm). We also reduce the max depth of RFs from ten to five to avoid overfitting. Table 2.10 shows that these tweaks have improved the performance of CBFS with average and intersection methods.

Table 2.10. Frequency of selecting main features among top N = 11 variables in 50 replications for wine quality data with updated experiment settings

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Fixed acidity	50	47	50
Volatile acidity	50	50	50
Citric acid	44	45	50
Residual sugar	50	50	50
Chlorides	50	40	50
Free sulfur dioxide	50	50	50
Total sulfur dioxide	50	28	50
Density	50	50	50
pH	50	45	50
Sulphates	27	44	50
Alcohol	50	50	50

2.2.3 Bike Sharing Data

The bike sharing is a regression dataset where the response variables is the number of rented bikes for casual and registered bikers in every hour of 2011 and 2012 in Capital bikeshare system. This dataset contains 17389 instances and 12 features: season, year, month, hour, holiday (binary variable), weekday, workingday (binary variable), weather situation (categorical with 4 categories), temperature, feeling temperature, humidity, and windspeed. Five independent noise variables with standard normal distribution are added to the dataset to compare variable selection performances of CBFS and regular RF.

Intuitively, we think casual bikers and registered bikers might have different sets of important features. Thus, we replicate this experiment 50 times for each group by changing the random_state argument of RandomForestRegressor function that is used in regular RFs and RFs in CBFS.

2.2.3.1 Registered Bikers Group

Let us start with the registered bikers group. Table 2.11 shows the variable selection results for the three methods.

Table 2.11. Frequency of selecting main features among top N = 12 variables in 50 replications for registered bikers

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Season	0	0	50
year	50	36	50
Month	0	47	50
Hour	50	50	50
Holiday	0	0	0
Weekday	50	17	50
Working day	23	0	50
Weather situation	0	0	50
Temperature	50	50	50
Feeling Temperature	50	50	50
Humidity	50	50	50
Wind speed	27	50	25

This table shows that CBFS methods do not select season and weather situation among top features. One possible explanation is that these features are highly correlated with some other features such as temperature and humidity. Or, similar to wine quality data, these results are due to the formation of small clusters in data. Let us repeat this experiment by fixing both the number of response bins and the number of clusters in each response bin to two (M and K parameters in the CBFS algorithm). This should result in larger clusters. In addition, we reduce the max depth of RFs from ten to five to avoid overfitting. Table 2.12 shows the variable selection results for this new setting. The results from the CBFS with average method is in agreement with regular RF except for the season feature.

Table 2.12. Frequency of selecting main features among top N = 12 variables in 50 replications for registered bikers with updated experiment settings

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Season	1	0	50
year	50	50	50
Month	43	27	50
Hour	50	50	50
Holiday	0	0	0
Weekday	50	50	50
Working day	50	50	50
Weather situation	50	2	50
Temperature	50	50	50
Feeling Temperature	50	50	50
Humidity	50	50	50
Wind speed	50	48	50

2.2.3.2 Casual Bikers Groups

Now, let us redo this experiment for casual bikers with the default settings described at the beginning of section 2.2. Table 2.13 presents the variable selection results for all candidate methods.

Table 2.13. Frequency of selecting main features among top N = 12 variables in 50 replications for casual bikers

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Season	0	0	50
year	0	0	50
Month	0	50	50
Hour	50	50	50
Holiday	0	0	0
Weekday	50	50	50
Working day	50	0	50
Weather situation	0	0	34
Temperature	50	50	50
Feeling Temperature	50	50	50
Humidity	50	50	50
Wind speed	50	50	50

Similar to our first attempt at registered bikers, CBFS methods do not pick some of the main features. We redo this experiment with the same limitations on the number of response bins, clusters, and RF depth that we used in our second experiment for registered bikers. Table 2.14 shows the improvement in CBFS results.

Table 2.14. Frequency of selecting main features among top N = 12 variables in 50 replications for casual bikers with updated experiment settings

	CBFS (Average Method)	**CBFS (Intersection Method)**	**Regular RF**
Season	50	36	50
year	50	28	50
Month	50	50	50
Hour	50	50	50
Holiday	0	0	50
Weekday	50	50	50
Working day	50	50	50
Weather situation	50	0	50
Temperature	50	50	50
Feeling Temperature	50	50	50
Humidity	50	50	50
Wind speed	50	46	50

2.3 Summary

In some cases, such as the motivational example in chapter 1, RF does not detect the effect of locally important variables in presence of dominant features. This section introduces CBFS with average and intersection methods as variable selection techniques which aim to address this issue. CBFS uses binning and clustering to segment data into homogenous regions where the effect of variables can be locally isolated and detected. The variable

selection performance of CBFS is compared to regular RF in simulations provided in this chapter.

Simulation 1 presents a response surface which is dominated by an exponential term. The main challenge for CBFS and regular RF is to detect the effect of the less significant feature which is masked by this exponential term. Results show that the CBFS with average and intersection methods outperform regular RF. It is hypothesized that the success of CBFS is a result of the clustering step in its algorithm. Some clusters are probably formed in regions where the value of the exponential term is small, and this makes it possible to isolate and detect the effect of the locally important feature in these regions.

A highly nonlinear data with interaction terms is used in simulation 2. CBFS with average method clearly exceeds other candidates in this case. The comparison between CBFS with intersection method and regular RF is challenging. CBFS with intersection method picks locally salient features more often than regular RF. However, regular RF detects globally important features in more iterations.

Simulation 3 is meant to illustrate how the 3 methods perform when applied to low signal-to-noise ratio data. CBFS with intersection method is found to be the inferior candidate. Regular RF does a slightly better job than CBFS with average method. One possible explanation for this result is that RFs in CBFS are exposed to a smaller number of observations in each cluster, in comparison to regular RF which is fitted to the whole data, and more data points are needed for RFs to learn patterns in low signal-to-noise ratio data.

Wine quality and bike sharing datasets are used as real datasets to explore the application of our methods. In both datasets, regular RF is found to be more successful. We hypothesize that these results are due to the formation of small clusters in the CBFS clustering step. Having a small number of observations in clusters can result in overfitting of RFs and selection of noise variable. In an attempt to evaluate our hypothesis, we repeated this experiment with a few tweaks in our settings. A smaller number of bins and clusters are used to avoid forming small clusters. This improved the performance of CBFS. In general, CBFS requires a large enough sample size in each cluster to be able to select important features effectively.

These results show the potential of CBFS in detecting features in a variety of settings. It is notable that our results are only generalizable to simulation settings included in this chapter. Further research is required before inferring our findings to broader population of data structures.

CHAPTER 3: LOCAL PREDICTION – AN EXTENSION TO CBFS

3.1 Introduction

As described in the previous chapter, features are used to form clusters in clustering-based feature selection (CBFS), and one random forest (RF) is fitted to each feature-based cluster. We only use features in making these clusters so that we will be able to assign unseen points to these clusters and use the corresponding RF to make local predictions. This chapter explores two local prediction methods based on the same idea: local prediction with supervised clustering (LPSC) and local prediction with unsupervised clustering (LPUC). LPSC uses cluster assignments of training set data as labels and assigns new points to clusters using a RF classifier. The LPSC steps are summarized in Algorithm 3.1.

Algorithm 3.1: Local Prediction with Supervised Clustering (LPSC)
Input: Number of response bins M; Maximum number of feature-based clusters in each response bin K; Train and Test sets.
Output: Local predictions for Test set data points.

1. Split Train set into M same-sized bins based on response value rankings.
2. Scale each feature in Train set to have zero mean and unit variance. Store mean and standard deviations of each feature, i.e. $\overline{X_j}$ and S_j for $j \in \{1, \dots, p\}$ where p is the total number of features.
3. Within each response bin, use only scaled features to make clusters and find optimal number of clusters in range 2 to K using Silhouette score.
4. Fit separate local RF regressors to data points in each cluster.
5. Use clustering assignments of data points in step 3 as labels to make a RF classifier.
6. Scale features in Test set by subtracting the corresponding $\overline{X_j}$ and dividing by corresponding S_j from step 2.
7. Predict cluster assignment of Test set data points using the RF classifier in step 5.
8. Use the corresponding RF regressor from step 4 to predict the response value of Test set cases.

In LPUC, new points are assigned to clusters based on proximity to cluster centers in feature space. Algorithm 3.2 outlines the steps of LPUC.

Algorithm 3.2: Local Prediction with Unsupervised Clustering (LPUC)
Input: Number of response bins M; Maximum number of feature-based clusters in each response bin K; Train and Test sets.
Output: Local predictions for Test set data points.

1. Split Train set into M same-sized bins based on response value rankings.
2. Scale each feature in Train set to have zero mean and unit variance. Store mean and standard deviations of each feature, i.e. $\bar{X}_j$ and S_j for $j \in \{1, \dots, p\}$ where p is the total number of features.
3. Within each response bin, use only scaled features to make clusters and find optimal number of clusters in range 2 to K using Silhouette score.
4. Fit separate local RF regressors to data points in each cluster.
5. Scale features in Test set by subtracting the corresponding $\bar{X}_j$ and dividing by corresponding S_j from step 2.
6. Assign Test set data points to the clusters in step 3 based on Euclidean proximity to cluster centers in feature space.
7. Use the corresponding RF regressor from step 4 to predict the response value of Test set cases.

Local prediction performances of these methods are compared to regular RF in generated and real datasets in this chapter.

3.2 Results

This section presents real-world and generated datasets to compare the prediction performance of regular RF, LPUC, and LPSC. We use simulation functions from section 2.2 for our generated datasets. Regarding real-world datasets, we use wine quality and bike sharing datasets from University of California Irvine machine learning repository [50]–[52].

For each dataset, 75% of the data points are randomly selected for the train set and the remaining 25% make the test set. Each experiment is replicated 50 times. LPSC and LPUC split data into 5 same-sized bins based on response value rankings. KMeans and silhouette_score functions with their default parameters from the scikit-learn library in Python are used to form feature-based clusters and determine the optimal number of clusters, respectively [53], [54]. In each response bin, we limit the maximum number of feature-based clusters considered by silhouette score to eight to reduce computation time and avoid forming very small clusters. RandomForestRegressor and RandomForestClassifier functions from the same Python library are used to fit RF regressors and classifiers, respectively [55], [57]. For the number of features to consider at each node, RF regressors use one-third of features and RF classifiers use the square root of total number of features. All RFs consist of 100 trees with a maximum depth of ten. The rest of RF parameters are set to scikit-learn default values. Finally, for the sake of fairness in comparing regular RF, LPUC, and LPSC, the number of trees in regular RF is set to the same total number of trees as is used by RF regressors in LPUC or LPSC. For example, if four RF regressors with 100 trees each are used in LPUC, then a regular RF with 400 trees is used for comparison. All the other parameters of regular RF and RF regressors in LPUC and LPSC are identical.

We use the above-mentioned setting as our default for all experiments in this chapter. However, depending on experiment outcomes, we may repeat some experiments with another set of parameters in some cases. The results for default and updated settings are presented in such cases.

3.2.1 Simulations

3.2.1.1 Simulation 1

We start with the motivational example introduced in section 1.4. Here is the simulation function:

$$Response = e^{|x_1|} + 10\,sin(x_2) + \varepsilon \qquad (4)$$

10000 data points are generated with x_1 and x_2 independently following a uniform(-5,5) distribution, and the error term having a standard normal distribution. Ten independent noise variables, each with a standard normal distribution, are added to the data set. Figure 3.1 compares the prediction performance of regular RF, LPUC, and LPSC based on mean squared error (MSE). Note that LPUC and LPSC perform the same on the train set because these methods only differ in how they assign unseen points (test set data) to feature-based clusters.

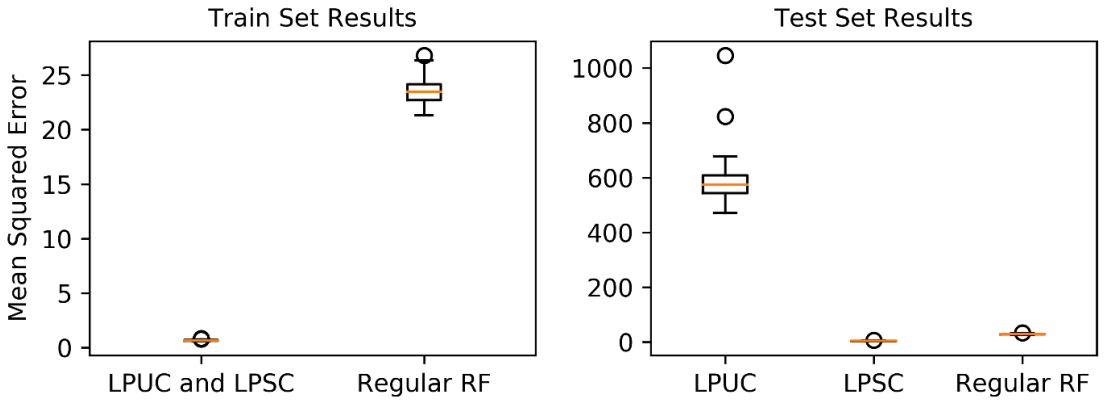

Figure 3.1. Prediction performance comparison between local methods and regular RF in Simulation 1 with 10 noise variables

Local prediction methods outperform regular RF in the train set. This is not surprising since local RF regressors are fitted to homogenous sets of data resulting from the initial binning step, based on response value rankings, in local methods' algorithms. In the test set, LPSC achieves the lowest MSE. This is more evident in Figure 3.2 which shows the paired difference in MSE of LPSC and regular RF in all replicates of this simulation. The test set MSE for LPUC is much larger than other methods. This is probably due to poor assignment of test set cases to feature-based clusters. For example, a large error occurs if a test set case with small response value is predicted by a local RF for a cluster of large response value cases.

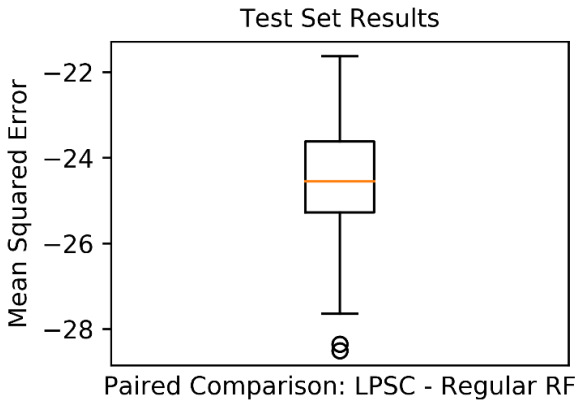

Figure 3.2. Paired comparison of test set MSE for LPSC and regular RF in Simulation 1 with 10 noise variables

We repeat this simulation with 50 noise variables to see the effect of extra noise variables on prediction performance of these methods. Figure 3.3 shows that MSEs are increased but the ranking of methods based on their performance is the same as simulation with 10 noise variables.

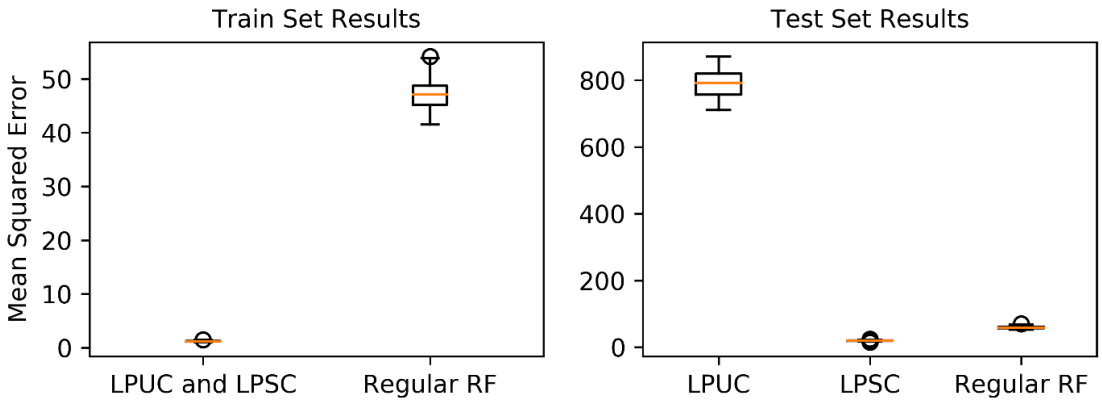

Figure 3.3. Prediction performance comparison between local methods and regular RF in Simulation 1 with 50 noise variables

Finally, Figure 3.4 provides a closer look at MSE differences of regular RF and LPSC in the test set. LPSC outperforms other methods in this simulation.

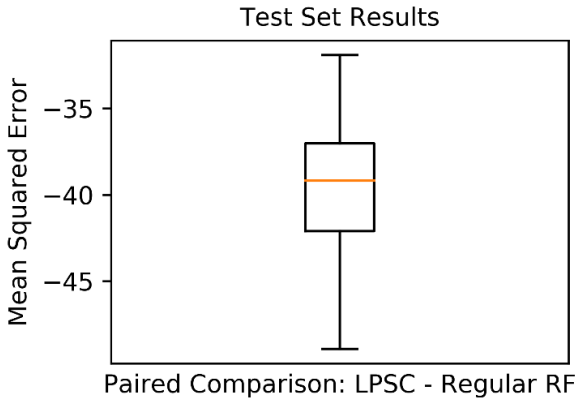

Figure 3.4. Paired comparison of test set MSE for LPSC and regular RF in Simulation 1 with 50 noise variables

3.2.1.2 Simulation 2

A highly nonlinear response surface with interaction terms is simulated in this section to compare local prediction methods and regular RF. A sample of 10000 data points is generated using the formula in the equation below:

$$Response = x + y + z^2 + w + q + sin(xy^2) + e^{zwq^2} + \varepsilon \qquad (5)$$

The features and error term each independently follow the standard normal distribution. In addition to $x, y, z, w,$ and $q,$ 50 independent noise variables with the standard normal distribution are added to the data set. Figure 3.5 compares the prediction accuracy of the three methods.

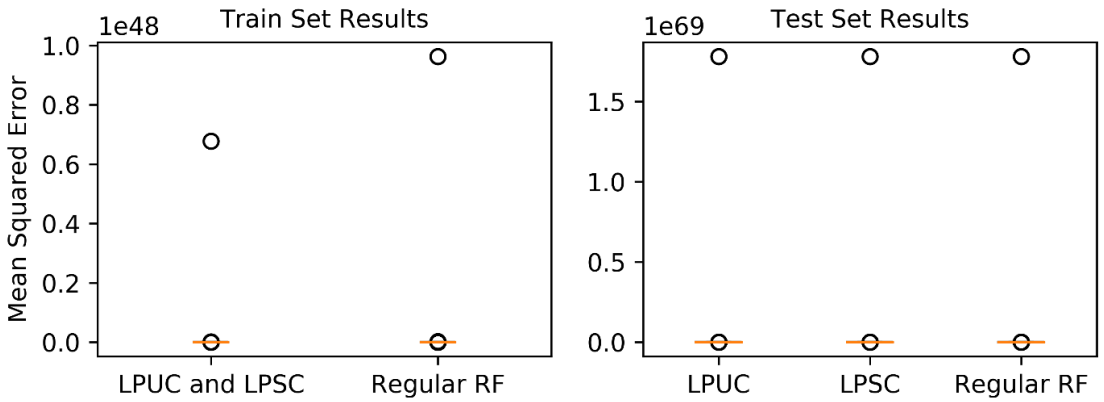

Figure 3.5. Prediction performance comparison between local methods and regular RF in Simulation 2

Note that the ranges of MSE values are substantially different in test and train sets. This level of variability in these results is probably due to the features in the exponential term in Equation 5. Figure 3.6 shows the results on a log scale.

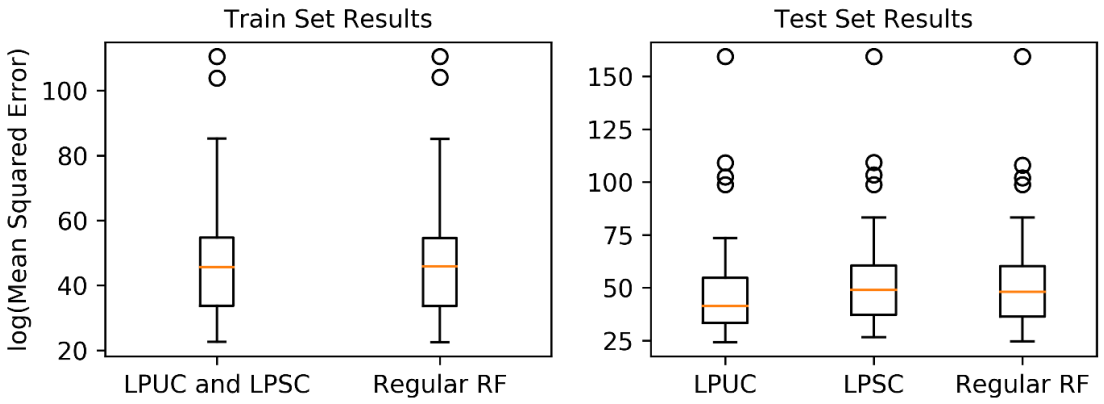

Figure 3.6. Prediction performance comparison between local methods and regular RF in Simulation 2 on a log scale

Table 3.1 provides a numerical summary of MSE values for each method on original scale. In test set, MSE values for LPUC has a lower minimum, first quartile (Q1), median, and third quartile (Q3) than regular RF. The performances of all methods are relatively close in the train set. That being said, making any comparisons based on MSE is challenging for this highly non-linear simulation function.

Table 3.1. Numerical summary of MSE results in Simulation 2

	Train Set MSE		**Test Set MSE**		
	LPUC and LPSC	**Regular RF**	**LPUC**	**LPSC**	**Regular RF**
Min	6.64E+09	5.92E+09	3.51E+10	4.04E+11	5.61E+10
Q1	4.25E+14	4.06E+14	3.06E+14	2.21E+16	5.96E+15
Median	8.21E+19	9.86E+19	1.06E+18	2.04E+21	8.14E+20
Q3	6.02E+23	5.09E+23	5.73E+23	2.09E+26	1.46E+26
Max	9.99E+47	9.63E+47	1.78E+69	1.78E+69	1.78E+69

3.2.1.3 Simulation 3

The simulation function in Equation 6 presents a low signal-to-noise ratio response surface with underlying interaction terms. Note that the presence of 10ε in this formula makes this response surface very noisy. 10000 data points are generated using this equation.

$$Response = xy + \text{wzq} + 10\varepsilon \tag{6}$$

The main features and error term all independently follow the standard normal distribution. 50 independent noise variables with standard normal distribution are added to the data set. Boxplots shown in Figure 3.7 indicate the superiority of regular RF in the test set. Note that the MSE for local methods is significantly different between test and train sets.

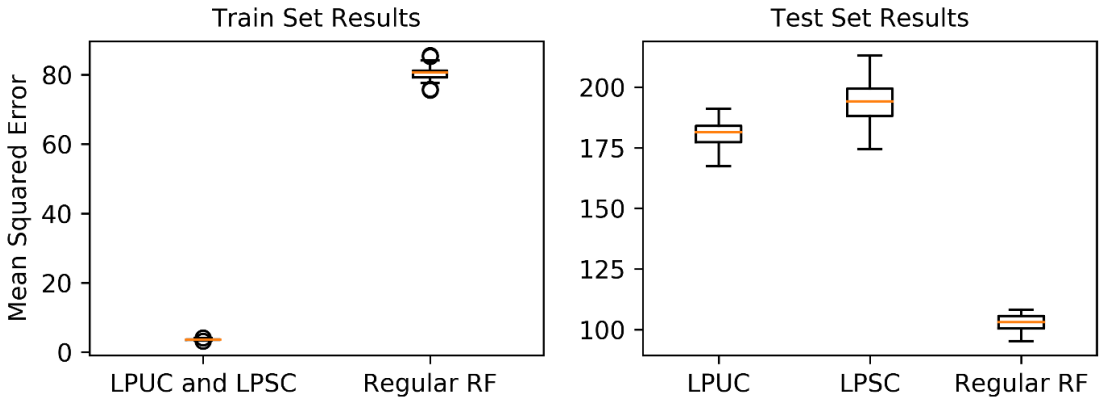

Figure 3.7. Prediction performance comparison between local methods and regular RF in Simulation 3

Local methods have a much smaller MSE in the train set in comparison to the test set. This could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters. Let us repeat this experiment by fixing both the number of response bins and the number of feature-based clusters within each response bin to two (M

and K parameters in the CBFS algorithm). This limits the total number of feature-based clusters to four. We hope this reduces the test set error in two ways: (1) by reducing the error due to a poor assignment of test set cases to feature-based clusters; (2) by resulting in a larger number of data points in each cluster which can decrease the chances of overfitting. Finally, we limit the maximum depth of RFs to five in a further attempt to reduce the probability of overfitting. Figure 3.8 shows that LPSC achieves a lower MSE than regular RF in both train and test sets with this new experiment setting.

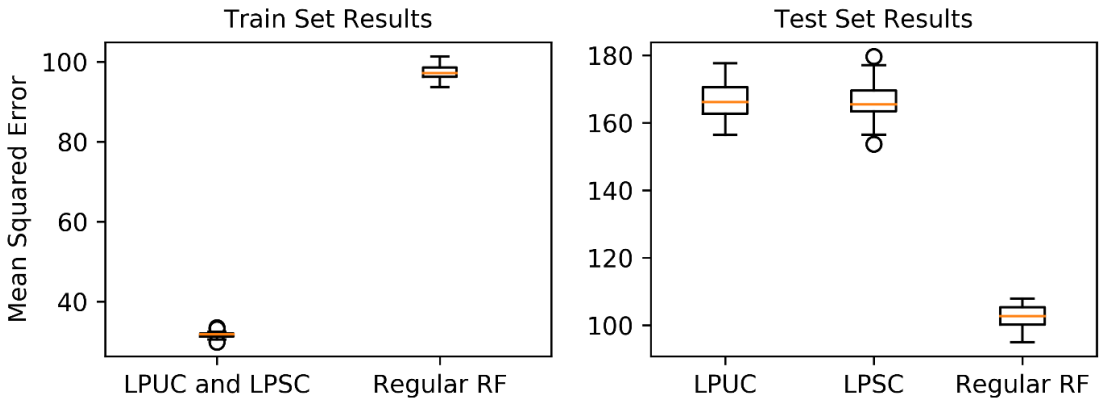

Figure 3.8. Prediction performance comparison between local methods and regular RF in Simulation 3 with updated settings

Figure 3.9 shows the paired comparison between MSE values of LPSC and regular RF on test sets.

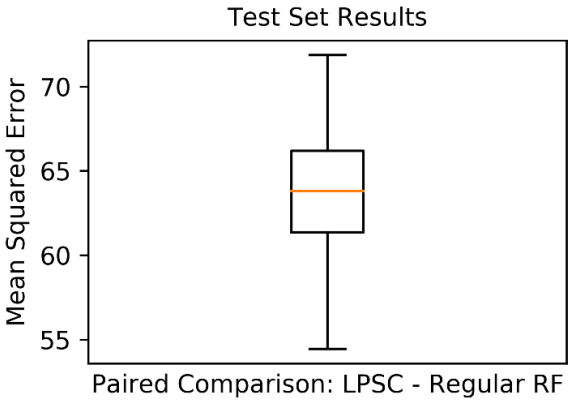

Figure 3.9. Paired comparison of test set MSE for LPSC and regular RF in Simulation 3 with updated settings

3.2.2 Wine Quality Data

The wine quality data has been used in several data mining studies [50], [56]. The response variable in this dataset is wine quality score, which ranges from one to ten, based on physicochemical tests. It contains about 4900 instances with 11 features: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. Five independent noise variables with standard normal distribution are added to the dataset. 75% of the data points are randomly selected for the train set and the remaining 25% make the test set. We replicate this experiment 50 times by changing the random seed which results in different train and test sets at each replicate. Figure 3.10 compares the prediction accuracy of the three methods.

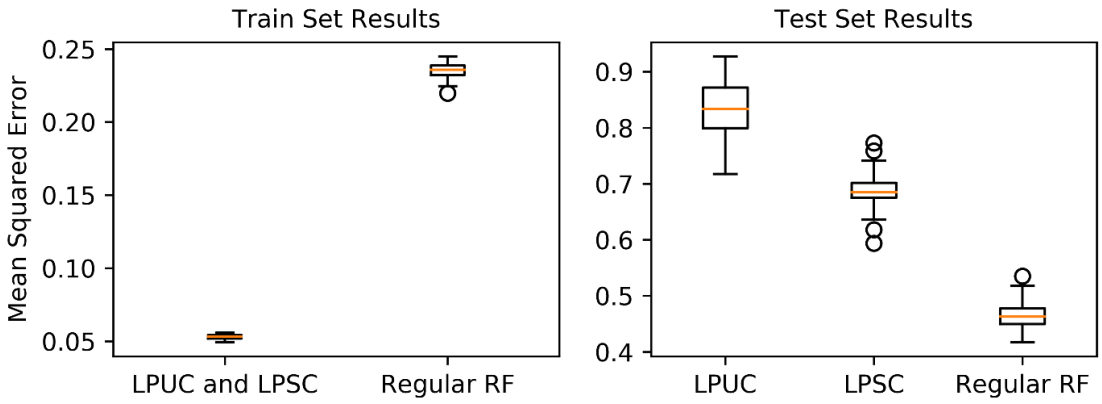

Figure 3.10. Prediction performance comparison between local methods and regular RF when applied to the wine quality data

LPUC and LPSC achieve a lower MSE in train sets but a higher MSE in test sets in comparison to regular RF. This could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters.

Let us repeat this experiment by fixing both the number of response bins and the number of feature-based clusters within each response bin to two (M and K parameters in the CBFS algorithm). This limits the total number of feature-based clusters to four. We hope this reduces the error due to poor cluster assignments of test set cases. Furthermore, we limit the maximum depth of RFs to five in an attempt to reduce the probability of overfitting. Figure 3.11 shows an increase in MSE values for all methods in train sets. This is expected as we decrease the maximum depth of RFs. However, the MSE values in test sets are very similar to values in Figure 3.10. Again, regular RF outperforms local methods in test sets.

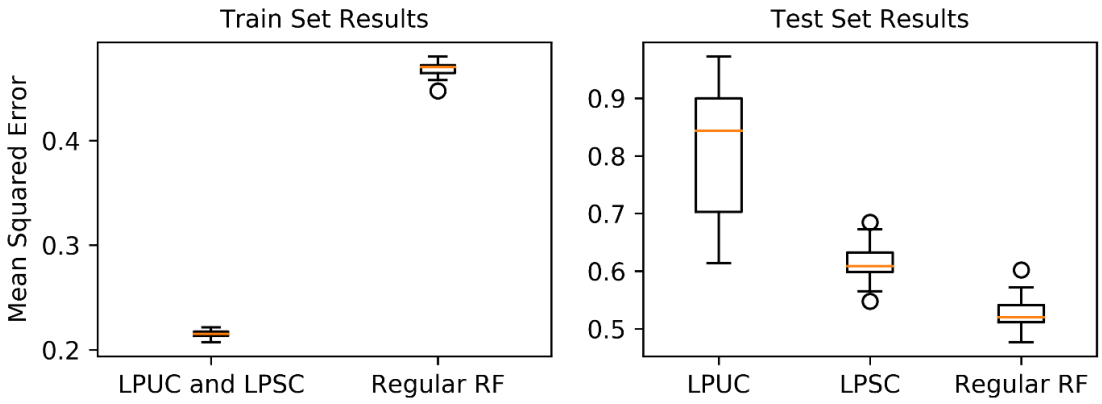

Figure 3.11. Prediction performance comparison between local methods and regular RF when applied to the wine quality data with updated experiment settings

3.2.3 Bike Sharing Data

The bike sharing is a regression dataset where the response variables is the number of rented bikes for casual and registered bikers in every hour of 2011 and 2012 in Capital bikeshare system. This dataset contains 17389 instances and 12 features: season, year, month, hour, holiday (binary variable), weekday, workingday (binary variable), weather situation (categorical with 4 categories), temperature, feeling temperature, humidity, and windspeed. Five independent noise variables with standard normal distribution are added to the dataset. We compare the prediction accuracy of local methods and regular RF for registered and casual bikers. We replicate this experiment 50 times by changing the random seed which results in different train and test sets at each replicate. We do separate experiments for casual and registered bikers in this section.

3.2.3.1 Registered Bikers Group

Let us start with the registered bikers group. Figure 3.12 shows the results for the registered bikers group.

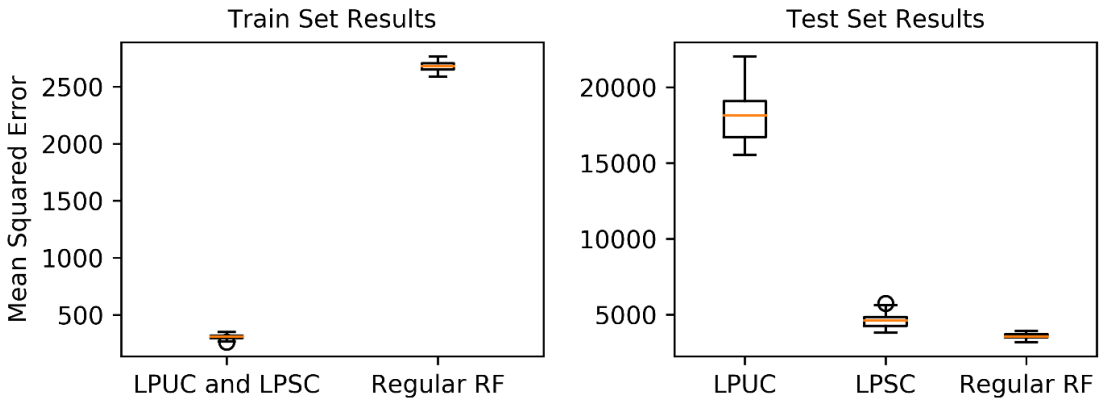

Figure 3.12. Prediction performance comparison between local methods and regular RF for registered bikers

Regular RF outperforms local methods in test sets. Figure 3.13 shows the paired comparison between MSE values of LPSC and regular RF on test sets.

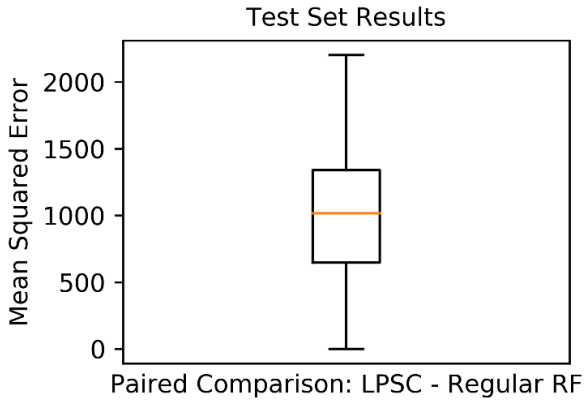

Figure 3.13. Paired comparison of test set MSE values between LPSC and regular RF for registered bikers

Similar to wine quality data, LPUC and LPSC achieve a lower MSE in train sets but a higher MSE in test sets in comparison to regular RF. This could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters.

Let us repeat this experiment by fixing both the number of response bins and the number of feature-based clusters within each response bin to two (M and K parameters in the CBFS

algorithm). This limits the total number of feature-based clusters to four. We hope this reduces the error due to poor assignment of test set cases to feature-based clusters. Furthermore, we limit the maximum depth of RFs to five to reduce the probability of overfitting. Figure 3.14 shows an increase in MSE values for all methods in train sets. This is expected as we decrease the maximum depth of RFs. In the test set, on the other hand, LPSC achieves a lower MSE than regular RF.

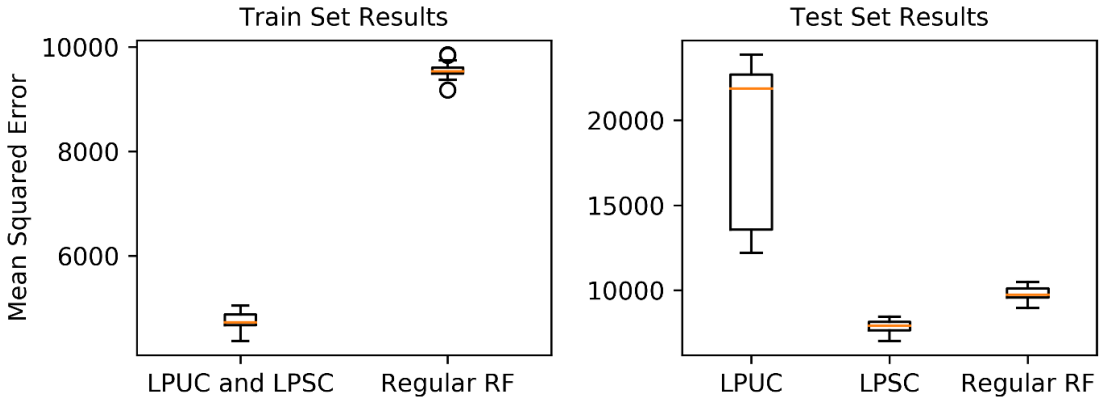

Figure 3.14. Prediction performance comparison between local methods and regular RF for registered bikers group with updated experiment settings

Figure 3.15 shows the paired comparison between MSE values of LPSC and regular RF on test sets in our new experiment for registered bikers. Clearly, LPSC achieves a lower MSE on test sets in comparison to regular RFs.

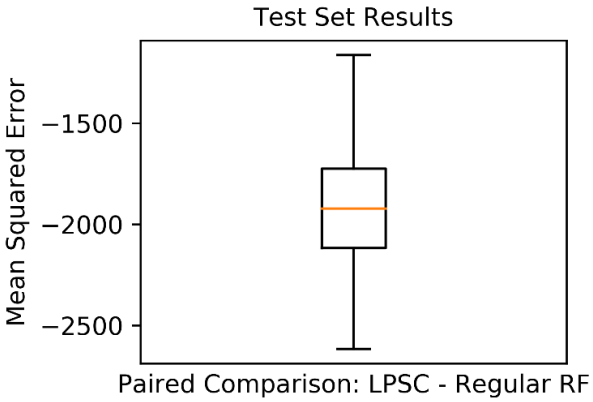

Figure 3.15. Paired comparison of test set MSE values between LPSC and regular RF for registered bikers with updated experiment settings

3.2.3.2 Casual Bikers Group

This section compares the performance of the three methods when applied to casual bikers group data. Figure 3.16 presents MSE values obtained from each method on test and train sets.

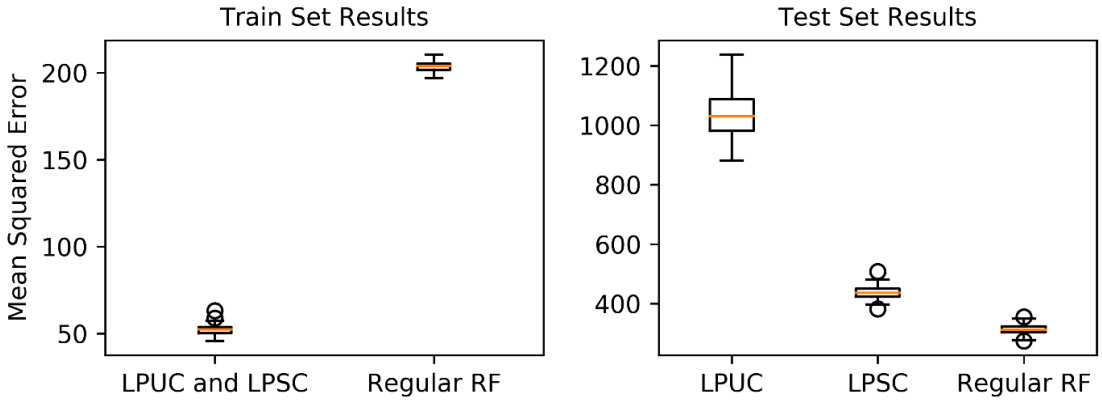

Figure 3.16. Prediction performance comparison between local methods and regular RF for casual bikers

Similar to our first attempt at registered bikers, regular RF outperforms local methods in test sets. We redo this experiment with the same limtations on the number of response bins,

clusters, and RF depth that we used in our second experiment for registered bikers. Figure 3.17 shows the results of this new experiment.

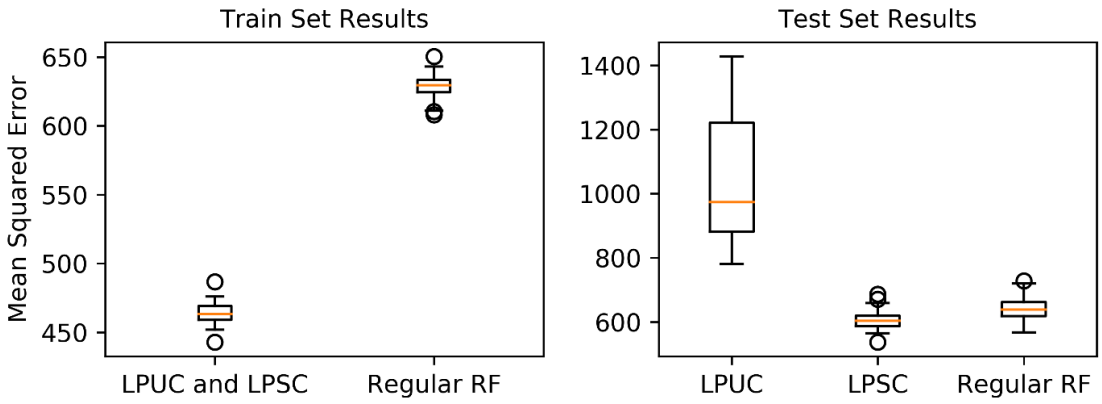

Figure 3.17. Prediction performance comparison between local methods and regular RF for casual bikers group with updated experiment settings

This time LPSC acheives a lower MSE in test sets in comparison to regular RFs. This is more evident in Figure 3.18 which shows the paired comparison of these MSE values.

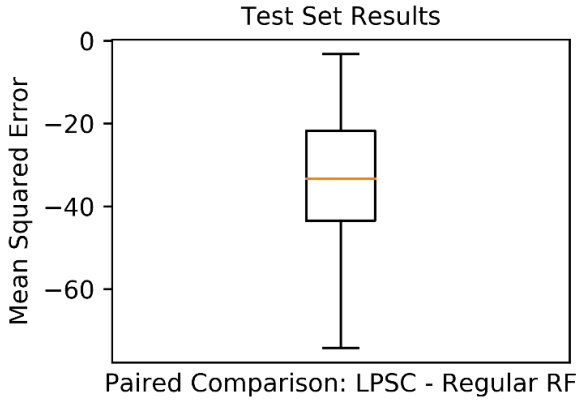

Figure 3.18. Paired comparison of test set MSE values between LPSC and regular RF for casual bikers with updated experiment settings

3.3 Discussions

This chapter proposes LPUC and LPSC as local prediction methods. These methods are compared to regular RF in three simulations. In Simulation 1, LPSC outperforms regular RF in both train and test set. However, LPUC shows poor performance in the test set.

A highly nonlinear response function with interaction terms is used in Simulation 2. For all three methods, MSE values exhibit a large variance. This makes it particularly challenging to make a comparison between candidate methods. That being said, LPUC achieves the lowest MSE among three methods in the test set.

Simulation 3 considers a low signal-to-noise ratio response surface with interactive features. Local methods achieve a lower MSE in the train set but the reverse is true for the test set. This behavior could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters. For example, a large error occurs if a test set case with small response value is predicted by a local RF for a cluster of large response value cases. To further investigate this behavior, we repeat this experiment by limiting the number of response bins and the number of feature-based clusters within each response bin to two (M and K parameters in the CBFS algorithm). This limits the total number of feature-based clusters to four. We hope this reduces the test set error in two ways: (1) by reducing the error due to a poor assignment of test set cases to feature-based clusters; (2) by resulting in a larger number of data points in each cluster which can decrease the chances of overfitting Finally, we limit the maximum depth of RFs to five in a further attempt to reduce the probability of overfitting. Regular RF achieves the lowest

MSE in both test and train sets even after these changes in experiment settings. Note that observations with similar features may end up in different response bins because of the multiplier of the error term in the simulation function. This could be the reason for this behavior.

We use wine quality and bike sharing data as our real-world dataset. Using our default experiment settings, local methods achieve a lower MSE in train sets but higher MSE in test sets in comparison to regular RF in both datasets. Similar to the simulation 3, this behavior could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters. Thus, we repeat these experiments with the same limitations we used in our second experiment for Simulation 3. This time LPSC outperforms regular RF in bike sharing data. In wine quality data, however, regular RF still achieves a lower MSE than both local methods. This could be due to the smaller number of instances in the wine quality dataset in comparison to our generated datasets and bike sharing data.

CHAPTER 4: LOCALLY ADJUSTED FEATURE IMPORTANCE

The CBFS algorithm, which is introduced in chapter Chapter 2:, shows the potential of using local RFs for variable selection. As part of CBFS algorithm, we use only features in making clusters to be able to assign unseen points to these clusters and use the corresponding RF to make local predictions. This idea is explored in local prediction methods proposed in chapter Chapter 3:. It is hypothesized that some part of the prediction error for these local prediction methods come from misassignment of test set points to the feature-based clusters. Thus, this clustering step is not part of the variable selection method introduced in this chapter. This chapter presents a variable selection method called locally adjusted feature importance (LAFI), which identifies locally important features based on different regions of the response variable values.

4.1 LAFI Algorithm

LAFI uses a binary tree approach to split data into bins based on response variable rankings. Next, it uses a RF-based variable selection method to find important variables for each bin separately. This method is described in section 0 below. Finally, the LAFI score is calculated by aggregating the variable selection results from all bins. The binning and aggregating steps are outlined in section 4.1.2.

4.1.1 Variable Selection with Random Forest

The variable selection algorithm described here is applied to each response bin separately. This approach to variable selection is derived from Genuer et al. work on variable selection using RF [10]. The first step is the crude variable selection that is designed to eliminate variables with very little promise. This step is computationally cheap and can reduce feature dimensionality when applied to datasets with a large number of noise variables. The idea behind this step is that noise variables have a close to zero mean feature importance (FI), and any variable with mean FI lower than a threshold should be dropped. This threshold is set to the minimum predicted value obtained from fitting a classification and regression tree (CART) to standard deviations of variables' FI, after sorting variables based on decreasing mean FI. This threshold can be thought of as a conservative estimate of standard deviation of FI scores for low importance variables. The steps of this method are shown in Algorithm 4.1.

Algorithm 4.1: Crude Variable Selection
Input: Tree depth to use in CART, Number of RFs to fit to data M
Output: Selected variables in crude step

1. Fit M RFs to data and calculate mean and standard deviation of FI for each variable
2. Plot standard deviations of FIs in decreasing order of mean FIs
3. Fit a CART to the plot in step 2 with standard deviation of FI for the variables as response and the ranking of mean FI for the variables as predictor. Find the minimum predicted value for SD of FI (response variable), and set this as threshold.
4. Drop any variables with a mean FI smaller than threshold
5. Output remaining variables

We revisit the simulation function in Equation 4 to illustrate the steps of this algorithm.

$$Response = e^{|x_1|} + 10\,sin(x_2) + \varepsilon \qquad (4)$$

10000 data points are generated with x_1 and x_2 independently following a uniform(-5,5) distribution, and the error term with standard normal distribution. Eight independent noise variables with standard normal distribution are added to the data set. The left plot in Figure 4.1 shows FI scores obtained from fitting 50 RFs with 500 trees to this data. The right plot presents the fitted CART to the standard deviation of FI scores. Note that the features are sorted based on mean FI in both plots.

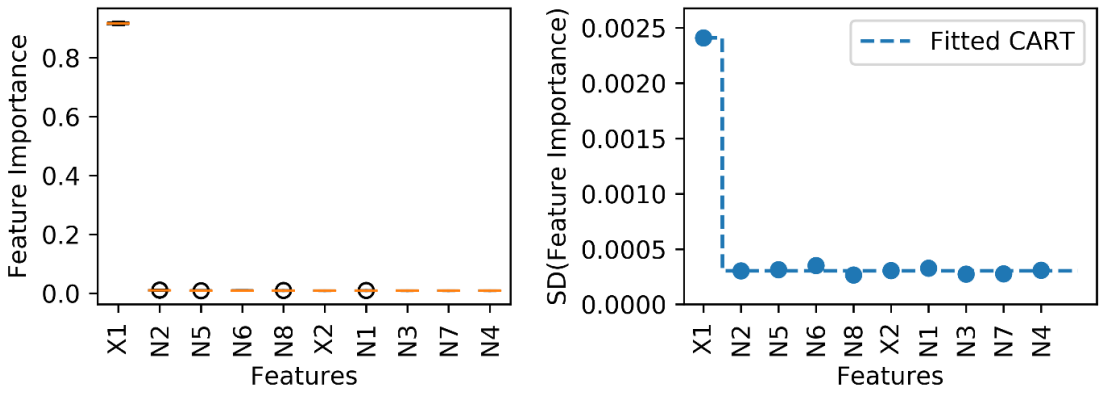

Figure 4.1. FI scores for features in simulated data (left) and fitted CART to standard deviations of these FI scores (right)

The CART's minimum fitted value, which is used as the threshold in this step, is less than 0.0005. Figure 4.2 shows that this threshold is below mean FI of all variables in this simulation. Thus, none of the features are dropped in this step.

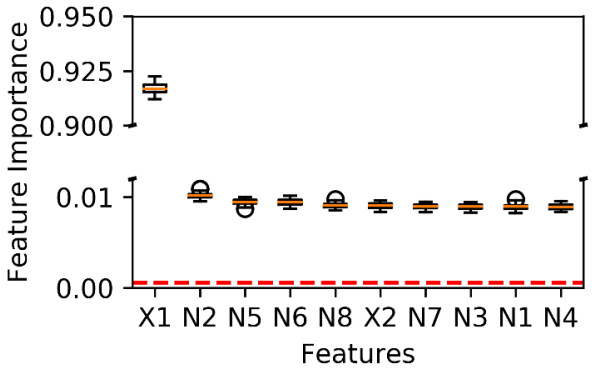

Figure 4.2. FIs obtained from 50 RFs and dashed line indicating variable elimination threshold

The crude variable selection is followed by refined variable selection. This step uses out of bag (OOB) scores as its variable selection criterion. To calculate OOB scores, we first calculate R^2 values on unseen data for each tree. As mentioned in section 1.3, each tree in RF uses a bootstrap sample of original data as input, and unseen data, for each tree, are the ones that are not selected in this bootstrap sample. Next, OOB score is calculated as the average of these R^2 values obtained from all the trees in a RF. See the appendix for further details on OOB scores.

A simple approach would be choosing the model that results in the highest OOB score. However, this can produce unstable results due to the natural randomness in RF algorithm. A more stable option is to pick the smallest model that has a mean OOB score greater than mean OOB score of the best model minus its standard deviation. Finally, the variables used in this model are the selected features in the refined variable selection step. Algorithm 4.2 summarizes the refined variable selection steps.

Algorithm 4.2: Refined Variable Selection
Input: Set of selected features from crude step S, Number of RFs to fit to data H
Output: Selected variables in refined step

1. Fit H RFs to data and use variables in S as input features
2. Sort features in S based on mean FI obtained from the fitted RFs
3. Fit another set of H RFs while using only the most important variable from step 2 as input feature, record OOB scores
4. Add next most important variable to feature space and fit H new RFs, record OOB scores
5. Repeat step 3 until all variables in S have been included
6. Pick the model with highest mean OOB score and subtract the standard deviation of its OOB score from its mean. Set this as threshold.
7. Pick smallest model with mean OOB score larger than the threshold in step 6.
8. Input features of the selected model in step 7 are selected features in this step.

OOB scores for the nested models presented in Figure 4.3 indicate that x_1 is selected in the refined step. This is not surprising if we only consider global behavior of variables since x_1 in the exponential term is dominating the response surface.

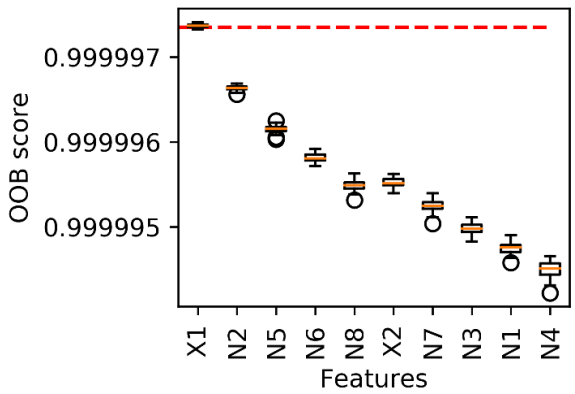

Figure 4.3. Refined variable selection selects x_1 with mean OOB score larger than dashed line threshold

4.1.2 Binning and Aggregating

This step uses the variable selection method described above to pick locally salient variables at different levels. It starts from top level where it uses all data for variable

selection. For the following levels, it recursively binary splits data into bins based on response value rankings, and selects important variables in each bin. At each level, the fraction of bins where a variable is selected is recorded. LAFI is the average of these fractions over all levels. Variables with LAFI smaller than a user-defined threshold are dropped. This threshold and the number of levels are the two parameters of this algorithm. Using more levels in this algorithm increases the chances of selecting locally important variables. However, this also increases the chances of picking noise variables as the bins get smaller and fitted RFs become prone to error due to small sample size. It also significantly increases the computation time. Algorithm 4.3 outlines the steps of binning and aggregating.

Algorithm 4.3: Binning and Aggregating
Input: Number of levels L, User-defined threshold T
Output: Selected Variables based on LAFI

1. Implement variable selection method described in 0 to the whole data (only one bin in level 1)
2. Split bin(s) into two same-sized bins based on response value rankings.
3. Run variable selection in each bin in step 2.
4. At each level, record the fraction of bins in which each variable is picked.
5. Repeat steps 2 through 4 until reaching L levels (should have 2^{L-1} bins in the last level).
6. For each variable, define LAFI as the average of fractions calculated in step 4.
7. Output variables with LAFI larger than T

Figure 4.4 shows the localized variable selection results at level 2 of the simulated data. In bin 1, where cases with lower 50% of response value are kept, both x_1 and x_2 are picked. However, in bin 2, where x_1 inflates the response, x_1 is selected as the only important variable.

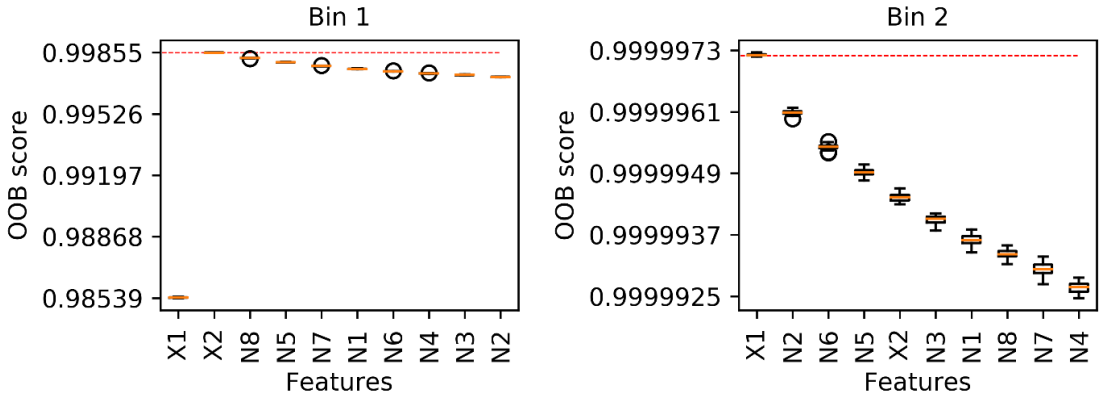

Figure 4.4. Localized variable selection in level 2 bins

Table 4.1 shows the number of bins in which variables are picked at each level. LAFI score is calculated based on this table. For example, N8 is picked at 50% of bins in level 3 (2 out of 4), 12.5% of bins in level 4 (1 out of 8), and 25% of bins in level 6 (8 out of 32). Thus, the average fraction of times N8 is selected over these 6 levels is 14.58%. This translates to LAFI score of 0.1458.

Table 4.1. Selection frequency of variables at each level

	Variables in descending LAFI score order						
	X1	**X2**	**N8**	**N5**	**N1**	**N7**	**N4**
Level 6	31/32	20/32	8/32	14/32	8/32	8/32	6/32
Level 5	16/16	11/16	0/16	4/16	3/16	2/16	3/16
Level 4	8/8	6/8	1/8	1/8	1/8	1/8	1/8
Level 3	4/4	3/4	2/4	0/4	1/4	1/4	1/4
Level 2	2/2	1/2	0/2	0/2	0/2	0/2	0/2
Level 1	1/1	0/1	0/1	0/1	0/1	0/1	0/1

Figure 4.5 shows the results of our proposed method. Variables x_1 and x_2 are selected as the only variables with LAFI larger than T. We use T = 0.2 as an ad hoc value because it separates main features from noise variables in most of our experiments in this chapter. Next, we fit two RF regressors to the data - one with all the variables, i.e. full model, and one with only x_1 and x_2. Excluding all noise variables in the reduced model results in significantly lower MSE.

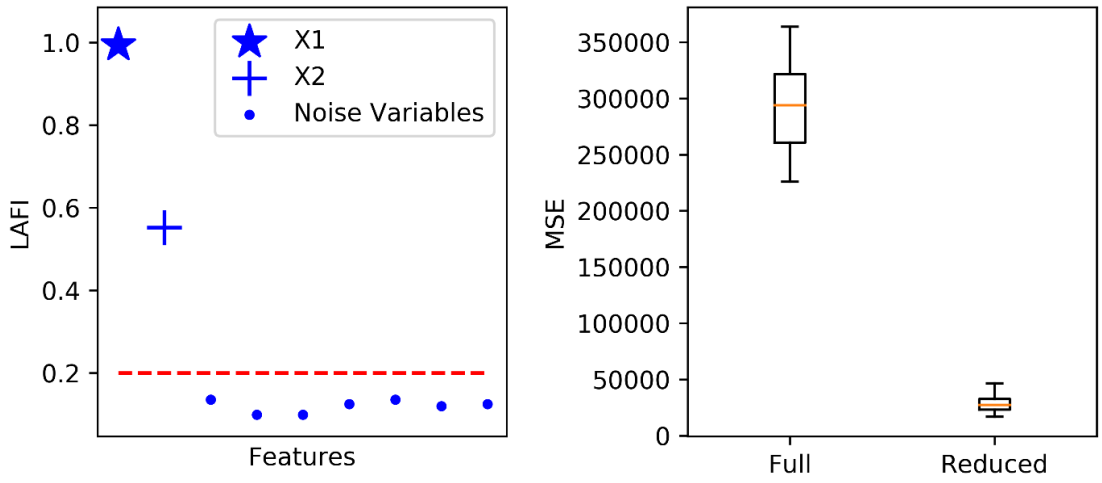

Figure 4.5 Variables x_1 and x_2 are picked based on LAFI using a threshold of T = 0.2. The reduced model with just these selected variables results in improved MSE

4.2 Results

This section presents real-world and simulated datasets to explore the performance of LAFI. We use simulation functions from section 2.2 and the ozone dataset from mlbench package in R to compare the variable selection performance of regular RF and LAFI [58].

RandomForestRegressor and tree functions from the scikit-learn library in Python are used to build RFs and CARTs, respectively. A CART with a depth of 4 and 20 RFs are used in the crude variable selection step. Regarding the refined variable selection step, the H parameter, which is the number of RFs to fit to data, is set to 20. Each RF considers one-third of features at each node and uses 100 trees with a maximum depth of ten. The rest of RF and CART parameters are set to RandomForestRegressor and tree's default values, respectively. Lastly, we use 3 layers and a threshold of 0.2 for the binning and aggregating step.

4.2.1 Simulations

Each simulated dataset consists of 10000 data. We randomly select 75% of the data points are for the train set and the remaining 25% make the test set. Each simulation scenario in this chapter is replicated 50 times. Finally, in each replicate, we compare the prediction accuracy of a RF model with all input features, i.e., a full model, vs. the prediction acccuracy of a RF model with selected variables from LAFI, i.e., a reduced model. These RF models share the same parameters with RF models used in the variable selection step.

4.2.1.1 Simulation 1

We start with the motivational example introduced in section 1.4. Here is the simulation function:

$$Response = e^{|x_1|} + 10\,sin(x_2) + \varepsilon \qquad (4)$$

10000 data points are generated with x_1 and x_2 independently following a uniform(-5,5) distribution, and the error term having a standard normal distribution. Ten independent noise variables, each with a standard normal distribution, are added to the data set. Ideally, x_1 and x_2 would be selected over the noise variables. Figure below shows that LAFI successfully distinguishes the main features, x_1 and x_2, from the noise variables. Furthermore, dropping variables with LAFI score below 0.2 improves the prediction accuracy in both test and train sets. It is notable that each replicate of this experiment takes about 10 minutes to run on 20 CPUs with 2.1 GHz processing speed. This time is longer than the run time of CBFS because of the refined variable selection step which uses nested models. All RF and clustering functions are run in parallel using these CPUs.

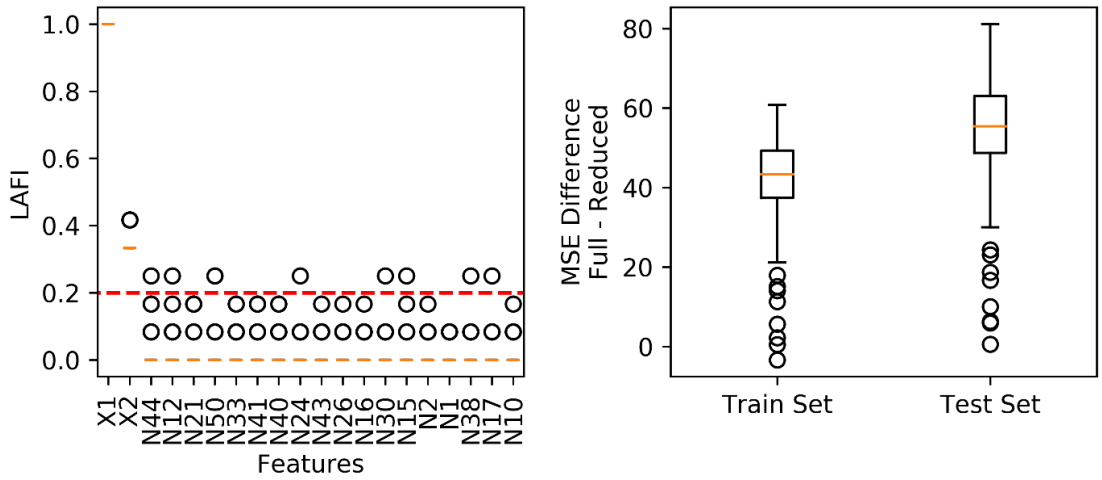

Figure 4.6. Boxplots of LAFI scores across 50 simulations: sorted features based on mean LAFI score (left). Prediction accuracies of full and reduced models (right) in Simulation 1

To compare LAFI scores with FI scores obtained from regular RF, we fit 50 RFs to the same simulated datasets. The results for top 20 variables based on mean FI scores are

presented in Figure 4.7. Several noise variables are picked over x_2 which is a locally important feature.

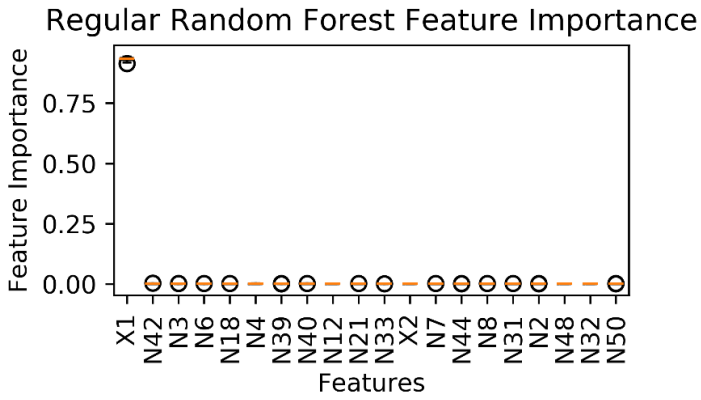

Figure 4.7. Top 20 variables based on mean feature importance obtained from Regular random forest in 50 replications of Simulation 1

4.2.1.2 Simulation 2

A highly nonlinear function with interaction terms is simulated in this section to compare LAFI and regular RF. A sample of 10000 data points is generated using the formula in the equation below:

$$Response = x + y + z^2 + w + q + sin(xy^2) + e^{zwq^2} + \varepsilon \qquad (5)$$

The features and error term all independently follow the standard normal distribution. In addition to $x, y, z, w,$ and $q,$ 50 independent noise variables with the standard normal distribution are added to the data set. The results from 50 replication of this simulation are presented in Figure 4.8. The left plot shows that LAFI picks the main features over noise variables. The right plot compares prediction accuracy of reduced models, which use variables with LAFI score larger than 0.2, and full models.

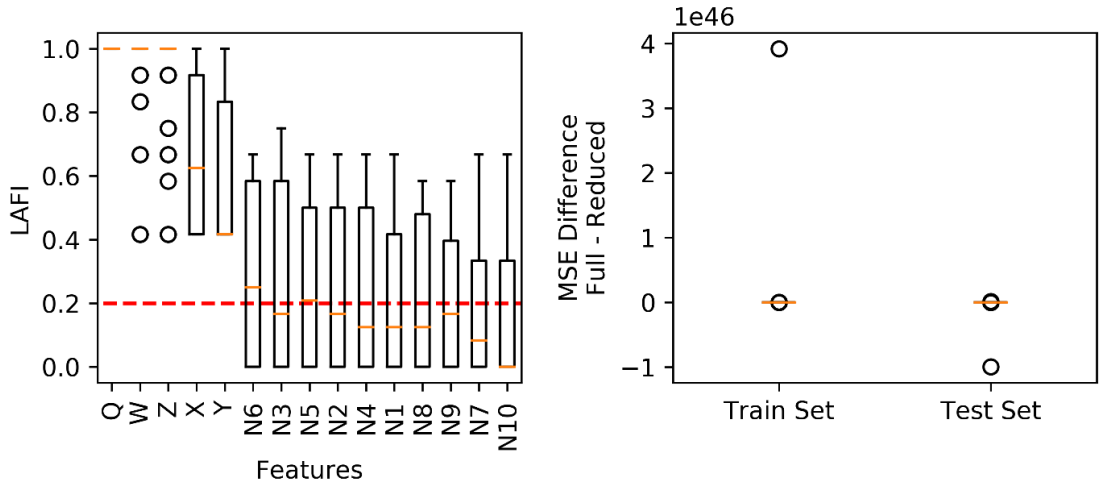

Figure 4.8. Boxplots of LAFI scores across 50 simulations: sorted features based on mean LAFI score (left). Prediction accuracies of full and reduced models (right) in Simulation 2

Note that the ranges of MSE values are very large due to the features in the exponential term in Equation 5. Table 4.2 provides a numerical summary of MSE values. In test sets, MSE values for reduced models have a lower minimum, first quartile (Q1), and third quartile (Q3) but a larger median than full model. Due to this large variability in MSE values, making any comparisons based on MSE is challenging for this simulation function.

Table 4.2. MSE comparison between full and reduced models in Simulation 2

	Train Set MSE		**Test Set MSE**	
	Full Model	**Reduced Model**	**Full Model**	**Reduced Model**
Min	6.90E+08	7.00E+08	6.49E+10	4.68E+10
Q1	7.57E+13	7.62E+13	2.44E+16	2.32E+16
Median	3.04E+19	3.11E+19	8.79E+20	1.24E+21
Q3	4.78E+23	4.90E+23	4.50E+26	2.52E+26
Max	6.89E+47	6.50E+47	1.78E+69	1.78E+69

For the sake of comparison, Figure 4.9 presents the FI scores obtained from regular RF when fitted to the same simulated datasets. Variables are sorted based on mean FI on x-axis. Regular RF only selects three of the main features over noise variables.

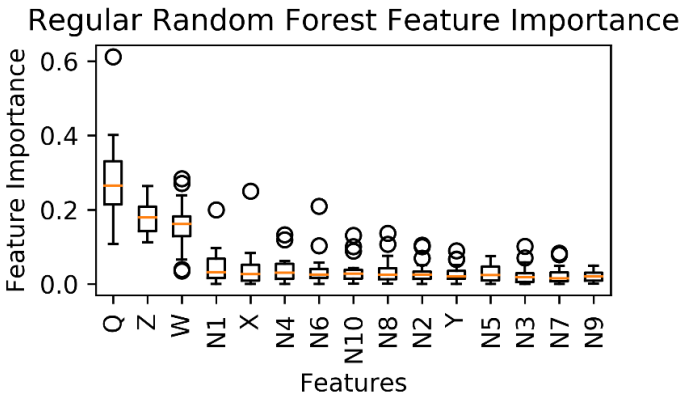

Figure 4.9. Regular random forest feature importance for 50 replications of Simulation 2

4.2.1.3 Simulation 3

The simulation function in Equation 6 presents a low signal-to-noise ratio response surface with underlying interaction terms. Note that presence of 10ε in this formula makes this response surface very noisy. 10000 data points are generated using this equation.

$$Response = xy + \text{wzq} + 10\varepsilon \tag{6}$$

The main features and error term all independently follow the standard normal distribution. 50 independent noise variables with standard normal distribution are added to the data set. Figure 4.10 shows that LAFI selects main features over the noise variables. However, most LAFI scores are above the 0.2 threshold. Thus, variables are rarely dropped and there is not a significant difference in prediction accuracy of full and reduced models.

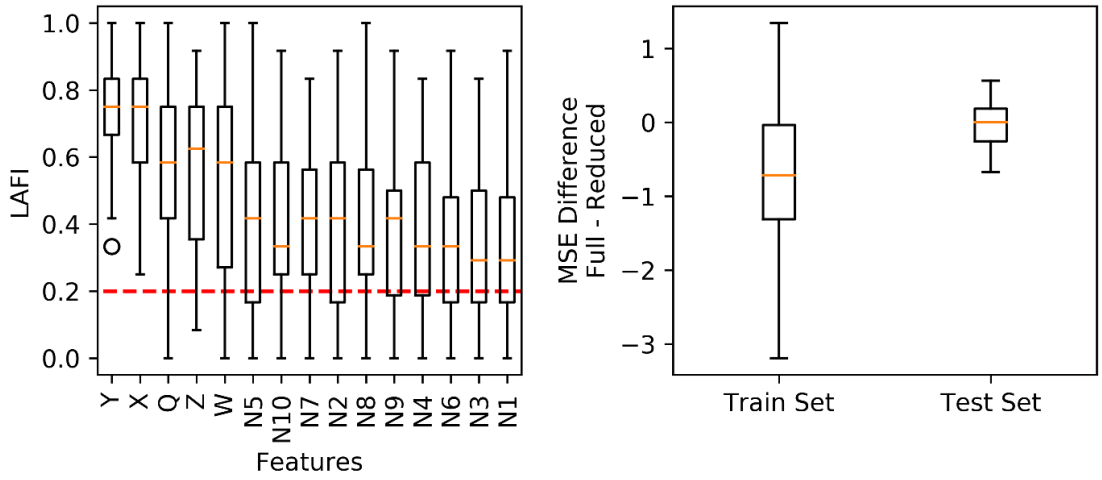

Figure 4.10. Sorted features based on mean LAFI score (left) and prediction accuracies of full and reduced models (right) in Simulation 3

We use regular RF to on the same set of simulated datasets. The resulting FI scores from regular RFs are presented in Figure 4.11. Regular RF fails to select all the main features over noise variables.

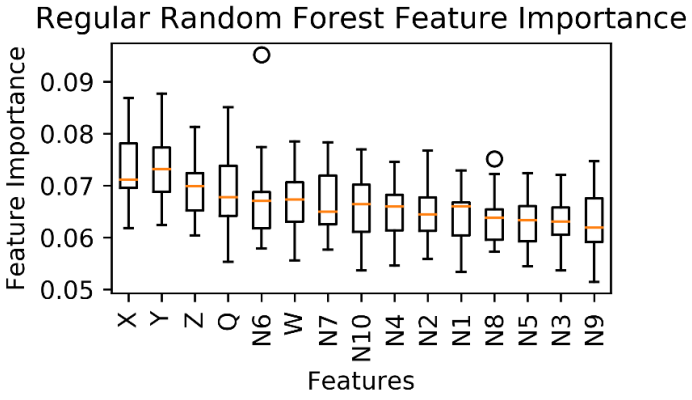

Figure 4.11. Regular random forest feature importance for 50 replications of Simulation 3

Let us repeat this simulation with an improved signal-to-noise ratio by changing the error term multiplier from ten to one in Equation 6.

Figure 4.12 shows a wide gap between LAFI scores of main features and noise variables in most replicates. This results in lower MSE values for the reduced model in both train and test sets.

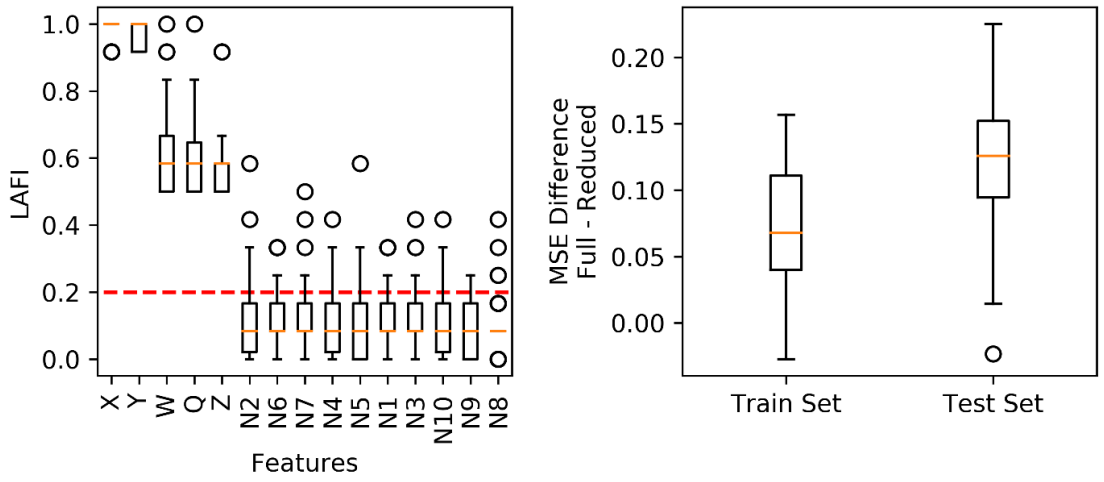

Figure 4.12. Sorted features based on mean LAFI score (left) and prediction accuracies of full and reduced models (right) in Simulation 3 with improved signal-to-noise ratio

Finally, Figure 4.13 presents the FI scores obtained from regular RFs fitted to the same simulated sets of data. There is a clear gap between FI scores of main features and noise variables. Furthermore, none of the noise variables has a larger mean FI score than the main features.

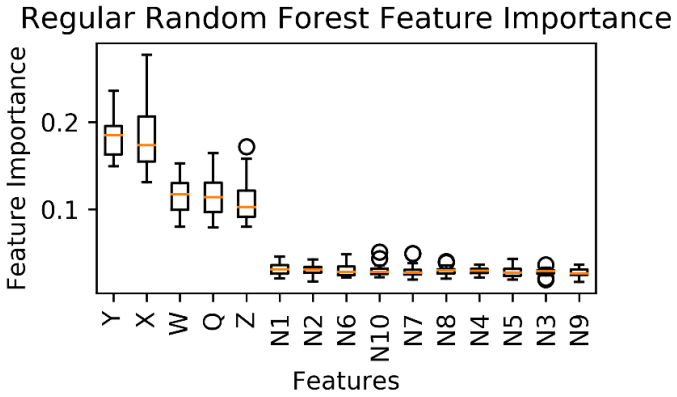

Figure 4.13. Regular random forest feature importance for 50 replications of Simulation 3 with improved signal-to-noise ratio

4.2.2 Ozone Data

This dataset is obtained from mlbench package in R [58]. This is a regression dataset where the reponse variables is daily maximum one-hour-average ozone reading in Los Angeles in 1976. There are 12 features in this dataset: month, day of month, day of week, pressure height, wind speed, humidity, temperature at Sandberg, temperature at El Monte, inversion base height, pressure gradient, inversion base temperature, and visibility. We randomly selected 75% of the data points for the train set and the remaining 25% make the test set. LAFI scores are calculated based on train set data. Next, we compare the prediction accuracy of a RF model with all input features, i.e., full model, to the prediction accuracy of a RF model with selected variables from LAFI, i.e., reduced model. We replicate this experiment 50 times by changing the random seed which results in different train and test sets at each replicate.

Wind speed and weekday have the lowest LAFI scores in Figure 4.14. Intuitively, day of the week does not seem to be a good predictor for ozone level. Regarding wind speed,

Genuer et al. and Chèze et al. came to the same conclusion [10], [59]. In most iterations, both of these variables have a lower than 0.2 LAFI score and are excluded in reduced models. There is no significant difference between MSE values of full and reduced models.

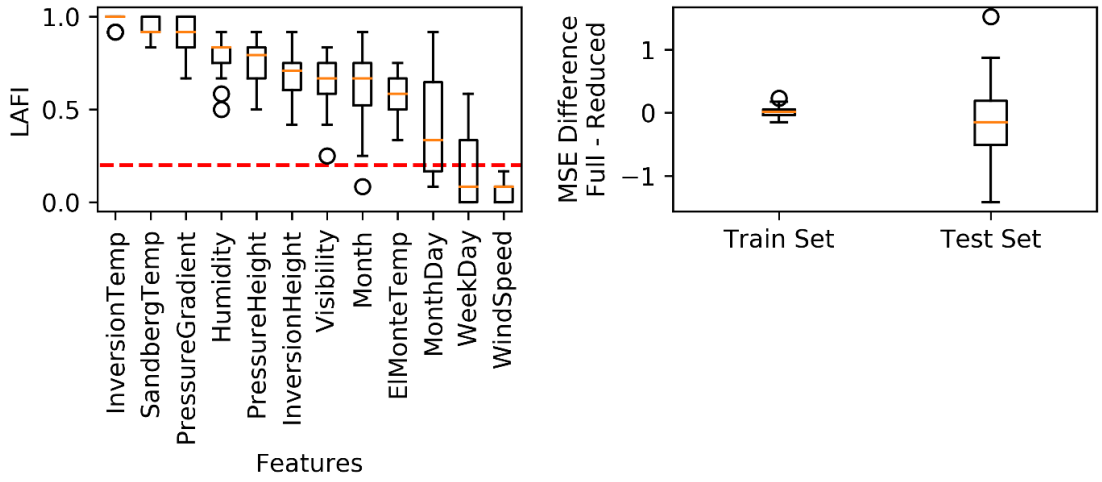

Figure 4.14. Sorted features based on mean LAFI score (left) and prediction accuracies of full and reduced models (right) in Ozone data

Figure 4.15 shows the FI scores from regular RF. Again, wind speed and weekday are found to be the least important features.

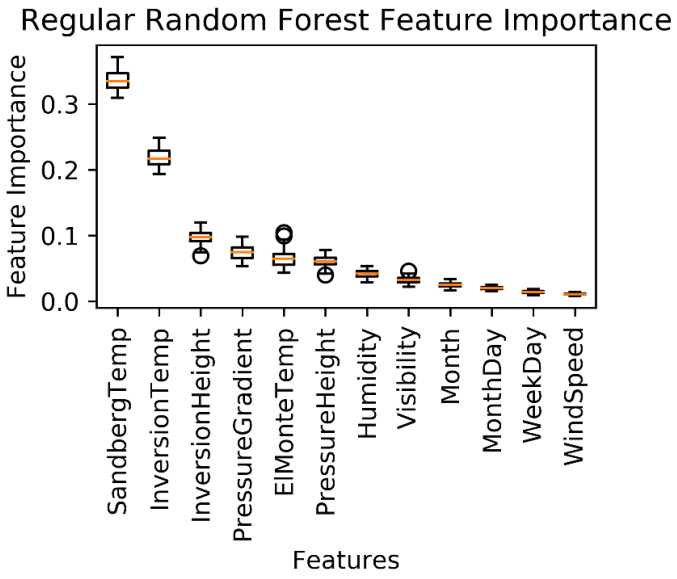

Figure 4.15. Regular random forest feature importance for Ozone data

4.3 Summary

Simulations in this chapter show that FI obtained from regular RF might fail in detecting locally important features in some cases. To address this issue, we introduce LAFI, which takes local importance into consideration. Three simulated datasets, as well as a real dataset, are used to compare LAFI with FI from regular RF. Features with LAFI score larger than a threshold are selected. Next, we compare the prediction accuracies of RF model with all input features (full model) vs. a RF model with selected variables from LAFI (reduced model).

Simulation 1 presents a response surface which is dominated by an exponential term. The main challenge is to detect the effect of less significant feature which is masked by this exponential term. The main features achieve a higher LAFI score than noise variables. On

the other hand, regular FI scores of several noise variables are larger than the FI score of the locally salient feature. In addition, the reduced model, which includes features selected by LAFI, shows a better prediction performance than the full model with all the input features.

A highly nonlinear function with interaction terms is used in simulation 2. All the main features achieve a higher LAFI score than noise variables, while this is not the case for regular FI. The MSE values obtained from full and reduced models are very large due to the features in the exponential term in this simulation function. This makes it challenging to make comparisons based on MSE in this case. That being said, MSE values for reduced models have a lower minimum, first quartile, and third quartile but a larger median than full model.

Simulation 3 presents a low signal-to-noise ratio response surface with interaction terms. Similar to previous simulations, the LAFI scores for main features are larger than noise variables. However, regular RF fails to pick all the main features over noise variables. In most replicates of this simulation, LAFI scores of all input features are larger than our threshold. Thus, the prediction performance of reduced and full models are very close. We repeat this simulation with an improved signal-to-noise ratio. This time both LAFI and FI pick main features over noise variables. This results in lower MSE for reduced models.

Lastly, we use ozone dataset from mlbench package in R. The response variables is daily maximum one-hour-average ozone reading in Los Angeles in 1976. Day of the week and wind speed are found to be the features with lowest LAFI and regular FI score. This result

is in agreement with two other studies that use this dataset. The reduced models achieve similar MSE values to full models but have fewer input features.

CHAPTER 5: CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In some cases, such as the motivational example in chapter 0, regular RF does not detect the effect of locally important variables in presence of dominant features. We have introduced cluster-based feature selection (CBFS) and locally adjusted feature importance (LAFI) as local variable selection methods which aim to address this issue. We have also explored the extension of CBFS for local prediction in chapter 3. This chapter summarizes our conclusions and presents future research directions for our work in previous chapters. Here are future research directions that are applicable to all methods presented in this study:

1) One of the main reasons that we used RF in our algorithms is that it can detect nonlinear patterns in data. However, our methods could work with other variable selection and prediction methods that are capable of detecting nonlinear patterns. Gradiant boosting machine (GBM) is an example of such methods [60]. It is notable that GBMs requires longer training times than RFs. This is why we chose RF over GBM.

2) Our methods are developed for regression problems. Investigating ways to extend these methods to classification problems could be a fruitful research path. This requires changing the split criterion in RF and using a different binning strategy for the response variable. In other words, we can not split data based on response value rankings when the response variable is categorical.

3) Naturally, our results are only generalizable to datasets and simulation settings included in this chapter. Further research is required before inferring our findings to a broader population of data structures. Some examples of such interesting extensions will be – (a) multivariate response data sets, for which our response-based methods have to be modified for deriving locally importance features; (b) graph based learning, for which networks can be considered as part of both response and explanatory variables.

Method-specific conclusions and future research opportunities are presented in the following sections of this chapter.

5.1 Localized Variable Selection with CBFS

We have introduced CBFS which uses binning and clustering to segment data into homogenous regions where the effect of variables can be locally isolated and detected. Next, it uses RF to select features within each cluster. Finally, average and intersection methods are proposed to aggregate the variable selection results from these clusters.

We have used simulations and real datasets to compare the variable selection performance of CBFS and regular RF. In simulations, where we generated 10000 data points for each simulation function, CBFS performed better than regular RF in most cases. However, in real datasets, regular RF was more successful in terms of correctly identifying original input features from added noise variables. We hypothesized that these results were due to

the formation of small clusters in the CBFS clustering step. Having a small number of observations in clusters can result in overfitting of RFs and selection of noise variables. In an attempt to evaluate our hypothesis, we repeated this experiment with a few tweaks in our settings. We used a smaller number of bins and clusters to avoid forming small clusters. This improved the performance of CBFS. In general, CBFS requires a large enough sample size in each cluster to be able to select important features effectively. Finally, we believe CBFS could be improved in the following ways:

1) In some cases, small clusters are formed in CBFS. As a result, RFs in CBFS are exposed to a smaller number of observations and can not efficiently learn patterns in data and detect important features. This could be addressed by using a clustering method with size constraints. Although there has been some research on such algorithms, we could not find an efficient implementation of these algorithms in Python [61], [62].

2) We have used similar parameters for RFs fitted to clusters in CBFS. However, we hope that tuning individual RFs could enhance the results. One way to do this would be to split data in each cluster into train and validation sets and use a grid search on main RF parameters such as number of trees, maximum depth, and number of features to consider at each node. It is also notable that this would make CBFS computationally more expensive.

3) One of the steps of CBFS is feature-based clustering with KMeans algorithm in Python [53]. We have used Euclidean distance in the KMeans algorithm but other distance measures should be investigated as well. Especially, in cases

where categorical features are present in a dataset. We refer the reader to *Similarity measures for categorical data: A comparative evaluation* by Boriah et al. for a literature review on distance measures for categorical variables [63].

4) The feature-based clustering step in CBFS may suffer from the curse of dimensionality in high-dimensional problems. In short, the points would be similarly distant from each other in high-dimensional space and this would make clustering less efficient. See *On k-anonymity and the curse of dimensionality* by Aggarwal for further information on this topic [64]. It is hoped that including an initial crude variable selection step, similar to the one in LAFI's algorithm, could reduce feature dimensionality before the clustering step.

5.2 Local Prediction Methods

As described in chapter chapter 2, features are used to form clusters in CBFS, and one RF is fitted to each feature-based cluster. We use only features in making these clusters to be able to assign new observations to these clusters and use the corresponding RF for the assigned cluster to make local predictions. We have explored two local prediction methods: local prediction with supervised clustering (LPSC) and local prediction with unsupervised clustering (LPUC). LPSC uses cluster assignments of training set data as labels and assigns new points to clusters using a RF classifier. In LPUC, new points are assigned to clusters based on proximity to cluster centers in feature space.

Simulation 3 considers a low signal-to-noise ratio response surface with interactive features. Local methods achieve a lower MSE in the train set but the reverse is true for the test set. This behavior could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters. For example, a large error occurs if a test set case with small response value is predicted by a local RF for a cluster of large response value cases. To further investigate this behavior, we repeat this experiment by limiting the number of response bins and the number of feature-based clusters within each response bin to two (M and K parameters in the CBFS algorithm). This limits the total number of feature-based clusters to four. We hope this reduces the test set error in two ways: (1) by reducing the error due to a poor assignment of test set cases to feature-based clusters; (2) by resulting in a larger number of data points in each cluster which can decrease the chances of overfitting. Finally, we limit the maximum depth of RFs to five in a further attempt to reduce the probability of overfitting. Regular RF achieves the lowest MSE in both test and train sets even after these changes in experiment settings. Note that observations with similar features may end up in different response bins because of the multiplier of the error term in the simulation function. This could be the reason for this behavior.

We use wine quality and bike sharing data as our real-world dataset. Using our default experiment settings, local methods achieve a lower MSE in train sets but higher MSE in test sets in comparison to regular RF in both datasets. Similar to the simulation 3, this behavior could be either due to overfitting of local RF regressors or poor assignment of test set cases to feature-based clusters. Thus, we repeat these experiments with the same limitations we used in our second experiment for Simulation 3. This time LPSC

outperforms regular RF in bike sharing data. In wine quality data, however, regular RF still achieves a lower MSE than both local methods. This could be due to the smaller number of instances in the wine quality dataset in comparison to our generated datasets and bike sharing data. As a potential future research, one could further investigate these results by cross-validating each RF to omit the overfitting option. This would isolate the prediction error due to cluster misassignment.

5.3 Localized Variable Selection with LAFI

In chapter 4, we have introduced LAFI which uses a binary tree approach to split data into bins based on response variable rankings. Next, it uses a RF-based variable selection method to find important variables in each bin. Finally, LAFI score is calculated by aggregating the variable selection results from all bins.

Three simulated datasets, as well as a real dataset, are used to compare LAFI with FI from regular RF. Features with LAFI score larger than a threshold are selected. Next, a RF model with all input features, i.e. full model, is compared to a RF model with selected variables from LAFI, i.e. reduced models, in terms of prediction accuracy. Each experiment is replicated 50 times.

In all simulations, main features had a higher mean LAFI score than noise variable, while this was not the case for regular FI. In addition, reduced models, which included features selected by LAFI, showed a better prediction performance than full models with all the

input features in most cases. Lastly, we used ozone dataset from mlbench package in R [58]. The response variable is daily maximum one-hour-average ozone reading in Los Angeles in 1976. Day of the week and wind speed are found to be the features with lowest LAFI and regular FI score. This result is in agreement with two other studies that used this dataset. Reduced models achieved similar MSE values to full models but had fewer input features.

Finally, here is a list of future research directions:

1) In all of our experiments, we used a depth of 3 for the binary tree that splits data based on response value ranking. Furthermore, we used a threshold of 0.2 on LAFI scores for dropping variables. Further research is needed to find optimal values of these parameters for each dataset. One possible approach would be a grid search to find the set of values which maximize the prediction accuracy of reduced models.

2) For variable selection within each bin, we used an algorithm derived from the work of Genuer et al. on variable selection using RF [10]. We picked this algorithm because of RF's ability to detect nonlinear patterns. However, LAFI could work with other variable selection methods as well.

Bibliography

[1] T. J. Mitchell and J. J. Beauchamp, "Bayesian variable selection in linear regression," *Journal of the American Statistical Association*, vol. 83, no. 404, pp. 1023–1032, 1988.

[2] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1, pp. 389–422, 2002.

[3] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1–3, pp. 37–52, 1987.

[4] H. Wold, "Systems analysis by partial least squares," 1983.

[5] H. Wold, "Partial least squares," *Encyclopedia of statistical sciences*, vol. 9, 2004.

[6] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[7] G. Schwarz, "Estimating the Dimension of a Model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, Mar. 1978.

[8] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[9] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[10] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern recognition letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[11] A. Hapfelmeier and K. Ulm, "A new variable selection approach using random forests," *Computational Statistics & Data Analysis*, vol. 60, pp. 50–69, 2013.

[12] R. Díaz-Uriarte and S. A. De Andres, "Gene selection and classification of microarray data using random forest," *BMC bioinformatics*, vol. 7, no. 1, p. 3, 2006.

[13] V. Svetnik, A. Liaw, C. Tong, and T. Wang, "Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules," in *International Workshop on Multiple Classifier Systems*, 2004, pp. 334–343.

[14] H. Jiang *et al.*, "Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes," *BMC bioinformatics*, vol. 5, no. 1, p. 81, 2004.

[15] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[16] R. Tang, J. P. Sinnwell, J. Li, D. N. Rider, M. de Andrade, and J. M. Biernacka, "Identification of genes and haplotypes that predict rheumatoid arthritis using random forests," in *BMC proceedings*, 2009, vol. 3, p. S68.

[17] M. Wang, X. Chen, and H. Zhang, “Maximal conditional chi-square importance in random forests,” *Bioinformatics*, vol. 26, no. 6, pp. 831–837, 2010.

[18] W. Rodenburg *et al.*, “A framework to identify physiological responses in microarray-based gene expression studies: selection and interpretation of biologically relevant genes,” *Physiological genomics*, vol. 33, no. 1, pp. 78–90, 2008.

[19] M. A. Efroymson, “Stepwise regression–a backward and forward look,” *Florham Park, New Jersey*, 1966.

[20] H. J. Cordell and D. G. Clayton, “A Unified Stepwise Regression Procedure for Evaluating the Relative Effects of Polymorphisms within a Gene Using Case/Control or Family Data: Application to HLA in Type 1 Diabetes,” *The American Journal of Human Genetics*, vol. 70, no. 1, pp. 124–141, Jan. 2002.

[21] C. Telmo, J. Lousada, and N. Moreira, “Proximate analysis, backwards stepwise regression between gross calorific value, ultimate and chemical analysis of wood,” *Bioresource Technology*, vol. 101, no. 11, pp. 3808–3815, Jun. 2010.

[22] R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[23] G. R. Doddington, “Phonetically sensitive discriminants for improved speech recognition,” in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, 1989, pp. 556–559.

[24] M. J. Hunt, S. M. Richardson, D. C. Bateman, and A. Piau, “An Investigation of PLP and IMELDA Acoustic Representations and of their Potential for Combination,” in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, 1991, pp. 881–884.

[25] R. Haeb-Umbach and H. Ney, “Linear discriminant analysis for improved large vocabulary continuous speech recognition,” in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, 1992, vol. 1, pp. 13–16.

[26] R. A. Johnson and D. W. Wichern, “Multivariate Analysis,” *Encyclopedia of Statistical Sciences*, Aug. 2006.

[27] C. R. Rao, “The utilization of multiple measurements in problems of biological classification,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10, no. 2, pp. 159–203, 1948.

[28] J. Fan and R. Li, “Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties,” *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, Dec. 2001.

[29] C.-H. Zhang, “Nearly unbiased variable selection under minimax concave penalty,” *Ann. Statist.*, vol. 38, no. 2, pp. 894–942, Apr. 2010.

[30] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[31] M. Yuan and Y. Lin, “Model selection and estimation in regression with grouped variables,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[32] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, “Sparsity and smoothness via the fused lasso,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[33] R. Tibshirani, “The lasso method for variable selection in the cox model,” in *Statistics in Medicine*, 1997, pp. 385–395.

[34] J. Gui and H. Li, “Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data,” *Bioinformatics*, vol. 21, no. 13, pp. 3001–3008, 2005.

[35] H. Hotelling, “Analysis of a complex of statistical variables into principal components.,” *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[36] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York: Springer-Verlag, 2002.

[37] P. Baldi and K. Hornik, “Neural networks and principal component analysis: Learning from examples without local minima,” *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

[38] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[39] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[40] F. Niyaghi, S. Bhattacharyya, and S. Emerson, “Variable Selection using Intersection and Average of Random Forests,” in *JSM Proceedings, Statistical Learning and Data Science Section*, Baltimore, MD, 2017.

[41] A. Liaw and M. Wiener, “Classification and regression by randomForest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[42] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[43] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, p. 2825−2830, Oct. 2011.

[44] “1.11. Ensemble methods — scikit-learn 0.19.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#random-forest-feature-importance. [Accessed: 24-Apr-2018].

[45] E. Bai, K. Li, W. Zhao, and W. Xu, “Kernel based approaches to local nonlinear non-parametric variable selection,” *Automatica*, vol. 50, no. 1, pp. 100–113, Jan. 2014.

[46] W. Zhao, H.-F. Chen, E.-W. Bai, and K. Li, “Local variable selection of nonlinear nonparametric systems by first order expansion,” *Systems & Control Letters*, vol. 111, pp. 1–8, 2018.

[47] M. A. Winkel, J. W. Stallings, C. B. Storlie, and B. J. Reich, “Sequential Optimization in Locally Important Dimensions,” *arXiv:1804.10671 [stat]*, Apr. 2018.

[48] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[49] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1, pp. 281–297.

[50] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[51] H. Fanaee-T and J. Gama, “Event labeling combining ensemble detectors and background knowledge,” *Progress in Artificial Intelligence*, vol. 2, no. 2–3, pp. 113–127, 2014.

[52] D. Dua and E. K. Taniskidou, “UCI Machine Learning Repository,” 2017.

[53] “sklearn.cluster.KMeans — scikit-learn 0.20.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Accessed: 17-Jan-2019].

[54] “sklearn.metrics.silhouette_score — scikit-learn 0.20.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html. [Accessed: 22-Jan-2019].

[55] “3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 19-Jan-2019].

[56] R. R. Curtin *et al.*, “MLPACK: A Scalable C++ Machine Learning Library,” *Journal of Machine Learning Research*, vol. 14, no. Mar, pp. 801–805, 2013.

[57] “3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 22-Jan-2019].

[58] F. Leisch and E. Dimitriadou, “mlbench: Machine Learning Benchmark Problems, 2005,” *URL http://CRAN. R-project. org/. R package version*, pp. 1–0.

[59] N. Chèze, J.-M. Poggi, and B. Portier, “Partial and recombined estimators for nonlinear additive models,” *Statistical inference for stochastic processes*, vol. 6, no. 2, pp. 155–197, 2003.

[60] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.

[61] B. Babaki, *A python implementation of KMeans clustering with minimum cluster size constraint (Bradley et al., 2000): Behrouz-Babaki/MinSizeKmeans*. 2018.

[62] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," *Microsoft Research, Redmond*, pp. 1–8, 2000.

[63] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: A comparative evaluation," in *Proceedings of the 2008 SIAM International Conference on Data Mining*, 2008, pp. 243–254.

[64] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 901–909.

[65] "scikit-learn/sklearn/ensemble at master · scikit-learn/scikit-learn · GitHub." [Online]. Available: https://github.com/scikit-learn/scikit-learn/tree/master/sklearn/ensemble. [Accessed: 26-Jan-2019].

Appendix: Out of Bag (OOB) Score Calculation

In our variable selection in chapter 4, we have used out of bag scores from RandomForestRegressor function of sklearn.ensemble module in Python [43]. The documentation of this function defines OOB score as coefficient of determination (R^2) on unseen data but does not provide a mathematical equation. Below is part of the source code that calculates OOB scores [65]:

```
for k in range(self.n_outputs_):
        self.oob_score_ += r2_score(y[:, k], predictions[:, k])
self.oob_score_ /= self.n_outputs_
```

It first calculates R^2 values on unseen data for each tree. Note that each tree in RF uses a bootstrap sample of original data as input and unseen data are the ones that are not selected in that bootstrap sample. Next, OOB score is calculated as the average of these R^2 values obtained from all the trees in RF. To write this in a mathematical form, let us define the following notations:

$\boldsymbol{T}$: Number of trees in RF

$\boldsymbol{C_t}$: Set of instances from the original pool of data points that are not picked in bootstrap sample of t^{th} tree; where t ranges from 1 to T

$|\boldsymbol{C_t}|$: Number of elements in set C_t

$\boldsymbol{x_{ti}}$: Feature vector of i^{th} instance in C_t

$\boldsymbol{y_{ti}}$: Observed response value for x_{ti}

$\overline{\boldsymbol{y}}_{\boldsymbol{t.}}: \frac{\sum_{i=1}^{|C_t|} y_{ti}}{|C_t|}$

$\widehat{f}_t(\boldsymbol{x_{ti}})$: t^{th} tree predicted response value for x_{ti} (note that x_{ti} is not used in training of t^{th} tree)

Finally, OOB score is calcuated by the formula below:

$$OOB\ Score = \frac{\sum_{t=1}^{T}[\ 1 - \frac{\sum_{i=1}^{|C_t|}\left(y_{ti} - \widehat{f}_t(x_{ti})\right)^2}{\sum_{i=1}^{|C_t|}(y_{ti} - \bar{y}_{t.})^2}\ ]}{T} \quad (7)$$