Rasmus Hilmer Henninen

Bachelor's thesis

Explainable Artificial Intelligence

Explaining predictions with partial dependence and accumulated local effects plots

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2021

NTNU
Norwegian University of
Science and Technology

Rasmus Hilmer Henninen

Explainable Artificial Intelligence

Explaining predictions with partial dependence and accumulated local effects plots

Bachelor's thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

This thesis aims to look at the potential usefulness of two explainable AI methods for analyzing black-box models. To do this a black-box model from statistical learning has to be used, for this thesis the tree based method random forest is used. The two methods that are used are partial dependency plots (PDP) and Accumulated Local Effect plots (ALE plots).

To build up an understanding of how random forests works the thesis starts with the theory behind tree based models. Then the core ideas that lead to random forests is presented. Finally the main topic of the thesis is introduced: PDP and ALE plots. Both the theory of the methods are covered, and a practical implementation on both real and simulated datasets is analyzed.

ALE plots should in theory perform better than PDP on correlated datasets, surprisingly this was not the case in practise. When trying to reproduce this on the Boston house dataset, where the covariates are quite correlated, and on a simulated dataset both methods produced almost the same results.

Contents

List of Figures

List of Tables

1 Introduction

In statistical learning there are two main categories that most problems fall into: unsupervised- and supervised learning. In unsupervised learning problems the data consists only of measurements denoted by x_i, and the goal is to discover some underlying structure in data. Examples are clustering algorithms like k-means that attempt to find grouping in the data, or dimensionality reduction algorithms like principal component analysis which tries to reduce the dimension of the data by finding a more efficient way of representing it.

Meanwhile supervised learning is when the data consists of pairs of measurements $\{x_i, y_i\}$ of measurements and labels. The goal of supervised learning is to construct a function $\widehat{f}(x)$ that can take new measurements x_i as input, and predict the label y_i. As an example consider the task of classifying a picture by whether it has a cat in it or not. The RBG values in a picture could be the x_i and the y_i could be a label "cat" or "no cat".

Depending on whether the function $\widehat{f}(x)$ returns a continuous or a discrete quantity the underlying theory is a little different. For discrete variables this is called a classification model. Classifying a picture as having a cat or not is an example of this. But another way of solving the problem would be predicting a probability of there being a cat in the picture instead, which would be a continuous variable.

When choosing which what model to use to solve a problem there is a trade-off in terms of interpretability and performance. Simpler models are easier to understand, but perform worse if they are unable to capture the complexities of the data they are used to model. Complex models are difficult to interpret, but have substantially better performance when it comes to capturing the complexities of data.

Recently interpretability has become more and more important, either due to countries enforcing interpretability by law, or businesses requiring it for practical reasons. In Europe the GDPR adds some restrictions in terms of a right to explanation (Selbst and Powles, 2017), which means models that perform evaluations need to be accompanied with a human understandable explanation. This becomes a quite strong limitation on which models can be used. For a lot of businesses the simpler models perform so poorly that they are unusable, while the complex models lack sufficient interpretability to be used. Tree-based models exemplifies the trade-off between interpretability and performance. Naive decisions trees are easy to understand, but their performance is so poor that they are practically unusable in practise. The complex counterpart to naive decision trees is random forests, which perform quite well but due to the complexity in the construction it is almost impossible to get a grasp of how they function.

To be able to use more complex models, such as random forests, methods to be able to analyze them have to be developed. Currently the field of Explainable AI (XAI) is quite new so not many such methods exists, but two potentially promising methods for analyzing complex models were quite recently published. In Apley and Zhu (2016) published a thesis with a method called Partial dependency plots (PDP), and in Apley and Zhu (2016) published a suggested improvement to PDP called Acumulated local effect plots, which might both be used to get a better understanding of complex models.

2 Tree models

One class of supervised models are tree based models, which can be used for both classification and regression problems. To accomplish this, tree-based models approximate an unknown underlying function with piecewise functions:

$$\widehat{f}(x) = \sum_{m=1}^{M} c_m 1_{(X \in R_m)}$$

The notation $c_m 1_{(X \in R_m)}$ is for an indicator function where if $x \notin R_m$ then the function evaluates

to 0 and if $x \in R_m$ it evaluates to c_m. This can be interpreted as splitting the domain of the function $\widehat{f}(x_i)$ into M different regions R_m, and if a new measurement x_i is in the region R_m then $\widehat{f}(x) = c_m$, where the function evaluates to 0 when $x \notin R_m$ and c_m otherwise.

As an example consider the function $f(x) = 3 + \sin(x\pi/2)$ where $x \in [0, 4]$. For a plot of the approximation where the feature space is split into $R_1 = [0, 2]$ and $R_2 = [2, 4]$ and

$$c_1 = \arg\max_{x \in R1} f(x) = 4$$

$$c_2 = \arg\min_{x \in R2} f(x) = 2,$$

see Figure 1.

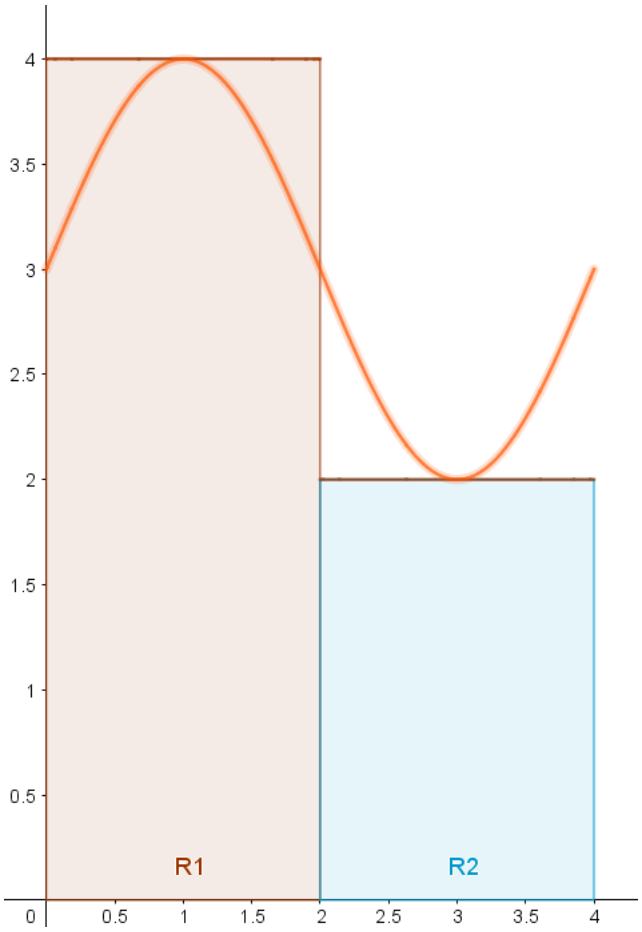


Figure 1: Graph of $\widehat{f}(x)$ made with GeoGebra

If a new measurement is in the region $R1$, which is the interval [0,2], then the model predicts that the associated label y_i should be $c_1 = 4$, while if the measurement is in $R2$, the model would predict $c_2 = 2$ instead. Here both the regions and the values were chosen arbitrarily, and the approximation doesn't look very good. How good or bad is this fit? Is there a way to select the regions and associated c_i's better? And how can we construct a method when the underlying function is not known?

2.1 Loss function

One of the ways of measuring how well a model fits is to introduces a loss function. Let $(x_i, y_i), i = 1, ..., N$ be a data set. We utilize the residual sum of squares (RSS) which is defined as:

$$\text{RSS} = \sum_i (f(x_i) - \widehat{f}(x_i))^2,$$

where $f(x_i)$ is the i-th value of the response in the data, and $\widehat{f}(x_i)$ is the models predicted value of the response. The sum can be taken over the entire data set, or just the training set if it has been split into a test set and a training set.

This gives the way of quantifying how different the model prediction is from the true function. If the model agrees with function on all the measurements, then the RSS will be zero. And if the RSS is far from 0 then that implies that the model doesn't fit very well with the true function. By squaring the errors the model will favor not having larger errors, and instead have more smaller errors when possible.

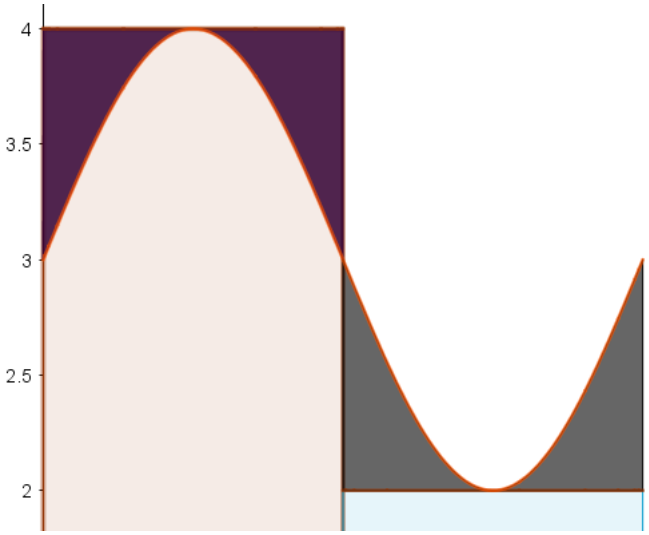


Figure 2: Shaded area shows $f(x) - \widehat{f}(x)$.

In Figure 2 the potential values of x_i are shown. When the actual values of $f(x)$ and the predicted values are close the difference is small, and this is shown as a small shaded line between them. But when the predicted values are far from the actual values there is a large shaded line between them.

2.2 Sampling $f(x_i)$ and different representations

Usually the underlying function is not known, but it is usually possible to sample from it with some errors. To simulate this: Sample x's uniformly on [0,4] calculate $f(x_i)$, and add an error term ϵ such that $\epsilon \sim N(0, \sigma^2)$. Doing this in R shows how regression trees can be implemented when f is unknown. To demonstrate this the R package ggplot2 Wickham (2016) is used to plot an example.

Figure 3 shows 50 sampled points from $f(x)$ with noise added, the original function $f(x)$, and a simple tree based model. Here the values for c_1 and c_2 are chosen such that they minimizing the RSS on the simulated data points.

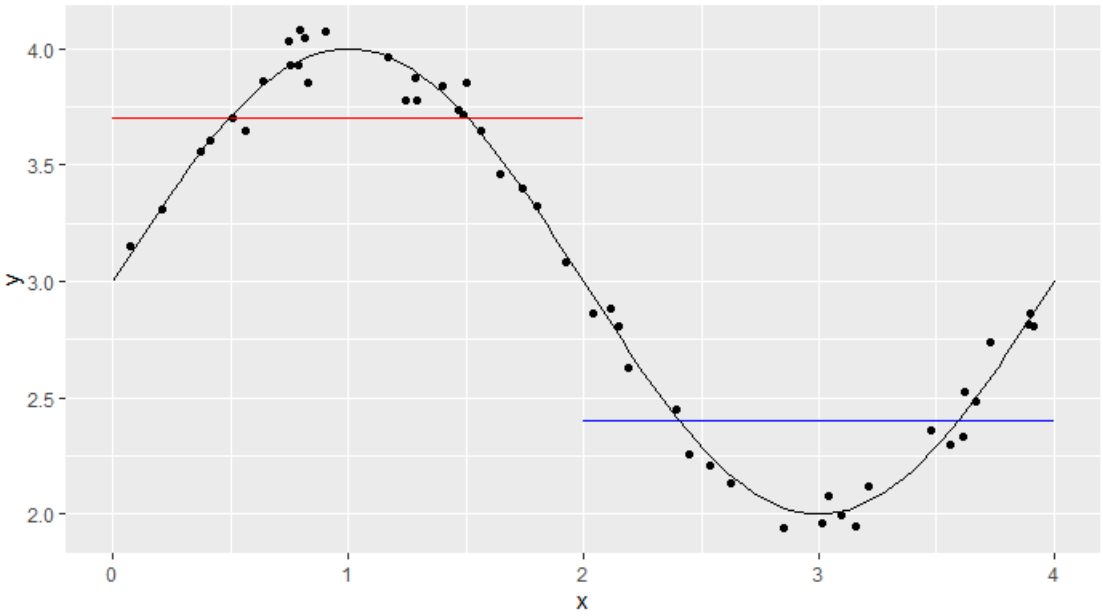


Figure 3: Plot of the function $f(x)$, sampled points, and lines representing $\widehat{f}(x)$.

The name "tree based models" comes from a more suitable way of representing these types of models. First notice that the estimated function $\widehat{f}(x)$ is a step function, namely:

$$\widehat{f}(x) = \begin{cases} 3.7 & x \leq 2 \\ 2.4 & x \geq 2, \end{cases}$$

which can more naturally be be represented by showing which rules lead to which prediction. Figure 4 shows how this tree model can more simply visualized as a binary decision tree. To see how a new data point would get classified: Start at the top of the tree and at each split go left or right depending on if the condition is meet or not.

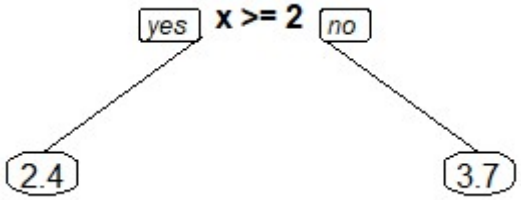


Figure 4: Representing the same $\widehat{f}(x)$ as a binary tree, using the rpart R package.

2.3 Construction of trees

Unfortunately picking the optimal region and c_i's has been shown to be NP-complete, which means any algorithm that attempts to find the optimal solution will not be able to run quickly on large datasets (Laurent and Rivest, 1976).

2.3.1 Algorithm 1: Brute force

With RSS as the loss function it is easy to find the optimal c_m by minimizing the RSS in the following manner:

$$\arg\min_{\widehat{f}(x)}(RSS) = \arg\min_{\widehat{f}(x)}(\sum_i (f(x_i) - \widehat{f}(x_i))^2),$$

which is minimal when $\widehat{f}(x) = ave(y_i)$ where $ave(y_i)$ is the mean value of y_i. Thus for smaller datasets a straight forward approach works. Assuming that the data consists of only discrete measurements, simply test every potential partition of the feature space where a split is made on each of the measurements. To test the partition, calculate the $c_m = ave(y_i|x_i \in R_m)$ for each region, then calculate the RSS. Then finally pick the partition that provides the lowest RSS. Doing this on the same function as in the previous examples produces the tree in Figure 5:

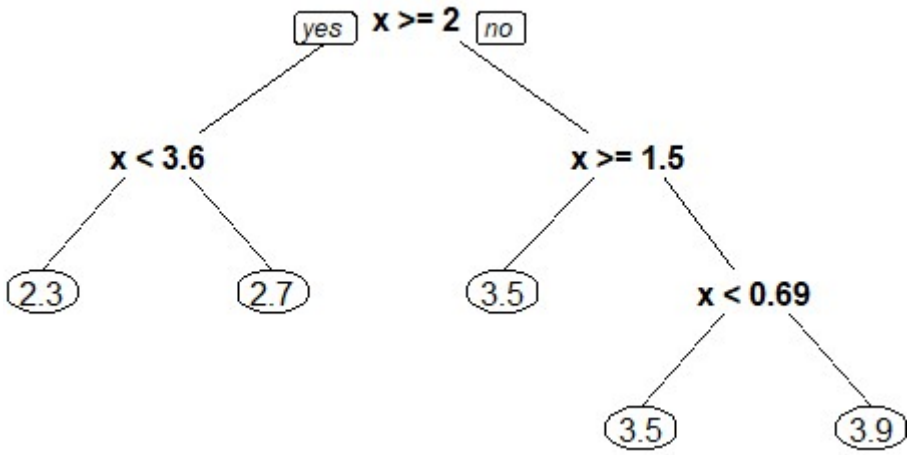


Figure 5: Tree representation of the best partition for the data in Figure 3.

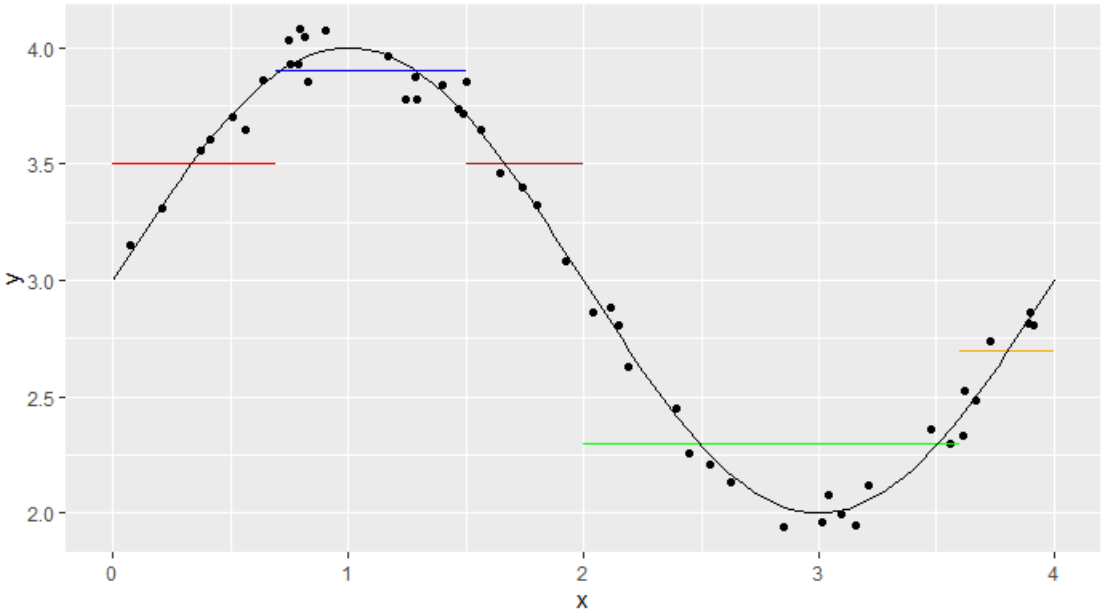


Figure 6: Function representation of the best partition, also shown as a tree in Figure 5.

2.3.2 Algorithm 2: Recursive binary splitting

As it is computationally infeasible to find the optimal solution for bigger datasets a greedy approach has to be implemented instead of using the naive implementation. One way of doing this is by attempting to minimze the RSS with recursive binary splits. The RSS can be decomposed the following way:

$$RSS = \sum_i (f(x_i) - \widehat{f}(x_i))^2 = \sum_{x_i \in R_1} (f(x_i) - \widehat{f}(x_i))^2 + \sum_{x_i \in R_2} (f(x_i) - \widehat{f}(x_i))^2.$$

Evaluate every combination of binary splits, and find the split with the lowest RSS. This splits the feature space into two new regions R_1 and R_2 with associated values c_1 and c_2. This procedure then is repeated on the new regions R_1 and R_2, and on subsequent regions until the stopping criterion is met. In this thesis only the simplest stopping criterion is considered: stop when there are too few observations in the leaves (default = 10 in the tree R package), or when the split would result in a node with too few observations (default = 5 in the tree R package).

2.4 Pros and cons of regression trees

"Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy." -Hastie et al. (2009) As the quote highlights the main flaw of decision trees is their natural tendency to overfit to the training data, and hence a low accuracy on the test data. Despite having good qualities like being easy to interpret, and being usable for both regression and classification, other methods outperform the naive implementations of decision trees. Improvements have been found to increase the accuracy of decision trees, but they come at a cost to the interpretability of the models.

3 Improving on regression trees

Some of the drawbacks of using decisions trees can be mitigated by a variety of different techniques, but in this text only the techniques relevant to random forests will be covered. The core ideas are bootstrapping, bootstrap aggregation and a slightly different way of constructing trees are the ideas to turn decision trees into random forests. So to understand random forests each of these ideas are covered in turn, starting with bootstrapping.

3.1 Bootstrap datasets

Bootstrapping is a procedure where we create new datasets from one original dataset. Start with a dataset D with n elements and make a new data set of the same size by randomly sampling with replacement from the old one.

By sampling with replacement it is likely that some of the observations in the data set may not be chosen and these are called out of bag samples. The out of bag samples consist of roughly one third of the data assuming the dataset is large enough.

Proof: The probability that an observation is not chosen is $(1 - \frac{1}{n})^n$, taking the limiting process of this as n gets large: $\lim_{n\to\infty}(1 - \frac{1}{n})^n = e^{-1} \approx 0.37 \approx \frac{1}{3}$

Out of bag samples can be used to estimate the loss of the model on new data by calculating the RSS of the out of bag samples. Regression tree are made with bootstrapped subset of the data and the subset that has not been used in the creation gets used to check the loss.

3.2 Bootstrap aggregation

The core idea behind bootstrap aggregation is that each decision tree is only capable of learning simple patterns, but by making multiple decision trees out of different bootstrapped data sets and aggregating the results (taking the average) the model can capture more complex patterns.

As each tree is made with a different bootstrapped dataset they are only able to use a subset of the data. This helps reduce the variance, but only if the trees that are made are not strongly correlated.

Algorithm 1 Bootstrap aggregation

procedure BOOTSTRAP AGGREGATION(X, Y, B) ▷ B = Hyperparameter
  for $b = 1$ to B **do**
    $X_b, Y_b \leftarrow$ samples with replacment from X and Y
    $f_b \leftarrow$ MakeTree(X_b, Y_b)
  end for
  return $\widehat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_b)$
end procedure

3.3 Random forests

The issue with using bootstrap aggregation is that dominant features might cause all the trees to be highly correlated. To see this consider the variance of B identically distributed random variables X with mean μ and Var $= \sigma^2$. Furthermore assume the covariance is given by $\text{Cov}(X_i, X_j) = \rho\sigma^2, i \neq j$ then

$$\text{Var}(\bar{X}) = \sum_{i}^{B} \frac{\text{Var}(X_i)}{B} + 2\sum_{i=2}^{B}\sum_{j=1}^{i-1} \frac{\text{Cov}(X_i, X_j)}{B^2} = \rho\sigma^2 + \frac{1-p}{B}\sigma^2.$$

As B grows large the second term is negligible, but the first term will remain. This can be interpreted simply as: having a lot of decision trees won't necessarily increase performance if all of them are nearly identical and only consider the same predictors. To remedy this a technique is used to attempt to decorrelate the trees: each tree is only allowed to use a random subset m of the features for splitting. The wisdom of the crowd is to set m equal to one third of the number of features. This is done to avoid the issues of each tree being highly correlated if they are based on the same features. Following the example of a hiring committee, this can be seen as only allowing each of the members of the committee to see a part of the CV of a potential employee. This would make sure that the whole CV is being considered, instead of just a subsection which could help avoid potential biases.

Algorithm 2 Random forests

procedure RANDOM FORESTS(X, Y, B)
  for $b = 1$ to B **do**
    $X_b, Y_b \leftarrow$ samples with replacement from X and Y
    $f_b \leftarrow$ MakeRandomTree(X_b, Y_b)
  end for
  return $\widehat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$
end procedure

3.4 The Boston house dataset

The Boston house dataset is a dataset that is commonly used to benchmark algorithms and methods in statistics and machine learning. The data was collected by the U.S. Census Service and contains information about the housing in 506 houses in the Boston Mass area, and was published in Harrison and Rubinfeld (1978). For each of the houses the following 14 attributes were recorded:

- **crim**: per capita crime rate by town
- **zn**: proportion of residential land zoned for lots over 25,000 sq.ft
- **indus**: proportion of non-retail business acres per town
- **chas**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **nox**: nitric oxides concentration (parts per 10 million)

- **rm**: average number of rooms per dwelling
- **age**: proportion of owner-occupied units built prior to 1940
- **dis**: weighted distances to five Boston employment centres
- **rad**: index of accessibility to radial highways
- **tax**: full-value property-tax rate per USD 10,000
- **ptratio**: pupil-teacher ratio by town
- **b**: $1000(B - 0.63)^2$ where B is the proportion of blacks by town
- **lstat**: percentage of lower status of the population
- **medv**: median value of owner-occupied homes in USD 1000's

Performing a datset analysis of the dataset is be a quite large task, and is outside the scope of this thesis. To get some insight into how the data looks like the R function ggpairs is used on the variables that we use later in the thesis. Ggpairs is function from the GGally package and produces a matrix of plots, and a plot can bee seen in Figure 7. The matrix contains scatter plots of the variables against each other below the variable distribution on the main diagonal, and Pearson correlation coefficients above.

As the data is quite correlated the correlation coefficients are quite large, and this can be seen by some quite clear patterns in the scattersplots. Looking for example at dis against nox there is a quite noticeable negative correlation, high values of nox usually coincide with low values of dis.

The distributions of the variables are quite different. The variables nox, lstat and dis are quite right-skewed and have mostly relatively low measurements, while ptratio is left-skewed with a lot of high measurements.The variable rm looks to be almost normally distributed, while the indus distribution almost looks like it is bimodal.

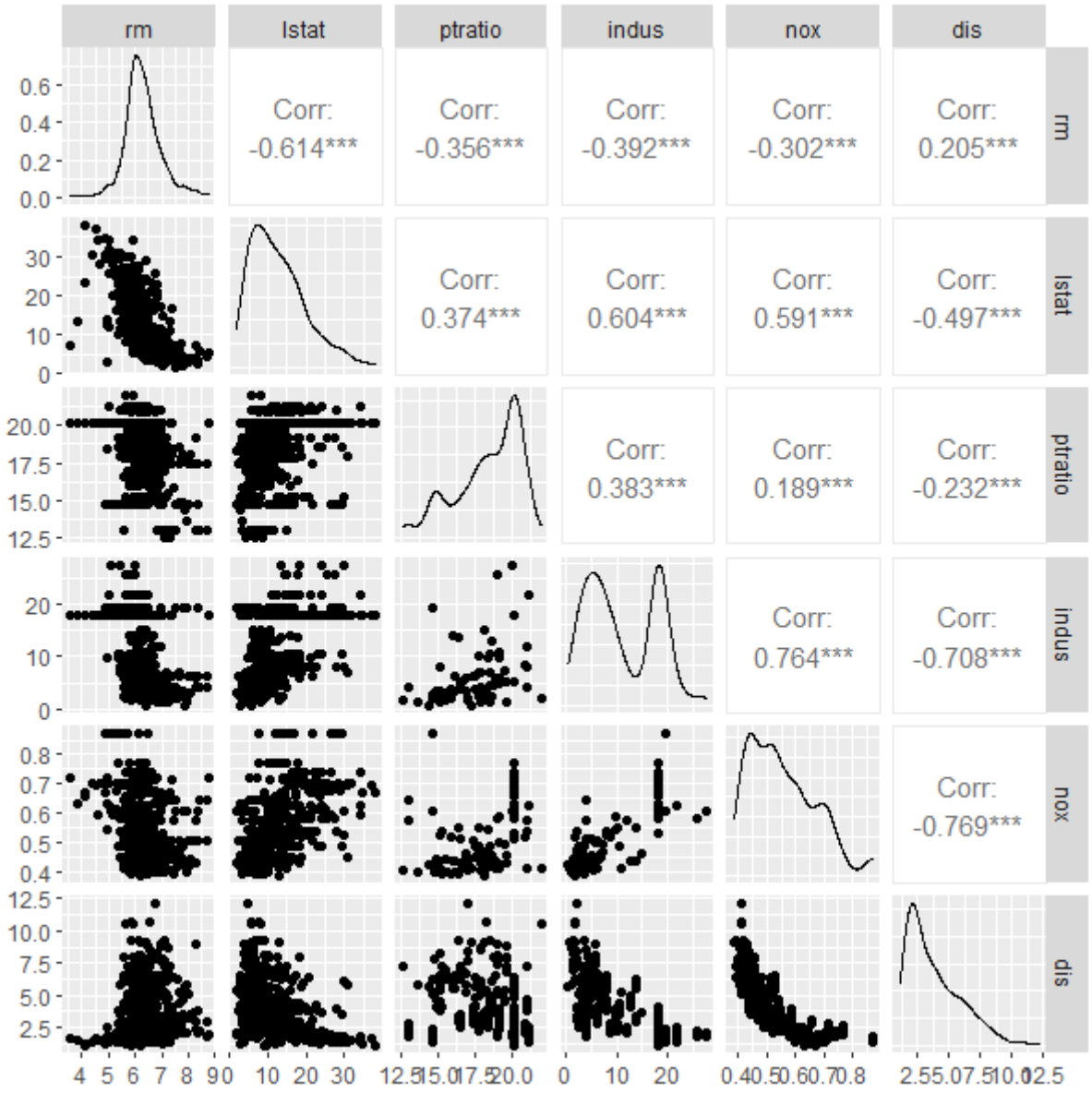


Figure 7: Matrix of plots made with the R package GGally

3.5 Verifying model improvements on the Boston house dataset

The models are implemented with house prices as the response and the other variables as predictors. The data is split into a training set containing 70% of the data, and the rest is used for testing.

We train the models in the way described so far, with $m = \lfloor \frac{14}{3} \rfloor = 4$ for random forest, on the training data. We then use the rest of the data to compare the predictions with the real values. For regression trees a RSS of 26.83 is achieved, while bagging the trees gets an RSS of 16.61. But, both methods get outperformed by random forest with a RSS of 15.20. This is a clear improvement using bagging, and just a slight improvement from bagging to random forests.

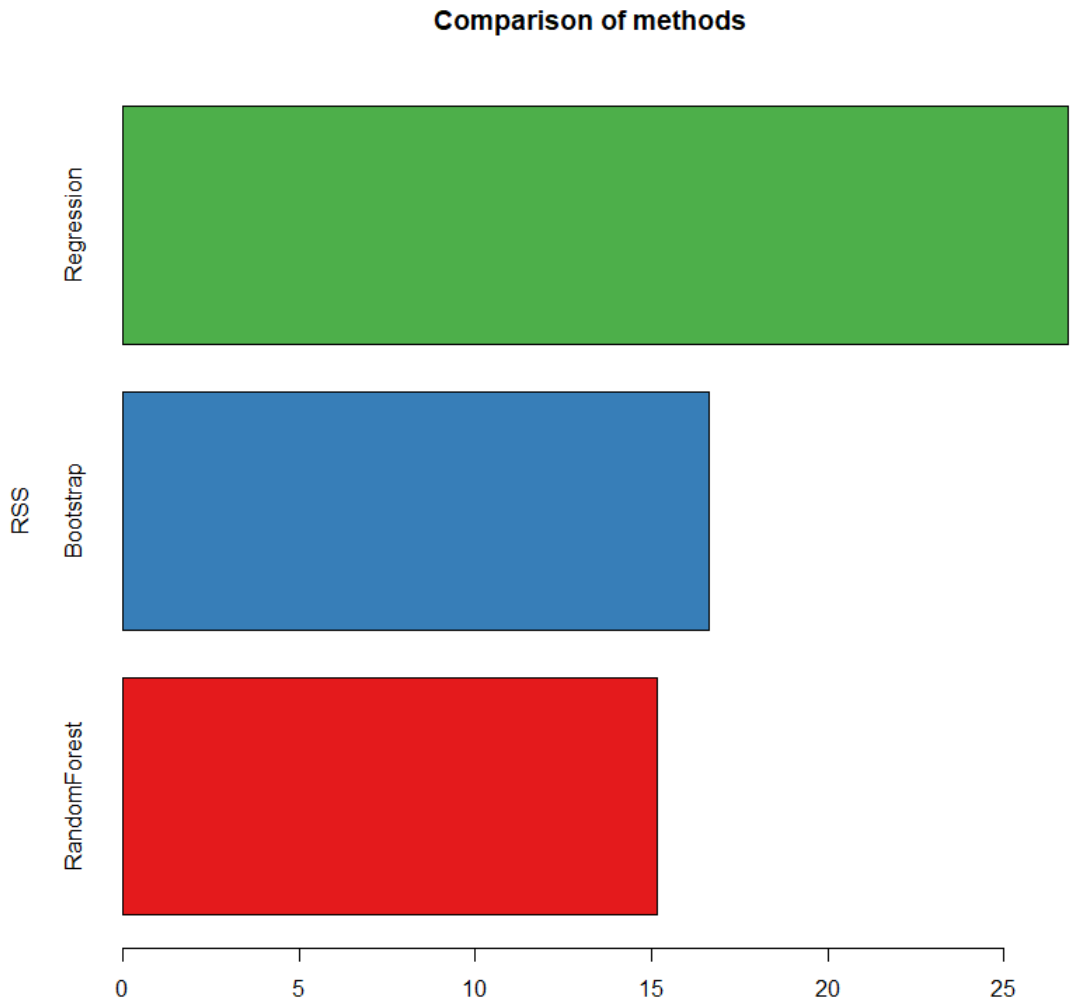


Figure 8: RSS of the three models on the Boston house test dataset.

By implementing decision trees in in R on the Boston house data there is a clear drop in the RSS from implementing bootstrapping and random forests, compared to decion trees. This can be seen in Figure 8.

Using bagging or random forests lead to a new problem. There is no longer an easy way of interpreting the models and explaining how inferences are being made. To attempt to alleviate this some techniques from Explainable AI (XAI) might help.

4 Explainable artificial intelligence

In many fields getting the right answer is not the only thing that matters, sometimes the reasoning behind the answer is equally important. This might be to comply with legal requirements, or simply for the sake of transparency.

To follow the GDPR rules in Europe banks can no longer use automated systems that perform automatic evaluations of a customers credit score unless they are accompanied by a human-understandable explanation. In online chess a player that gets flagged as a cheater and is banned by an automated system might want to know the reason for the ban.

Models can be put into two different categories: explainable models, or black-box model where the explanations are found by post-hoc methods. Models are considered black-box if they are constructed by algorithms on a computer in such a way that no human can understand what has been done or how it works. Post-hoc methods are called model agnostic if they work regardless of what model is implemented. This allows them to explain black-box models to some degree. According to Ribeiro et al. (2016) the advantage of using post-hoc methods on black-box models instead of only using interpretable models is that it allows for more flexible models. Additionally model-agnostic ways of of analyzing the data provides a way to explain different models with the same techniques and representations.

4.1 Partial dependency plots

> The Partial Dependence Plot (PDP) is a rather intuitive and easy-to-understand visualization of the features' impact on the predicted outcome. If the assumptions for the PDP are met, it can show the way a feature impacts an outcome variable. More precisely, mapping the marginal effect of the selected variable(s) uncovers the linear, monotonic or nonlinear relationship between the predicted response and the individual feature variable(s) - Dassen et al. (2020).

This quotation is taken from the introduction to chapter 2.1 in Dassen et al. (2020)

The partial dependency plots were introduced in Friedman (2001) as a model agnostic way of analyzing how the different predictors affect the response variable prediction.

4.1.1 Theory

Let X be the set of predictors, and let X_S be a subset of the predictors that are of interest. Then define X_C to be the complement of X_S in X such that $X_S \cup X_C = X$. A function of X will then typically depend on both of the subsets and $f(X)$, and can be written as: $f(X) = f(X_S, X_C)$. The idea behind partial dependence is to marginalize out the variables that are not of interest. To see how the predictors in X_S affect the predicted response the partial dependence (PD) is defined as:

$$PD = f_S(X_S) = E_{X_C}\left[f(X_S, X_C)\right] = \int f(X_S, X_C) dP(X_C).$$

For black-box models this can be estimated by

$$\hat{f}_S(X_S) = \frac{1}{N}\sum_{i=1}^{n} f(X_S, X_C^{(i)})$$

where $X_C^{(i)}$ are the predictors in C from the i-th observation from the training data.

4.1.2 Disadvantages of PDP

There are some issues about using PDP, which relate to how PDP are made. Suppose the goal is to predict the size of a house given the number of rooms, bathrooms, and garages. Then for the computation of the PDP of rooms the average over the marginal distribution of bathrooms and garages is computed. As the marginal distribution of bathroom might contain values from 1-5, and the number of garages might range from 1-3 this might lead to some very unrealistic numbers. As it would be quite unlikely that an apartment with one room also has five bathrooms and three garages.

Secondarily some effects might be hidden by this approach as marginalized variables might not show the whole picture. If the response variable increases with half of the values for a variable, and the other half decreases the response variable, then the average effect might be null, and the effect

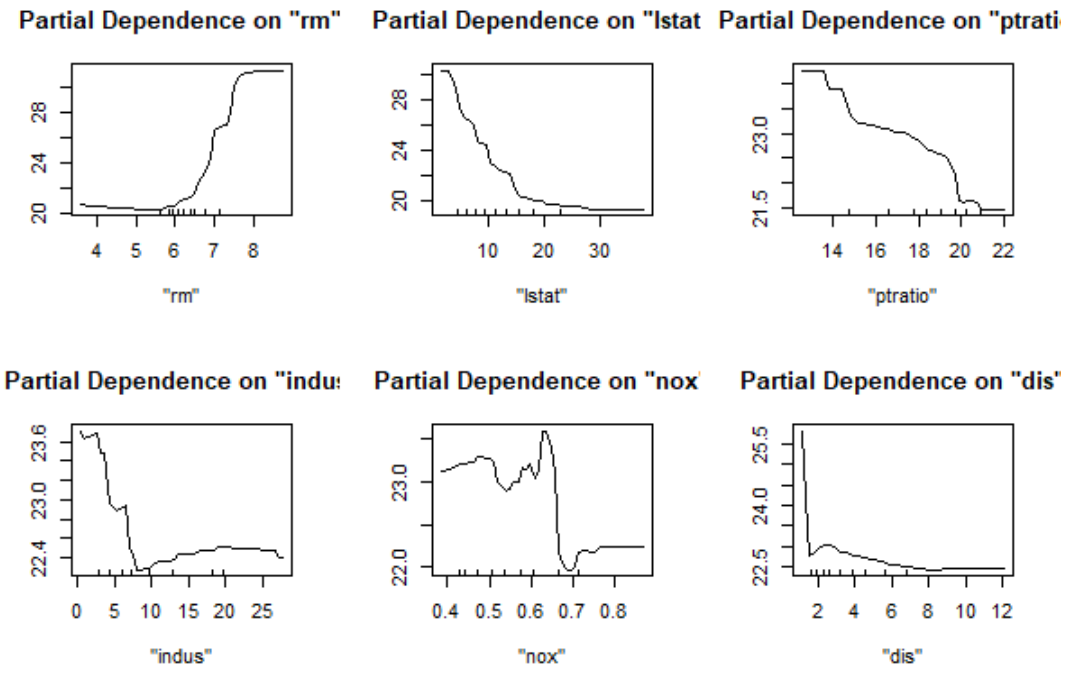


Figure 9: PDP plots of one variable

of the variable is hidden. Goldstein et al. (2014) suggests that Individual Conditional Expectation plots can be used to uncover such heterogeneities, but this topic will not be covered in this thesis.

Furthermore the size of X_S is realistically restricted to only one or two variables, due to difficulties with plotting more than three dimensions. In the case where X_S is uncorrelated with X_C, or at least not strongly correlated, the plot perfectly captures how the average predicted response varies with X_S. This makes it an easy to interpret plot, that can be used to see how variables affect the average response, and to see if there are any noticeable interaction between two variables for the average response.

4.1.3 Application of PDP to the Boston house dataset

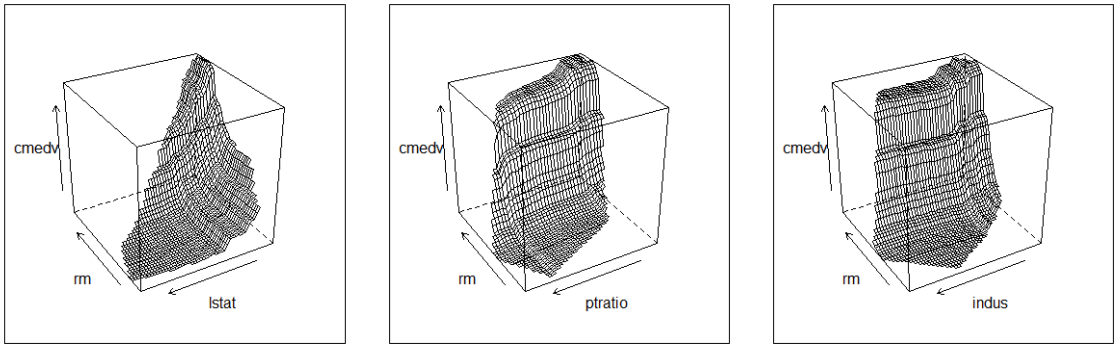
To see how PDP works on the random forest model that was fit previously, the PDP R library is used to plot the PDP plots for some of the variables. First let X_S contain only one predictor to see how the average predicted response is affected by each of the predictors separately. Then PDP with two predictors are constructed to see if it is possible to detect any interaction effects on the predicted response.

In the PDP for average number of rooms per dwelling (rm) it seems that there is a positive relationship to with house prices, while the percentage of lower status of population (lstat) and the pupil-teacher ratio per town have a negative relationship to. The other three plots have some more interesting patterns. The PDP plot for proportion of non-retail business acres per town (indus) decreases until 10, but increases after that.

For the nitric oxides concentration measured in parts per 10 million (nox) the PDP plot drops substantially around 0.7 but is relatively equal for values above, and for values below. This might imply that nox is not an issue until around 0.7, but values above that cause a noticeable decrease in house prices.

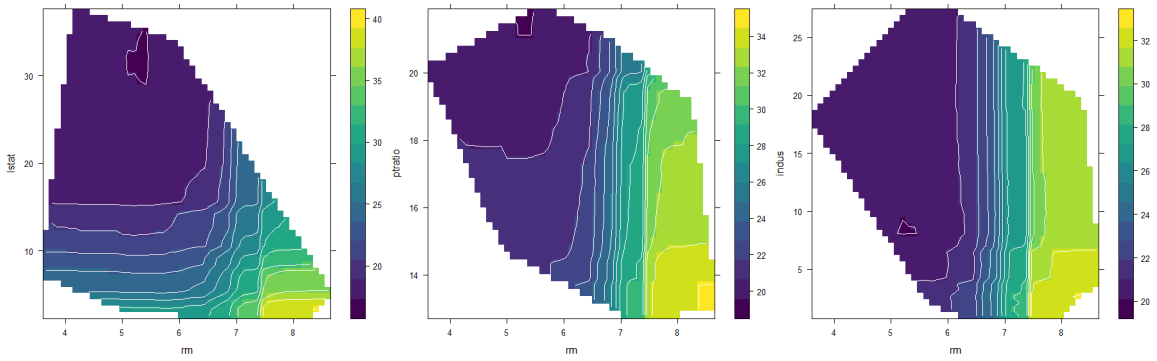
Finally the weighted distances to five Boston employment centres (dis) has a sudden drop at two. This implies that being sufficiently close to the employment centres would increase the price of a house.

To see how PDP can be used to check for effects on the predicted response the PDP of rm with three different predictors are shown in Figure 9 and 10.



(a) $X_S = \{rm, lstat\}$ (b) $X_S = \{rm, ptratio\}$ (c) $X_S = \{rm, indus\}$

Figure 10: 3d-plots of PDP of pairs of variables



(a) $X_S = \{rm, lstat\}$ (b) $X_S\{rm, ptratio\}$ (c) $X_S = \{rm, indus\}$

Figure 11: Contour plots of PDP of pairs of variables

The 3-d plots and contour plots convey the same information in different ways so both can be used. By looking at plot (b) and (c) there seems to be little interaction between the variables. The colors of the contours are mostly the same along the axis in (b) and (c). In (a) it appears to be an interaction between the variables, which can be seen in the sharp increase in the top right corner or the big dark blue contour. This implies that the house price increases more when both lstat is low and rm is high.

4.1.4 Issues with PDP

PDP might have some issues with correlated variables, and that the prediction method does not extrapolate well to weird combinations of covariates. To see if the data is correlated a correlation plot can be used. By using the corrplot package in R the following plot in Figure 12 can be made.

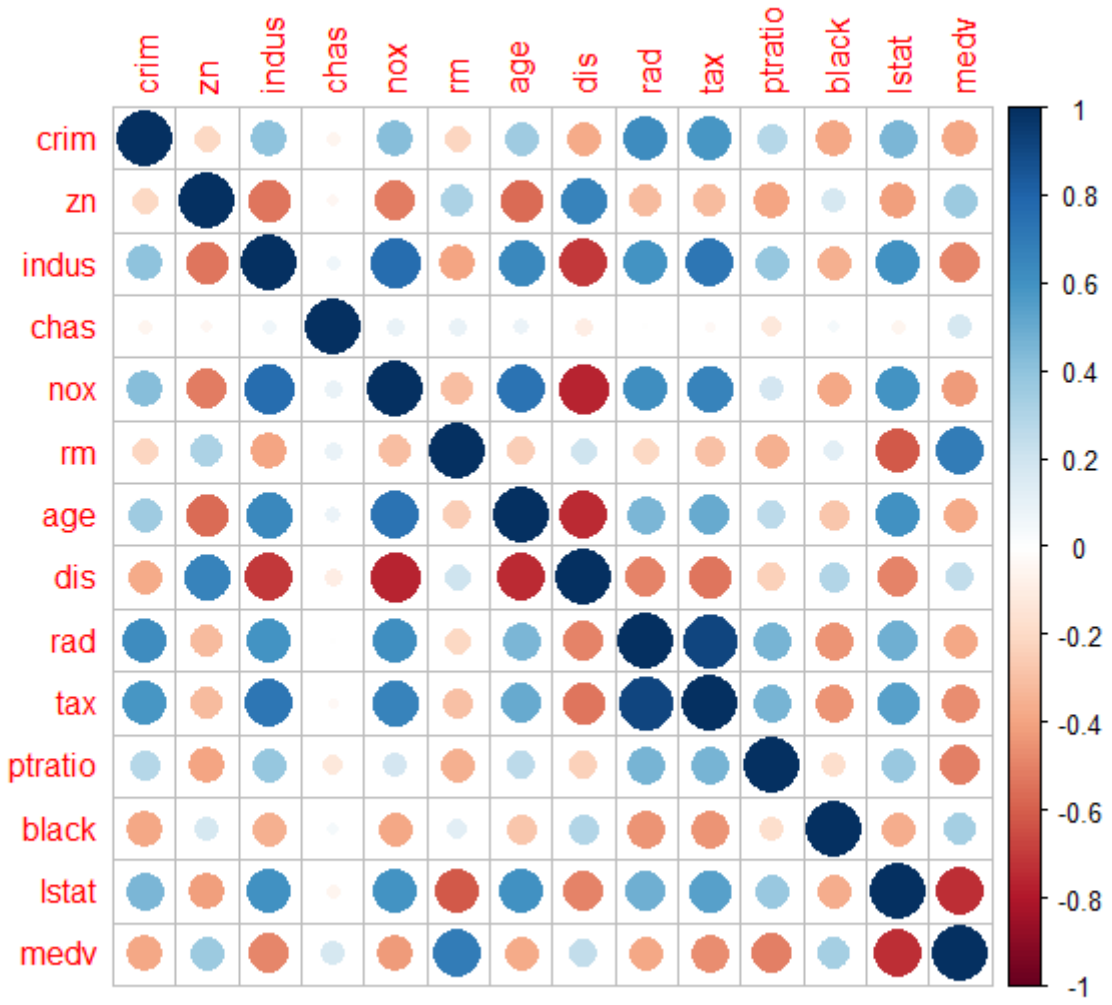


Figure 12: Correlation plot of the whole Boston house dataset

The plot shows the correlation between two predictors as circles, the hue and size represent the size of the correlation. Here positive correlations are color coded as blue, while negative correlations are color coded as red. Note that the correlation of two predictors is symmetric over the main diagonal since the correlation is symmetric i.e. $\text{Corr}(X, Y) = \text{Corr}(Y, X)$. Furthermore any predictor is perfectly correlated with itself, so $\text{Corr}(X, X) = 1$ which means the main diagonal entries are not of interest. By looking at the upper half of the triangle the correlations are not counted twice, which provides a better sense of how correlated the variables are. We check which correlations are above a threshold in absolute value. In other words which of them satisfy $|\text{Corr}(X, Y)| \geq T$ where T is the threshold. Visually this is the same as counting all the circles that have a radius bigger than a specific value.

Threshold	Correlations above threshold
0.5	27
0.7	8
0.9	1

Table 1: Table of correlations above threshold

By looking at Table 1 it is clear that the predictors in the Boston house dataset are quite correlated, having multiple predictors with above 0.5 in correlation, and a few with even higher correlations. As only the correlation in the data get checked this way, this method does not guarantee that the predictors in the model are also correlated. Assuming that the random forest model picks up on this correlation, this still implies that using PDP plots might not be the best choice. If we wish to use PDP to make predictions beyond the envelope of the training data this would require extrapolations (Apley and Zhu, 2016). As random forests are not very suited for extrapolating this would be make the PDP quite unreliable. A potential improvement is using a different way of plotting namely, ALE plots.

4.2 ALE plots

4.2.1 Theory

Accumulated local effect plots of a single variable x_1 is defined as:

$$f_{ale}(x_1) = \int_{x_{min,1}}^{x_1} E\left[f^1(X_1, X_2)|X_1 = z_1\right] dz_1 - constant$$

$$= \int_{x_{min,1}}^{x_1} \int p_{2|1}(x_2|z_1) f^1(z_1, x_2) dx_2 dz_1 - constant,$$

where $f^1 = \frac{\partial f(x_1,x_2)}{x_1}$, but this relies on knowing the underlying function f^1. For black-box models an estimate is used instead:

$$\widehat{f}_{ale}(x_1) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_{i,j} \in N_j(K)} \left[f(z_{k,j}, x_{i,\setminus j}) - f(z_{k-1,j}, x_{i,\setminus j})\right]. \tag{1}$$

To understand this formula it is easiest to deconstruct it step by step. First divide the values x_i can take into K intervals. Then, for an interval calculate the differences $f(z_{k,j}, x_{i,\setminus j}) - f(z_{k-1,j}, x_{i,\setminus j})$ for all the data points. This is done by finding the largest and smallest values of x_i in the interval and then calculate the difference in predictions when x_i is replaced in a data point with the largest value and when it is replaced the smallest value. Denote the maximum value in the j-th interval by $z_{k,j}$, and the minimum by $z_{k-1,j}$, then the notation $f(z_{k,j}, x_{i,\setminus j})$ means that x_i is replaced by the maximum on the interval and the other variables are left unchanged. Similarly $f(z_{k-1,j}, x_{i,\setminus j})$ means that x_i is replaced by the minimum on the interval.

Then, divide by the number of data points used in this calculation to find the average effect:

$$\frac{1}{n_j(k)} \sum_{i:x_{i,j} \in N_j(K)} \left[f(z_{k,j}, x_{i,\setminus j}) - f(z_{k-1,j}, x_{i,\setminus j})\right].$$

This constitutes the effect in one interval, and is the reasoning behind the local effect of Accumulated Local Effect plots.

Finally, accumulate the local effect of all the intervals. This means to take the sum of all the intervals. We have now explained all the necessary steps to arrive at the formula in equation (1). This is almost the formula used to construct ALE plots, but by convention the mean is subtracted for easier interpretation of the plots.

$$ALE(x_1) = f_{ale}(x_1) - E\left[f_{ale}(x_1)\right] \tag{2}$$

When talking about ALE plots the order is the number of variables being plotted. This section has covered the construction of first order ALE plots which are given by equation (2), but ALE plots can be generalized to higher dimensions. The formulas gets more complicated as the intervals need to be generalized to higher-dimensional boxes, and the difference formulas also need to account for

the higher dimensions. As the underlying core ideas are the same in higher dimensions, but the formulas grow in size and complexity they are not included.

For higher order plots it is no longer the main effect being plotted, but rather the interactions between a subset of variables on the predicted response. But, again due to limitation of our ability to interpret higher order plots the usefulness of higher order plots is quite limited to two-three variables.

4.2.2 Advantages and disadvantages of ALE plots

As the original author of ALE plots writes:

> The theoretical advantage of ALE plots is that the construction of these plots should be less affected by correlation between variables. Furthermore they can be constructed in a more computationally efficient way compared to pdp plots. -Apley and Zhu (2016).

One of the disadvantages of ALE plots is that they require a hyperparameter K to be chosen, which needs to be "sufficiently" large, but without a way of knowing what values are sufficiently large this is difficult.

4.2.3 Application of ALE plots to the Boston house dataset

Here ALE plots are made with the R package ALE plots. Surprisingly the ALE plots (Figure 13) look structurally the same as the PDP plots (Figure 9) at least when K grows sufficiently large. One of the theoretical advantages of ALE Plots is that it should work better on correlated data and it is a bit surprising that the plots are very similar.

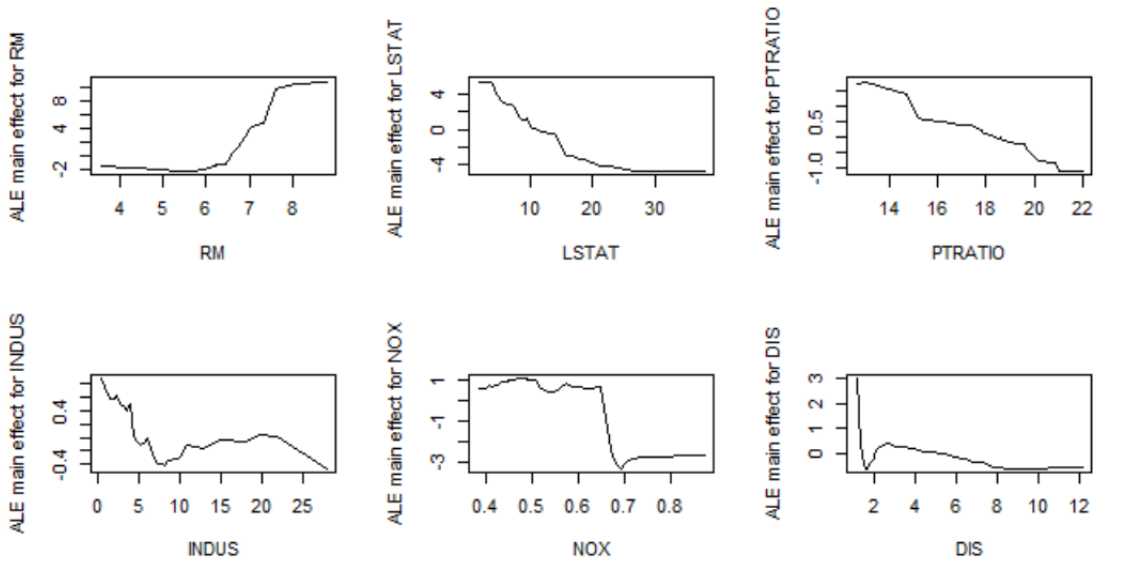


Figure 13: First order ALE plots for the Boston house dataset

Another issue is the choice of K, as the assumption of ALE plots is that K approaches infinity. In the implementation of ALE plots this is practically solved by picking a sufficiently large K. Plotting the ALE plots for K = {10, 20, 30, 40, ..., 90} gives the plots in Figure 14. Apley (2018) suggest that $K = 40$ should be large enough. From the figure it can be observed that the function still changes a bit with larger K's so setting $K = 100$ might be a better idea.

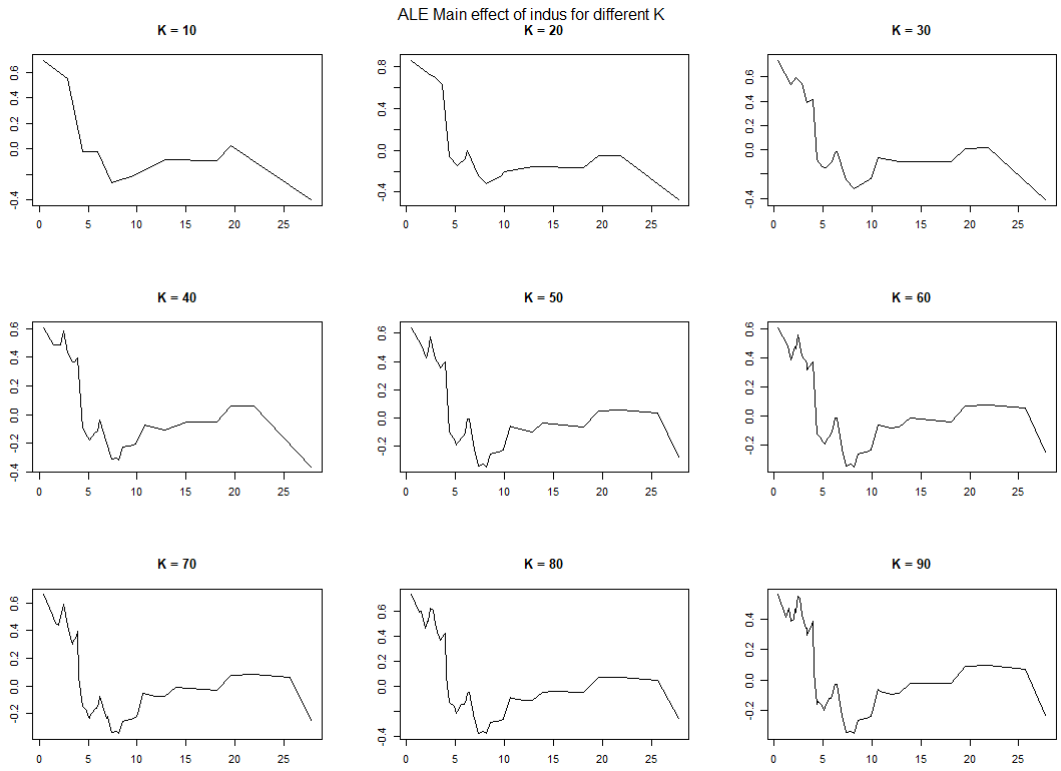


Figure 14: Plots showing the Accumulated Local effect of indus for different values of K.

Regardless of the choice of K there is a trade-off here. By picking a small K the ALE plots get smoother and might not pick up the real complexities, while picking a large K lead to complex and very shaky models. No preferred solution to this issue has been found at the current moment. As the Boston dataset was unable to showcase any of the strengths of ALE plots a simulated example might be enlightening.

4.3 Simulated example to compare PD and ALE plots

As the PD and ALE plots were quite similar for the Boston house dataset, simulated datasets can be used to highlight the advantage of ALE plots over PDP.

Example 1: Let $F(x, y) = xy$. A graph made with GeoGebra [1] is shown in Figure 15. Note that the function is symmetric along the lines $y = x$ and $y = -x$.

[1] https://www.geogebra.org/

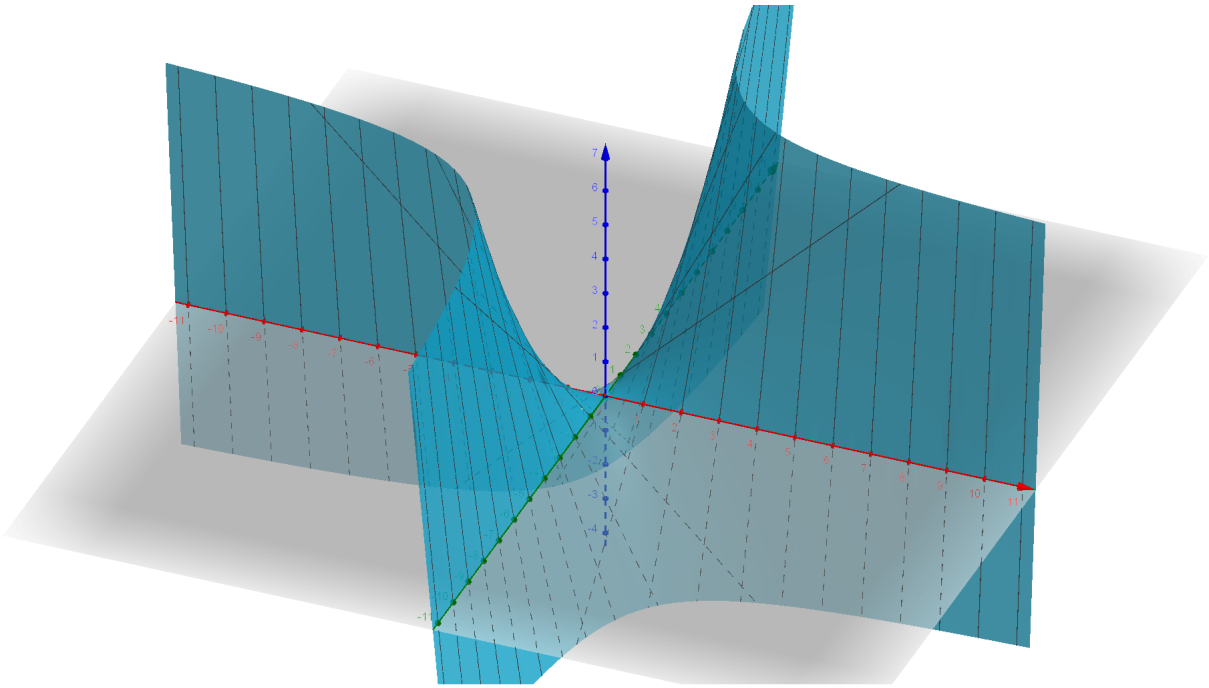


Figure 15: $F(x, y) = xy$

Data is created by calculationg the function $F(x, y)$ along equidistant points in x and y, and then adding normally distributed noise. Using this data a random forest is fit and the partial dependence plots are made, see Figure 15.

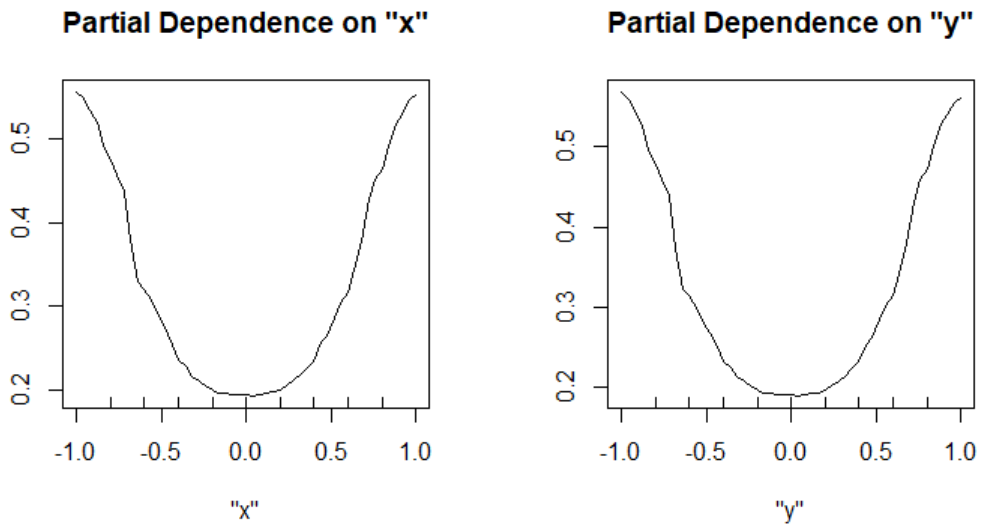


Figure 16: PDP of the random forest model fit to data from $F(x, y)$

The PDP capture identical dependences for x and y, and accurately depicts that both variables affect the outcome equally. Similarly, graphs of the main effect of x and y can be made with ALE plots, which look almost identical, see Figure 16.

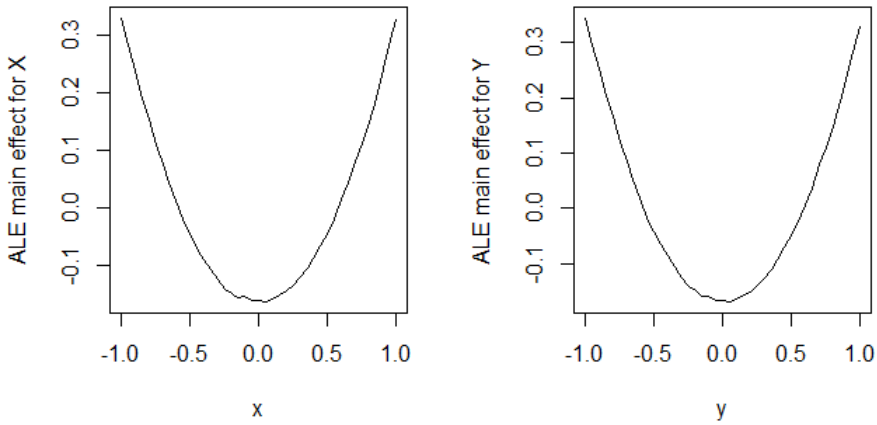


Figure 17: First order ALE plots of the random forest model fit to data from Example 1.

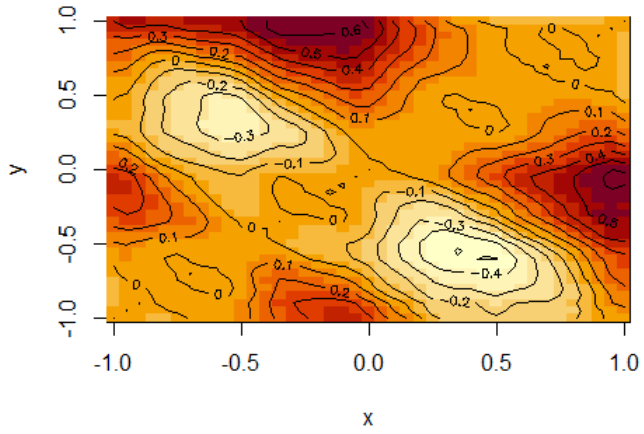


Figure 18: Second-order ALE plot of the joint effect x and y on the predicted response

Plotting the second order ALE plot it is possible to observer the contours of the fitted random forest model. Observe that the contours decrease along the main diagonal and decrease along the off-diagonal, which appears to match the contours of $F(x, y)$.

Example 2: Now to highlight an issue with both of the methods: neither method is invariant under rotations. Use the same function $F(x, y)$ but rotate it 45 degrees clockwise, so the lines of symmetry are along the axis. This is equivalent to the function $G(x, y) = \frac{x^2-y^2}{2}$.

Proof: Apply the rotation matrix on the coordinates of $F(x, y)$.

$$F(x\cos(\frac{\pi}{4}) - y\sin(\frac{\pi}{4}), x\sin(\frac{\pi}{4}) + y\cos(\frac{\pi}{4})) = \frac{x^2 - y^2}{2} = G(x, y).$$

Construct the dataset and fit the random forest in the same way as for Example 1 and plot PDP.

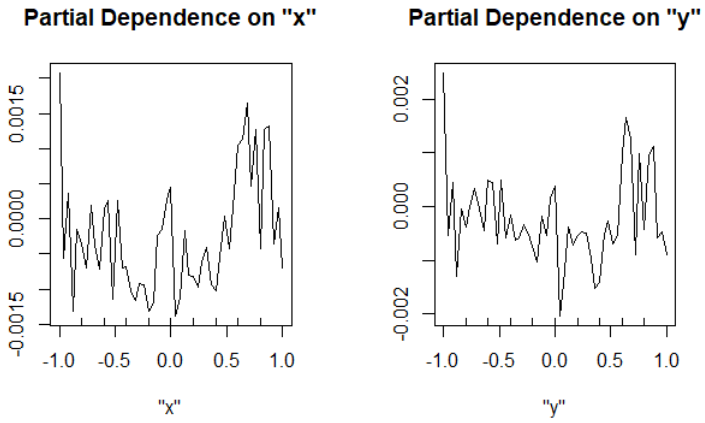


Figure 19: PDP of random forest made with data from $G(x, y)$

Note now that the PDPs are now highly unstable, and the range of the y-axis is smaller by three orders of magnitude.

The same behavior can be observed in the ALE plots of first order in Figure 20.

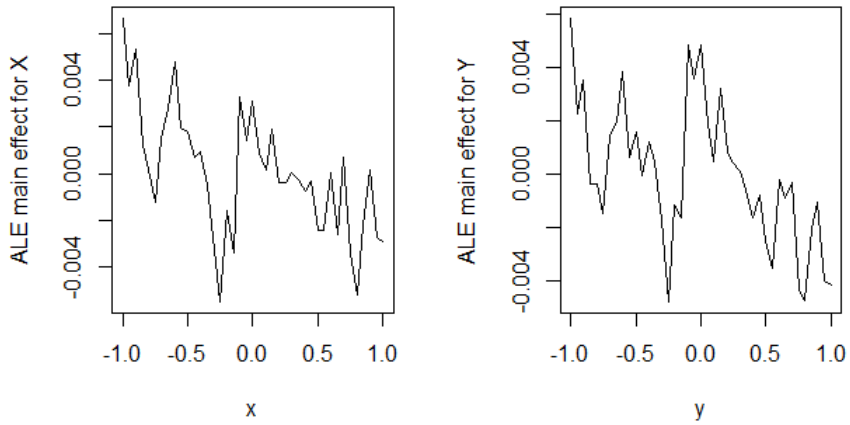


Figure 20: First order ALE plots of the random forest model fit to data from $G(x, y)$

Using second order ALE plots it is possible to get an idea of how the variables interact in the model. It should be similar to the contours for $F(x, y)$ but with the low points and high points changing places. Looking at the ALE plot this is the case, now it's increasing along the main diagonal and decreasing along the off diagonal.

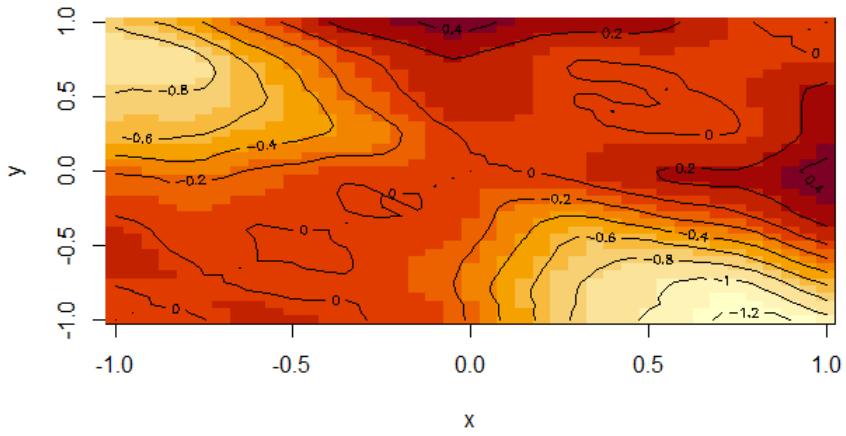


Figure 21: Second order ALE plots of the random forest model fit to data from $G(x, y)$

5 Conclusion

Decision trees have some quite desirable properties, but the naive implementation of decision trees tend to overfit to the data and this causes a bad model fit. Here the fit of the model is measured in terms of residual sum of squares (RSS) on test data. Using this choice of loss function provides a natural way of picking the values for each region in the decision trees.

Improvements to naive decisions trees can be made, and random forests increase the performance of decision trees models in our data analysis. Unfortunately this comes at a cost of interpretability. By using PD and ALE plots this can be remedied to some degree.

After implementing both PD and ALE plots on a random forest model trained on the Boston house dataset it appears that both methods have some merit. ALE plots allow for contour plots of the main effects, which can help identify the interactions between two variables, and produce similar results as PDP with lower computational costs.

Due to the complexity of the way ALE plots are constructed, and the requirement of figuring out what constitutes a sufficiently large value of K, PDP are easier to understand. And despite the theoretical advantage of using ALE plots both methods have performed quite similarly in the implementations tested here, and ALE plots are faster to compute.

From the examples in section 5 it seems like both methods can at least be used to make black-box models a bit more interpretable with different pros and cons. Unfortunately it seems like both methods are not very robust. Looking at the marginal (first order) PD/ALE plots of $F(x, y) = xy$ and the same function rotated 45 degrees provides astonishingly different results. This puts into question the practical usability of these methods on general datasets.

The consequences of this is that neither method can be used to satisfy the requirements for a right of explanation required by GDPR. But in general the methods might be useful tools for analyzing some black-box models.

References

D. Apley. *ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots*, 2018. URL https://CRAN.R-project.org/package=ALEPlot. R package version 1.1.

D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models, 2016.

T. Dassen, N. Hou, and V. Kronseder. *Interpretable Machine Learning, chapter 2.1.* 2020. https://compstat-lmu.github.io/iml_methods_limitations/pdp.html [Accessed 11.04.2021].

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation, 2014.

B. M. Greenwell. pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1):421–436, 2017. doi: 10.32614/RJ-2017-016. URL https://doi.org/10.32614/RJ-2017-016.

D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978. ISSN 0095-0696. doi: https://doi.org/10.1016/0095-0696(78)90006-2. URL https://www.sciencedirect.com/science/article/pii/0095069678900062. [Accessed 26.03.2021].

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009. URL http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

M. Kuhn. Building predictive models in r using the caret package. *Journal of Statistical Software, Articles*, 28(5):1–26, 2008. ISSN 1548-7660. doi: 10.18637/jss.v028.i05. URL https://www.jstatsoft.org/v028/i05.

H. Laurent and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information processing letters*, 5(1):15–17, 1976.

A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL https://CRAN.R-project.org/doc/Rnews/.

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-2.

M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning, 2016.

B. Ripley. *tree: Classification and Regression Trees*, 2019. URL https://CRAN.R-project.org/package=tree. R package version 1.0-40.

B. Schloerke, D. Cook, J. Larmarange, F. Briatte, M. Marbach, E. Thoen, A. Elberg, and J. Crowley. *GGally: Extension to 'ggplot2'*, 2021. URL https://CRAN.R-project.org/package=GGally. R package version 2.1.1.

A. D. Selbst and J. Powles. Meaningful information and the right to explanation. *International Data Privacy Law*, 7(4):233–242, 12 2017. ISSN 2044-3994. doi: 10.1093/idpl/ipx022. URL https://doi.org/10.1093/idpl/ipx022.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/. ISBN 0-387-95457-0.

T. Wei and V. Simko. *R package "corrplot": Visualization of a Correlation Matrix*, 2017. URL https://github.com/taiyun/corrplot. (Version 0.84).

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

Appendix

A R packages used

This is a list of R packages used in the R files, and what they were used for. The format used is:

Name of package: Description of package. What it was used for in this project. Source: where it was taken from

Code can be found at: github.com/RasmusHenninen/Bachelor-thesis

PDP: A general framework for constructing partial dependence (i.e., marginal effect) plots from various types machine learning models in R. Used for making PDP plots. Source: Greenwell (2017)

MASS: Functions and datasets to support Venables and Ripley, "Modern Applied Statistics with S'. Used to get the dataset for Boston housing. Source: Venables and Ripley (2002)

Caret: Misc functions for training and plotting classification and regression models. Used to split the dataset into train/test. Source: Kuhn (2008)

GGplot2: ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Used to make prettier plots. Source: Wickham (2016)

GGally: The R package 'ggplot2' is a plotting system based on the grammar of graphics. 'GGally' extends 'ggplot2' by adding several functions to reduce the complexity of combining geometric objects with transformed data. Some of these functions include a pairwise plot matrix, a two group pairwise plot matrix, a parallel coordinates plot, a survival plot, and several functions to plot networks. Used to make correlograms to present the Boston house dataset. Source: Schloerke et al. (2021)

Tree: Fit a Classification or Regression Tree. Used to fit regression tree. Source: Ripley (2019)

RColorBrewer: The RColorBrewer package offers several color palette for R. This post displays all of them to help you pick the right one. Used to pick colors for graphs. Source: Neuwirth (2014)

RandomForest: Classification and Regression by randomForest. Used to fit randomforests. Source: Liaw and Wiener (2002)

ALEplot: Visualizes the main effects of individual predictor variables and their second-order interaction effects in black-box supervised learning models. The package creates either Accumulated Local Effects (ALE) plots and/or Partial Dependence (PD) plots, given a fitted supervised learning model. Used to make ALEplots. Source: Apley (2018)

corrplot: The corrplot package is a graphical display of a correlation matrix, confidence interval. It also contains some algorithms to do matrix reordering. In addition, corrplot is good at details, including choosing color, text labels, color labels, layout, etc. Used to make a corrplot. Source: Wei and Simko (2017)