Danmarks Tekninske Universitet

Computational Data Analysis

# Case 1

*Authors:*
*Mads Esben Hansen, s173444*
*Anton Ruby Larsen, s174356*

Course number:
02582

March 22, 2020

## Introduction

Case 1 sets the task of creating the best model possible using 100 given data points, where both the observation, x, and the response, y, are known. This model should then be used to predict 1000 unknown responses to 1000 new observations. Together with these new predictions there should also be a measure of uncertainty for these. This task might seem trivial, however, the data is very noisy and contains 100 variables. A large part of our task is therefore to try to get something useful out of all this.

## Model and method

All models we looked at was on the basis that our data contains a lot of noise. We therefore wanted a model that would perform well in these conditions. The models we decided to try were:

**Elastic Net** The *elastic net model* is a combination of the *ridge regression* and the *LASSO regression*. The *ridge regression*, $\beta_{Ridge} = arg \min_{\beta} ||Y - X\beta||_2^2 + \lambda||\beta||_2^2$, regularizes by the $L_2$ norm and the *LASSO regression*, $\beta_{LASSO} = arg \min_{\beta} ||Y - X\beta||_2^2 + \lambda||\beta||_1$, regularizes by the $L_1$ norm. You then choose how much of each regression type you want by a $\alpha$ and a $(1 - \alpha)$. The more *LASSO* the more your model will tend to set features to zero.

**Support vector regression** The main idea behind *support vector regression* is the same as for a *support vector machine*; to minimize error, individualizing the hyper-plane which maximizes the margin, while still keeping in mind that some error is accepted. It works by simply minimizing the *loss* given by some weighted sum of everything that is outside this accepted margin.

**Random Forest** *Random forest* starts the same way as *bagging* by creating $m$ subsets with $n'$ samples from the original data at random with replacement, i.e. *bootstrapping*. From these $m$ subsets we will create m trees. In *random forest* the creation of the trees differ from *bagging* by picking $k$ features at random and the creating a node in the tree by finding the optimal feature of the $k$ chosen features by use of the *gini coefficient* in classification or MSE for regression.

**Gradient Boosting** *Boosting* can be understood as chained *bagging*. Each new tree we create is based on the old tree's error. In *gradient boosting* we start by predicting the average. We then apply a loss functions that creates a measure of how good this average predicts the data and then weigh the data accordingly. Then a tree of a predefined depth is created to predict the error made by the previous. The predicted value of the new tree is penalized by a learning rate as we know from the *gradient descent algorithm*. The loss function is then applied to reweigh the data and the procedure is repeated until some stopping criteria is reached.

### Model selection

When selecting the best model our approach was quite simple: Discuss which models could theoretically work, make a quick implementation of the given model, train it on 80 of the 100 data points, and predict the remaining 20 data points. We could then calculate the MSE of these predictions, which could then give us an idea whether it would be worth our time to spend more time on the given model. We then chose the models that showed promising preliminary results and tested them further. We then did cross validation on the models in order to determine e.g. number of trees, $\alpha$-coefficient, etc. of course depending on the model. We did so by again leaving a number of data points out, chosen at random, and testing which value in a reasonable interval (the value and interval depends on the model). In the end, this enabled us to choose a good model with good parameters.

### Missing values

We looked into different approaches to handle missing values. We thought about simply ignoring the measurements that contained missing values; however, since they made up a quite significant part of our categorical data we decided this was not a liable option. We therefore wanted a method that could estimate missing data-points based on mixed data. We found *Multivariate Imputation by Chained Equations (MICE)* which is a parametric test, and *Nonparametric Missing Value Imputation using Random Forest (missForest)*. We did not know if an assumption of normality noise was reasonable when having this much, and therefore we excluded the parametric test MICE and went with the non-parametric test missForest. Furthermore we have also learned in class that *random forest* in general do well with noisy data.

### Factor handling

Before we could train our models we needed to get rid of non-numerical data. We chose to model the categorical data as dummy variables, in particular we did so using *one-hot encoding* by applying the *onehot*-library.

### Feature selection

After having addressed he problem of having a incomplete and non numerical data, we need to handle the problem of having way to many features. When having 100 features and only 100 samples our models could fall victim to the curse of dimensionality. We therefore ran three tests to select only the most important features to proceed with.
Firstly we performed a backward selection using the *random forest* algorithm as model. This reveled that the set {x14, x34, x46, x24, x66} was the most important.

Then we calculated the correlation between all features and the response variable y. This tells us all linear relations and reveled that x14, x34 and x24 - that also showed up in the backward selection - was the most correlated with y. The last test we conducted was an importance measure in a random forest. This importance measure is based on two measures. The first is the normalized average difference between the prediction error for OOB data for each tree and the same thing after permuting each feature. The second measure is the average decrease in node impurity measured by sum of squared residuals. This test showed that feature x46, x14, x34 and x24 was important.
Because the features {x14, x34, x46, x24} showed up in all the tests we chose to proceed with those.

## Results

We tested quite a few models, out of these elastic net and random forest showed the best preliminary results. For the elastic net we found that a model with a fairly large $\alpha$-value produced the best results i.e. *lasso* generalized fairly well. For the random forest, we found that the number of trees did not matter that much, as long as it was above 200. After some further testing we found that our models were almost only using 4 variables, namely $x_{14}$, $x_{34}$, $x_{46}$, and $x_{24}$. We had issue with noise in the data and we therefore tried only using these 4 variables. We here saw a significant improvement in our results. In the end we found that a random forest, based on these 4 variables, with 15 trees and 2 variables in each tree gave us the best and most consistent results; this would therefore be our final model. We found the estimated prediction error as:

$$\sqrt{MSPE} = 4.65$$

Which means that the confidence intervals for our predictions will be:

$$y_{95\%} = \hat{y} \pm 1.984 \cdot \sqrt{4.65} \Rightarrow$$
$$y_{95\%} = \hat{y} \pm 4.28$$

## Model validation

As described the *random forest* model was our best model. In order to give an estimate of the prediction error for the 1000 predicted y-values we will use a R library called rfinterval. The implementation of this library is described here[1]. When implementing a *random forest* the algorithm will bootstrap a set of samples for every tree. This means that a given sample will not be present in $m$ trees, also called out of bag(OOB). More specifically approximately $(\frac{n-1}{n})^n \approx exp(-1) \approx 0.368$ trees will not contain the $i^{th}$ sample. We now use this fact to predict $y_i$ with all trees not containing $x_i$.

$$\text{Mean Square Prediction Error} = \text{MSPE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{x_i \notin RF})$$

Where the notation $\hat{y}_{x_i \notin RF}$ means the prediction of $y_i$ made with all tree not containing $x_i$. This is also called an OOB prediction.
We now assume the OOB prediction errors are normaly distributed and symmetric and calculate the prediction interval by

$$\hat{y} \pm t_{\alpha/2}(n-1) \cdot \sqrt{MSPE}$$

[1] Haozhe Zhang, Joshua Zimmerman, Dan Nettleton, and Dan Nordman. (2019). "Random ForestPrediction Intervals." The American Statistician. Doi: 10.1080/00031305.2019.1585288.