Sensitivity Analysis of Features and Methods in Crowd-Sensing Based Road Condition Estimation

Nisse Knudsen
Student
Information Engineering & Management
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: ujest@student.kit.edu

This paper applies a three dimensional sensitivity analysis to the problem of predicting the International Roughness Index (IRI) by using sensor data from consumer smartphones. The first part is the feature extraction based on z-axis acceleration responses through road profile wavelengths between 2.4 m *and* 15 m*, the second part is an intelligent feature selection algorithm to reduce the number of features independent of the used training method, and the third part about which training methods and tuning parameters perform best. The analysis showed that using a Fast Fourier Transformation no significantly useful features could be extract. The feature selection algorithm performed well, but possibly excluded global maxima by using a higher step size. The best method for training a prediction model was found in the Random Forest Regression which performed equally well even when varying tuning parameters.*

1 Introduction

Today, most of the street conditions are assessed using vehicles with special equipment as laser sensors. These vehicles are expensive which results in high cost and low frequency in street monitoring. Using a crowd based sensing approach could enable public infrastructure agencies to monitor streets in near real time. Almost every car driver is equipped with a smartphone which has a variety of sensing capabilities, especially 3-axis accelerometer, 3-axis gyroscope and GPS sensor. These data can be captured and used to estimate the current road condition, reducing or removing the need for specially equipped vehicles and covering streets from large highways to small countryside streets. The captured data can vary by the position of the phone and its sensors, the speed of the car and the low rate of GPS fixes. A fitting model need to adapt these condition, e.g. using GPS interpolation or speed dependent feature extraction.

Several studies on how to best predict road conditions or roughness have been done, as mentioned in 2, but few focused on the prediction of the International Roughness Index as a continuous predictor. Additionally, through the described sensing capabilities in modern smartphones, recorded data are high dimensional by features / variables and samples. To simply use all data in their full complexity and train an arbitrary model, can result in varying performance results, but will most likely not lead to satisfying results. Therefore, a more structured approach across three dimensions (Feature Extraction, Feature Selection, Method Selection) is required. This work tries to include such an intelligent approach, and to answer the following research questions:

- What is a fitting (a) extraction and (b) selection of feature to predict the IRI?
- What is a fitting method to predict the IRI using the selected features?

Both the feature selection and the method selection are interconnected, because some methods consider some features as relevant, while other methods consider other features. As a result, both questions can't be answered separately and need to be considered as connected during the solution approach in this work.

2 Related Work

The majority of publications, which focused on using sensors connected to or built in smartphones, used the captured data to correctly classify anomalies, e.g. bumps, potholes etc. While the first studies from [1] and [2] used a binary classification, the following study from [3] could classify anomalies in two types (small pothole or large pothole). The classification approach for the first two studies consisted of simple decision trees, while [3] used a support vector machine.

The International Roughness Index (IRI) was introduced by the World Bank in 1986 ([4]) and is one of the most used and standardized metrics for quantifying road roughness. Sayers describes in his book "The little book of profiling" [5] how the IRI is affected by sinusoid waves of certain wavelengths. Figure 1 shows how waves with certain cycles per meter have more impact on the IRI than other waves. Especially waves with 0.065 cycles/m to 0.42 cycles/m have a high factor gain. These cycles can be converted into a wavelength from 15 m to 2.4 m. Waves within this range will be especially considered in the 3.1 for feature extraction.

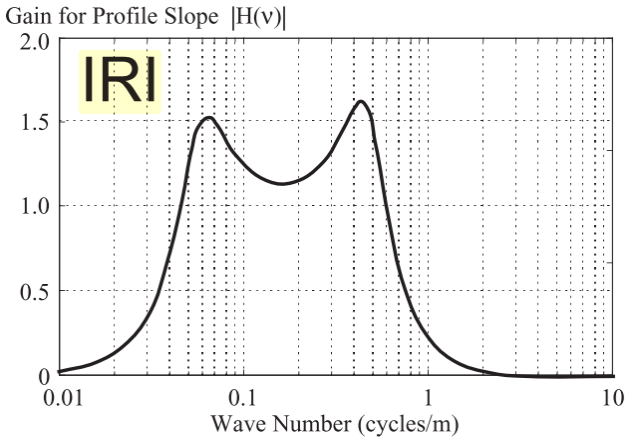

Fig. 1. Factor impact of sinusoid waves with specific cycles/m ([5])

3 Method

For this work a 3-dimensional sensitivity analysis was be run:

1. Feature Extraction
2. Model Selection
3. Feature Selection

After capturing the raw data from smartphone sensors, useful features were extracted. From these features the most important ones are being selected to improve model speed and determination. The feature selection depends on the selected model, since we are using embedded feature selection.

3.1 Feature Extraction

Three sensors are used for data recording, which are all built in by standard in today's smartphones.

- 3-axis accelerometer
- 3-axis gyroscope
- GPS

By substracting the constant gravitional acceleration from the 3-axis accelerometer, an addtional virtual 3-axis linear accelerometer is created. The recorded GPS fixes are matched to a map of streets and translated into velocity information. For all 4 sensors standard aggregation functioned are applied, which are

- mean
- range
- standard deviation
- variance
- root mean square

All aggregation functions are calculated for 100m road segments. Road segments have an offset of 20 m, which creates an overlap of 80m between consecutive segments.

Derived from information about longitudinal wavelengths and their impact on ride quality ([6]) and the calculation of IRI ([5]), the most impacting wavelengths are between 2.4 m and 15 m. These wavelengths can be used to calculate an acceleration frequency in z-axis, dependent of the vehicles speed. For this acceleration frequency the magnitude and other aggregation functions on the acceleration extract features for every segment. To have 5 equidistant wavelengths, we take the minimum wavelength as 2.5 m and the maximum 15 m. This results in considered wavelenghts of 2.5 m, 5 m, 7.5 m, 10 m, 12.5 m and 15 m. Assuming an example average vehicle speed of 20 $\mathrm{m\,s^{-1}}$ (70 $\mathrm{km\,h^{-1}}$) a frequency from 8 Hz to 1.33 Hz is created. For not violating the Nyquist sampling theorem we need a sampling frequency larger than double of the max frequency, i.e. 16 Hz ([5]).

By converting the z-axis acceleration from the time domain to the frequency domain using Fast Fourier Transformation, the amplitude for each frequency is being calculated. As mentioned above, the frequency is calculated dependent on the average speed in that segment. Due to variances in the actual speed of the car during one segment, the frequency itself created by the road

roughness wavelengths can also vary. Therefore, instead of taking the amplitude of the frequency calculated using the mean speed, the mean of the amplitude of n frequency steps to the left and right of the calculated frequency is used as a feature.
Using the *seewave* package in R([7]), for every road segment the z-axis accelerometer data is anaylzed. The package itself is mainly focused on audio signal processing, but the basic features suffice the purpose of extracting the amplitude as a feature.
To find the offset from the mean frequency, an iterative approach was chosen: An offset between 0 and 12 steps to the left and to the right of the calculated frequency is used to calculate a mean amplitude. After every offset increase, the mean amplitude (repectively for every wavelength) is used as a new feature in a random forest model. 1 step equals 0.2 Hz difference in frequency. Using the feature selection approach described in 3.2, all of the new features survive a certain number of rounds, until they are removed by being amongst the least important features. That number is used as a metric which offset creates the most meaningful or lasting features for prediction.
As seen in Figure 2, the features created using wavelengths of 2.5 m (orange) and 5 m (gold) are unmeaningful, independent of the mean range. Also, when using a range of 0 or ± 1 steps, all of the features survive barely any rounds of feature selection. The other features show no clear trend towards a certain mean range, and vary a lot. Yet, summarizing the score for every wavelength feature, the highest score is aggregated when using a mean range of ± 8 steps. Therefore, a mean range of ± 8 steps is chosen to define the features that are used in training models.
Even though there is no clear trend visible, the approach describe above provides a way to not completely arbitrarely chose a range for mean amplitudes, and also to circumvent too specific frequencies, which create unimportant features, as shown above. As mentioned in 4.1, the used data set is based on the work done by [8]. It includes a continuous wavelet transformation (CWT) for 5 bands as an additional aggregation function.
An overview of all aggregation functions for each sensor can be viewed in Table 1.

3.2 Feature Selection

The idea behind feature selection is to determine which features can be discarded without negatively impacting performance of the trained model, and which to retain because of their value to predict the outcome variable.
One method is the Best-Subset Feature Selection. Generally, it is a brute-force algorithm which simply creates all possible subsets of features to determine which creates the best performing model. While this works for very small sets of features, it becomes increasingly complex with larger numbers of features. There are algorithms to intelligently find the optimal subset, yet still only computationally feasible for maximum $p = 30$ to 40 features ([9] Section 3.3.1), where p is the number of features in a data set.
Another method is Forward Feature Selection, which starts with a linear model containing only the intercept. Then a first feature is added, namely the one which creates the best fit. Afterwards one additional feature is added with every step until the set of best features have been determined. This approach can be used even when $p \gg N$, where N is the number of samples in a data set ([9] Section 3.3.2). A disadvantage of Forward Feature Selection is that it creates nest subsets, which can not discard any features once they have been added to the subset.
The other way round works Backward Feature Selection, which starts using all features and iteratively removes the least significant feature using the z-score ([9] Section 3.3.2). It has the similar disadvantage of nested subsets as Forward Feature Selection, but in this case that features can not be retained or added once they got removed.
Besides explicitally discarding or retaining features, there is the possibility to use shrinkage methods. By shrinking the coefficient of features, unimportant ones can be weighted less than others. [9] describes the former approach as discrete and the latter as continuous. Variables get a penalty on large coefficients, so the minimization function tries to reduce coefficients as much as possible.

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{1}$$

A more intuitive way to look at shrinkage is to understand it as a condition that the minimization function is subjected to:

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \quad \text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t^2 \tag{2}$$

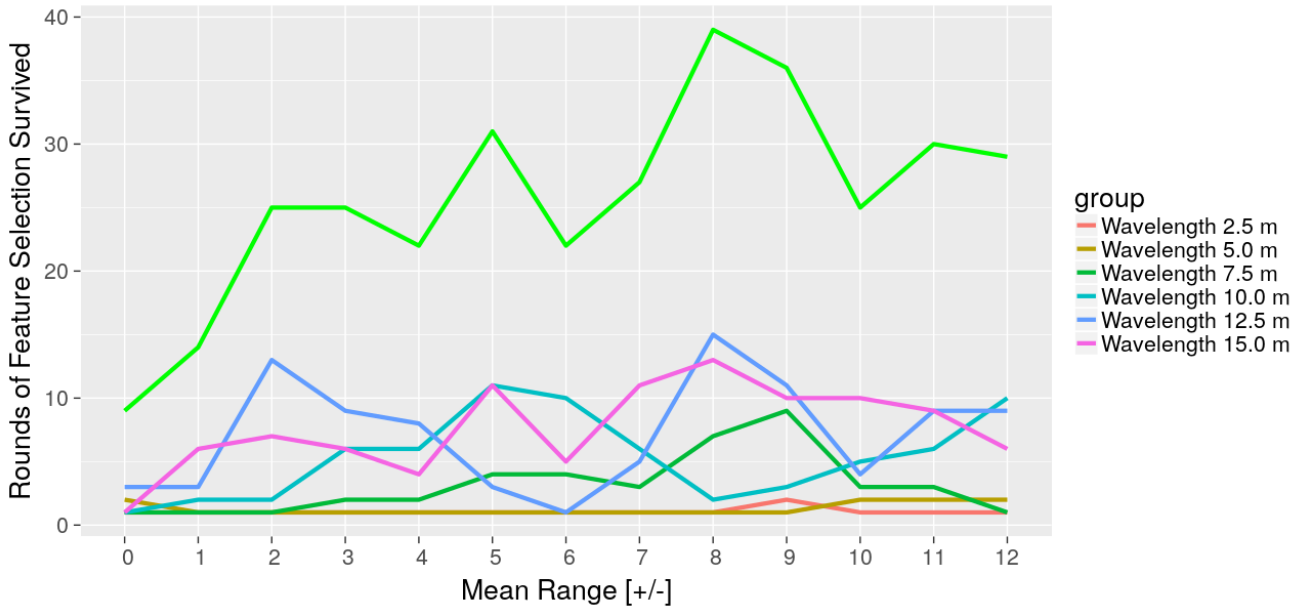

Fig. 2. Rounds of feature selection survived using mean ranges from 0 to ± 12 steps for 6 wavelengths; light green: summarized scores

Table 1. Aggregation functions for sensors

Sensor	**Aggregation Functions**
3-axis Accelerometer	mean, range, sd, var., rms, CWT for 5 bands, mean z-axis amplitude for 6 wavelengths
Linear 3-axis Accelerometer	mean, range, sd, var., rms, CWT for 5 bands
3-axis Gyroscope	mean, range, sd, var., rms, CWT for 5 bands
GPS	mean, range, sd, var., rms

Equation 1 and 2 are used in Ridge regression. It uses the L^2-Norm, which uses the euclidean distance to constrain the beta parameters. This shrinkage method is also used in Lasso regression, which is explained in 3.3.1, but having the L^1-Norm as constraint. Machine learning methods that have shrinkage built into their training do not require any additional feature selection method. The penalty coefficient λ (respectively the boundary t) is the only parameter that need to be changed to optimize the performance of a model.

Looking at the data set which was used in this work and the number of aggregation functions (see Table 1), a space with $p = 100$ features has been created. This high dimension already excludes the application of Best-Subset Feature Selection, even with intelligent methods. Furthermore Forward Feature Selection was discarded, because all variables should be used at least once for training to see any influences that could be created by highly collinear features, like they exist in the used data set. This requirement would be fulfilled by Backward Feature Selection, which starts with all features and removes iteratively one by one, as described above. Yet, part of this work was to train and validate several models using different methods and tuning parameters, and for this a fine-grained approach as Backward Feature Selection takes a long time.
To keep the general idea behind Backward Feature Selection, but increase the execution speed, a new feature selection algorithm was implemented.

3.2.1 Algorithm

Using the variable importance for each model (see 3.2.2, a feature selection wrapper method was written. Algorithm 1 in combination with Algorithm 2 provide an overview of the procedure using pseudo code.
First the total number of features in the data set is used for training the model. The resulting scaled variable importance is then ranked descending from best feature to worst feature. All features that were included in that training round are noted down and their counter for *number of rounds survived* is increased by 1. The process is repeated, but every time the M worst fea-

tures from the last run are removed, until only 1 variable is left for model training. Afterwards the *number of rounds survived* counter hints towards the most useful variables for a model, but - compared across different methods and tuning parameters - can also show model independent useful variables. After every training round the determination coefficient R^2 is saved, to also find the model with the best performance and the least variables.

The advantages of this feature selection wrapper is that it works with any training method, as long as the variable importance can be determined. Additionally, the wrapper itself has linear runtime, because it always removes a constant number of M worst variables. Instead of using the initial variable importance ranking repeatedly as a base for removing the worst M features, the recalculation of variable importance after every training round enables certain variables to increase their importance after a colinear variable has been removed.

A disadvantage is the number M if chosen larger than 1. This can prohibit the method to find a global maximum of R^2, and a non-optimal set of features might be selected as base for training a model.

Algorithm 1 Model Training Algorithm

```
1:  procedure MODELTRAINING
2:      allTrainedM ← ∅
3:      trainData ← read training partition
4:      trainConfig ← read config file
5:      mt ← trainConfig.method
6:      tune ← trainConfig.tuneGrid
7:      nextF ← all features of trainData
8:  loop:
9:      while !is.empty(nextF) do
10:         trainedM ← train(trainData,nextF,mt,tune)
11:         allTrainedM ← allTrainedM ∪ trainedM
12:         nextF ← SelectFeatures(trainedM)
13:     return allTrainedM.
```

3.2.2 Variable Importance

Variable importance can be determined by either using model specific or general staistical information. For some methods, especially trees (and thereby forests), model information are available and can be used.

Algorithm 2 Feature Selection Algorithm

```
1:  procedure SELECTFEATURES
2:      model ← last trained model
3:      prevF ← model.features
4:      nextF ← ∅
5:      p ← length of prevF
6:      omitM ← 5
7:  prepare:
8:      if p == 1 then return ∅
9:      sortedF ← sort(prevF,by=VarImp,desc)
10:     if p ≤ omitM then return sortedF[1]
11:     i ← 1
12: loop:
13:     while i < (p − omitM) do
14:         nextF ← nextF ∪ sortedF[i]
15:         i ← i + 1.
16:     return nextF.
```

3.2.2.1 Linear Methods

For all linear methods the absolute t-value of each variable is used as the importance metric.

3.2.2.2 Random Forests

For random forests the permutation importance is used as a metric. For every tree the mean squared error (MSE) is calculated. Afterwards, a variable is permutated with another one, and the MSE is calculated again. The resulting difference is averaged over all trees in the random forest, and at last normalized by the standard error. Another variable importance metric for random forests is the Gini Importance, which is based on the information gain that a variable creates at each split. This metric is not used in this work, though.

3.2.2.3 Neural Network

Neural Network Regression uses the weights that every input node has connected to all hidden layers. The idea is that important variables have higher weights and thereby higher impact on the model.

3.3 Method Selection

The selection of training methods included linear methods, random forests and single-hidden-layer neural network. All of these methods should be used iteratively by applying the feature selection algorithm described in 3.2.1 and using differen model specific tuning parameters.

3.3.1 Linear Method

The first and most simple linear method selected was Ordinary Least Squares (OLS). "The linear model either assumes that the regression function $E(Y|X)$ is linear, or that the linear model is a reasonable approximation" ([9] p. 44). By doing initial data sighting nearly all of the variables in the data set revealed a quasi linear relationship between them and the output variable. Therefore using a linear regression model should result in a good fit.

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{3}$$

As shown in Equation 3 a prediction output is created by applying trained coefficients β_j to the input variables.

The second linear method that got selected was the *least absolute shrinkage and selection operator* regression, or shorter *lasso* regression. The lasso regression is similar to the Ridge regression, which was introduced in 3.2, but with a penalty / constraint on the L^1-Norm of the β coefficients, as in the constraint in Equation 4. It is important to note that only feature coefficients are penalized, not the intercept coefficient β_0.

$$\begin{aligned} \hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t \end{aligned} \tag{4}$$

By using the L^1-Norm in contrast to the L^2-Norm, an interesting behaviour is created: Instead of shrinking single coefficients to continuous and possibly very small values, coefficients (and thereby features) can be completely set to zero / discarded. The reason for this behaviour is shown in Figure 3. The red contours are the results from the least squares error function. By constraining the error to the blue area, the β coefficients are selected as small as possible. Because of the diamond shape of the L^1-Norm in lasso regression, the contours are very likely to touch one of the corners first, rather than the side. In contrast to that the ridge regression on the right shows that setting a coefficient to zero is very unlikely.

This behaviour makes the lasso regression a very interesting method for training on the data set with highly correlated features. Although the feature selection process is still applied iteratively to the model, it is expected that the first model with $p = 100$ features performs best, because of the internal lasso feature selection. The size of the constraint t is usually expressed as the relative measure $s = \frac{t}{t_0}$, where $t_0 = \sum_1^p |\hat{\beta}_j^{\,ls}|$. $\hat{\beta}_j^{\,ls}$ are the estimated residuals from the OLS regression. If s is set to a value > 1, the lasso solution is the OLS solution.

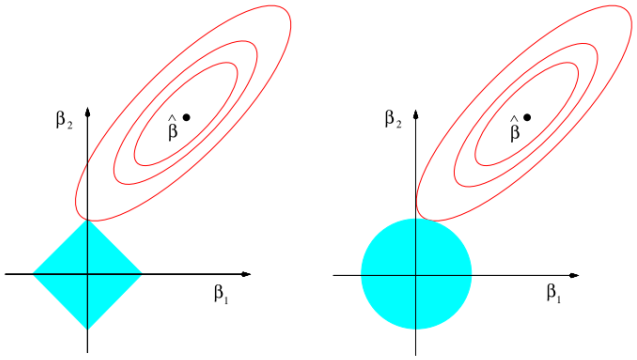

Fig. 3. lasso (left) and ridge (right) regression. Blue area is the constraint area, red ellipses are the contours of the least squares error function ([9] p. 71)

3.3.2 Random Forest Regression

Random Forest Regression is basically an ensemble of B decision trees that have been trained using a bootstrap sample of size N from the training data. For training the model, m out of p features are randomly selected and used for splits in the tree. Although decision trees themselves are used to classify data, the average of all classification results $T_b(x)$ of all trees can be used to predict continuous values as it is done in regression and shown in Equation 5 ([9] p. 588).

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{5}$$

The selection of m is crucial, because it results in the probability that a relevant feature is being selected amongst other irrelevant features. [9] shows that in a scenario with 6 relevant and 100 irrelevant features in the data set and $m = 10$ random splits, the probability to select a relevant feature at a split is already 0.46.

Additionally, if the depth of trees in the forest is controlled and they are not full-grown, the Random Forest Regression is quite robust against overfitting ([9] p. 596).

One disadvantage which could apply to the prediction of continuous IRI values is Random Forest Regression characteristic with untrained values at the start and end

of the prediction range. As an example, if a simple linear model is trained on certain input variables and IRI values with a range from 0-7 m/km as output, it can simply predict an output of 8 m/km if the input variables increase also linearly. In contrast, the trees within the Random Forest only classify values which they were trained on. Consequently, if a prediction should be made based on data that should result in an IRI prediction of 8 m/km, the forest will return at maximum the average over the largest trained outcome, i.e. 7 m/km. Using cross-validation during training this problem could become visible in the performance metrics. Yet, a training set including the full range of outcome values would be beneficial.

3.3.3 Neural Network

A Neural Network can be designed in several ways, but for this analysis the most simple Neural Network will be used: single hidden layer neural network. It consists of p input neurons, M hidden neurons and for regression of 1 output neuron. Every input feature from the data set is represented as 1 input neuron. Neurons between the input and hidden layer are interconnected with weight functions, as well as neurons between the hidden and the output layers. When training a neural network, a error function is used to quantify the error between prediction and expected outcome. This error is propagated back through the network and the weights between neurons are updated accordingly.

4 Evaluation

To evaluate all selected methods and compare their performance, the statistics programming language R was used. Through a variety of packages provided by the community, the implementation was straightforward. Especially the *caret* package by Max Kuhn made it simple to train model using a variety of methods and tuning parameters.

4.1 Data Collection

The data used for this work were provided by Laubis, and is based on his work [8].
7 drives with a regular passenger vehicle were executed on the German district road K3535. Each drive was 4.56km long, while half of it was driving 2.28km west to east, and the other half 2.28km east to west on the same road. Afterwards the data were matched onto a road map, missing values interpolated and matched with the IRI ground truth, provided by the Institute of Highway and Railroad Engineering (ISE) at the Karlsruhe Institute of Technology.
An example drive from east to west has been visualized in Figure 4. If the measured IRI on a road segment was high, it shows a deep red as color, while if the IRI was low, the color is more white.

Fig. 4. A east to west drive on the K3535. Red indicates higher IRI, white indicates lower IRI.

4.2 Interpretation per Model

4.2.1 Linear Methods

There were models for three linear methods trained:

- Ordinary Least Squares Regression (OLS)
- Lasso Regression with...
  - ... $s = 0.1$ (Lasso01)
  - ... $s = 0.3$ (Lasso03)
  - ... $s = 0.5$ (Lasso05)
  - ... $s = 0.7$ (Lasso07)
  - ... $s = 0.9$ (Lasso09)
  - ... $s = 1.0$ (Lasso10)

The best model for each method with performance metrics are shown in Table 2. The OLS regression performs best with 55 best features, but between 45 and 75 features it is not much worse, if R^2 is used as the performance metric. Compared with results from other methods, an $R^2 = 0.7895107$ is quite good performance, which emphasizes that the relationship between the input features and the predictor IRI could be a linear relationship. Observing the 6 lasso regressions in Fig-

ure 5, the models using fraction $s = 0.5, 0.7, 0.9, 1.0$ perform nearly equally well. Also the lasso regression with $s = 0.3$ follows the exact same function profile as the models with higher fraction values, only having a slight offset after reducing the number of features to 70. Solely the lasso regression with $s = 0.1$ performs worse than the rest. But also here the profile of the curve in Figure 5 follows the other regressions.
As a "safety check" the performance of the lasso regression with $s = 1.0$ with $p = 100$ features is supposed to be same as the OLS regression. Comparing the data this criterion evaluates as true, both regressions having $RMSE = 0.18193$ and $R^2 = 0.76934$.

Table 2. Linear Methods Performance

Model	# of Features	RMSE	R2
OLS	55	0.17461	0.78951
Lasso01	100	0.18919	0.75498
Lasso03	100	0.18039	0.77360
Lasso05	100	0.17900	0.77691
Lasso07	100	0.17983	0.77487
Lasso09	100	0.18120	0.77127
Lasso10	100	0.18193	0.76934

4.2.2 Random Forest

There were models for three Random Forests trained:

- Random Forest with ...
  - ... $B = 500$, $m = 5$ (RF05)
  - ... $B = 500$, $m = 7$ (RF07)
  - ... $B = 500$, $m = 10$ (RF10)

The best model for each method with performance metrics are shown in Table 3. While the number of trees per Random Forest were held constant at $B = 500$, the number of splits per tree m were varied. Across all Random Forests the performance increases slightly the more unimportant features are removed. In Figure 6 it can be easily observed that the performance in general and in dependence of the used number of features does not depent on m. All models perform equally well and vary only slightly.
Looking closer at the difference in performance between all tuned models within RF10, it shows in Table 4 that using far less features than the best performing model, the performance gets only marginally worse, but could improve training speed.
Even though the m size is held constant during model training, it is changed when not enough features are left to build a tree with m splits. Interestingly, the RF10 model performs better than the RF05 model with only 5 features left. This is in contrast to the idea that the feature selection wrapper would select the best variables needed for a model with specific tuning parameters, e.g. m.
Generally, all Random Forests have the best performance in RMSE and R^2 across all methods that were evaluated during this work.

Table 3. Random Forests Performance

Model	# of Features	RMSE	R2
RF05	15	0.14925	0.85225
RF07	30	0.15259	0.84667
RF10	35	0.15160	0.84855

Table 4. RF10 performance difference towards best performing RF10 with 35 features

Model	# of Features	$\Delta RMSE$	$\Delta R2$
RF10	20	-0.00157	-0.00041
RF10	15	-0.00051	-0.00329
RF10	10	+0.00240	-0.01131

4.2.3 Neural Network

There were models for three Neural Networks trained:

- Neural Network with...
  - ... M = 5 (NN05)
  - ... M = 10 (NN10)
  - ... M = 20 (NN20)

The best model for each method with performance metrics are shown in Table 5. The parameter size M is the number of neurons that are used in the single hidden layer in the model. Firstly, it can be stated that all Neural Networks performed worst across all models evaluated in this work. The best Neural Network NN05 with 5 features has an $R^2 = 0.4986547$ and $RMSE =$

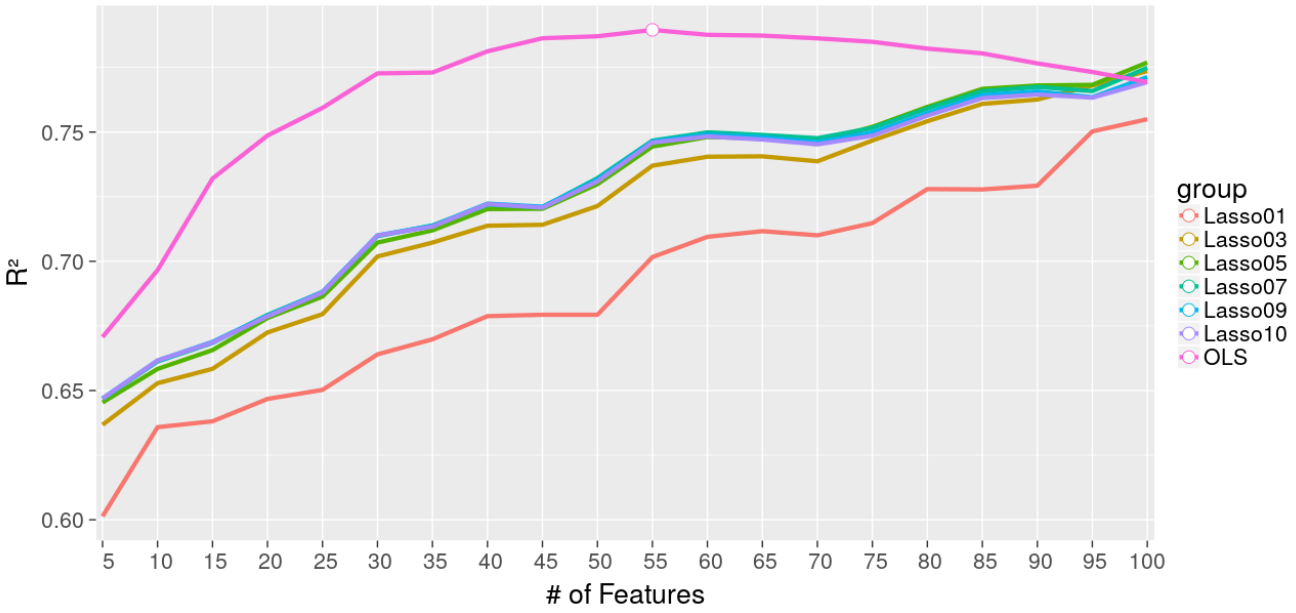

Fig. 5. OLS + Lasso Regression with $s = 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$

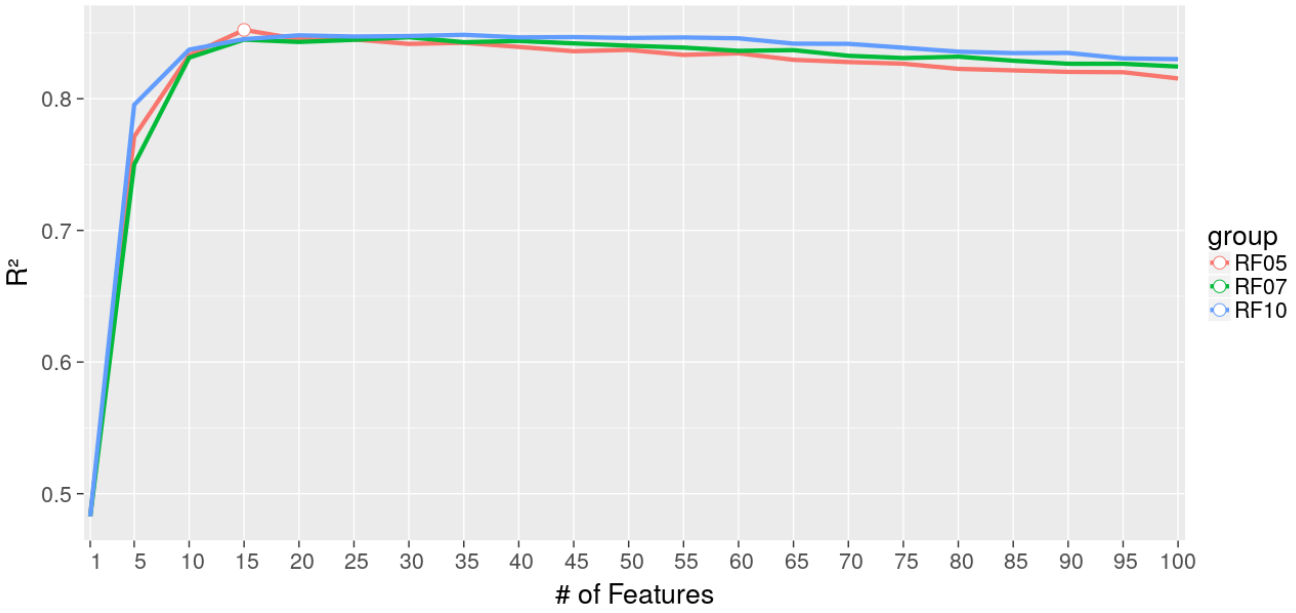

Fig. 6. Random Forest with $m = 5, 7, 10$

0.2728746. Looking at the performance graphs in Figure 7, there can be no clear trend observed between the number of features used for training the model and R^2 performance increase / decrease. Only a slightly positive trend can be observed when reducing the input variables for NN05 and NN10. The assumption could be made that by removing unimportant variables, the hidden units can focus more on less but more important variables.

In a Neural Network, the number of features represent the number of neurons in the input layer. After looking at the results of NN05 in Figure 7, the best performing model has the same number of input neurons (features) and hidden layer neurons. Unfortunately, this observation appears to be an incident, because for NN10 and NNET 20, the number of features in the best performing model are much higher than the number of neurons in the hidden layer. Also, the number of features in the best NN10 model are higher than in the best NN20 model, which removes the possibility that there could have been an ordinal relationship between hidden neurons and features.

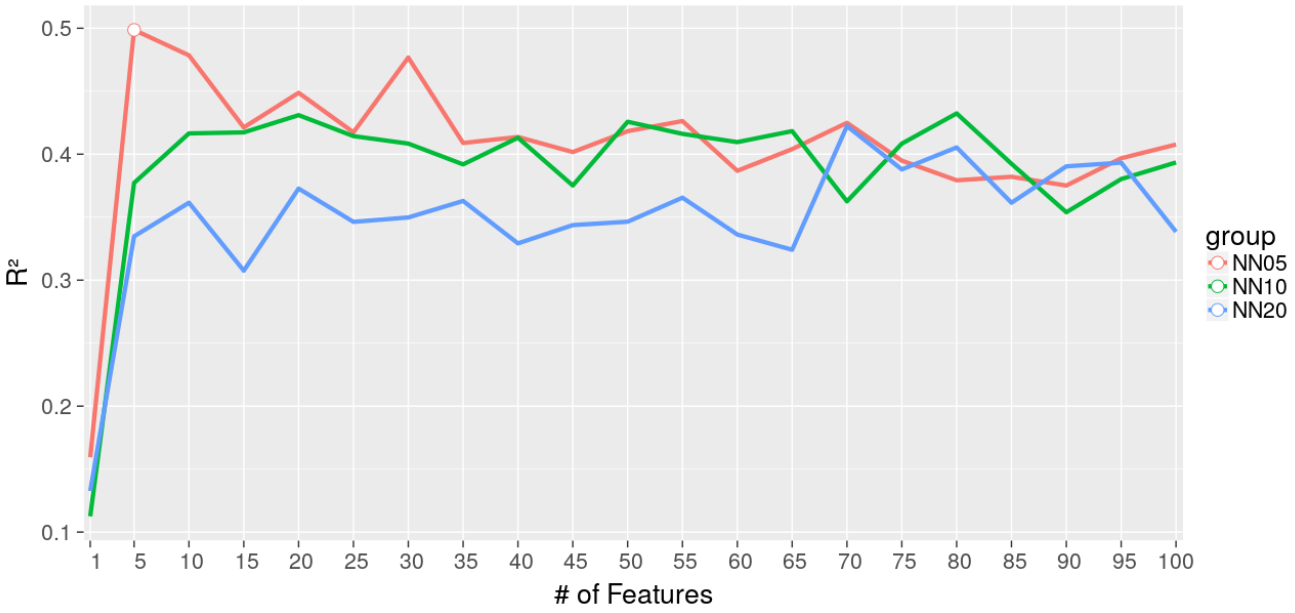

Fig. 7. Neural Network with $M = 5, 10, 20$ hidden units

Table 5. Neural Networks Performance

Model	# of Features	RMSE	R2
NN05	5	0.2728746	0.4986547
NN10	80	0.2893122	0.432392
NN20	70	0.2924961	0.4222825

Table 6. Performance Overview across all best models

Model Name	R2	# of Features
RF05	0.85225	15
RF10	0.84855	35
RF07	0.84667	30
OLS	0.78951	55
Lasso05	0.77691	100
Lasso07	0.77487	100
Lasso03	0.77360	100
Lasso09	0.77127	100
Lasso10	0.76934	100
Lasso01	0.75498	100
NN05	0.49865	5
NN10	0.43239	80
NN20	0.42228	70

4.3 Performance across Models

Table 6 show a summary with the best performing models across all methods and tuning parameters.

The overall best performing models were all Random Forest Regressions, independent of their tuning parameter m.

4.4 Feature Importance across Models

By using the feature selection algorithm described in 3.2.1, all models become quite well comparable when analysing which features were most important. As a metric the number of rounds which a feature survived the selection process is used. Because all models started with $p = 100$ features and every round $M = 5$ worst features are removed, a feature can survive at maximum 21 rounds. In Table 7 the best 5 features by mean numbers of rounds survived are listed. The best features are actually those that are based on a roll movement around the x-axis. By having the best mean it can be said that those features are the most general ones and can be used in all models to predict the outcome as precisely as possible. Especially *gyr_x_roll_Z_cwt_l_mean* ranks with a mean survival score of over 19 in nearly all models top 10 features.

Looking at feature performance from a maximum number of rounds survived score, the top feature is the same as in the best features by mean score: *gyr_x_roll_Z_cwt_l_mean*. Also the second best feature with max score is in the top 5 of the mean score

Table 7. Best 5 features by mean numbers of feature selection survived

Feature Name	Mean	Std. Dev.
gyr_x_roll_Z_cwt_l_mean	19.23	3.70
gyr_x_roll_Z_cwt_ll_mean	17.77	1.64
gyr_y_pitch_rms	17.00	5.34
gyr_x_roll_rms	16.85	3.13
gyr_x_roll_sd	16.69	4.09

Table 8. Best 5 features by max numbers of features selection survived

Feature Name	Max	Mean
gyr_x_roll_Z_cwt_l_mean	21	19.23
gyr_x_roll_sd	21	16.69
lin_acc_x_Z_cwt_m_mean	21	10.54
lin_acc_z_Z_cwt_m_mean	21	8.23
lin_acc_z_Z_cwt_h_mean	21	7.85
...	...	...
gyr_y_pitch_Z_cwt_hh_mean	13	7.31
lin_acc_z_Z_cwt_ll_mean	13	5.69
lin_acc_x_sd	12	8.38
speed_rms	12	5.31
magn_5_0	11	7.54

ranking. Besides these two feature that are based on the roll movement around the x-axis, the other max score features are based on linear acceleration along the x- or z-axis. Especially *lin_acc_z_Z_cwt_m_mean* and *lin_acc_z_Z_cwt_h_mean* are highly specialized feature. While they survived in one model all rounds and were selected the most important feature, the are not useful in other models, because they have a very low mean survival score. Also observing the worst 5 features by max score is interesting in this case: every feature in the data set survived at least once 11 rounds of feature selection before being discarded. This is either caused by a variety of variables which can be used well in specific models, but are unmeaningful in all other models, or by some arbitrary which features get removed first, e.g. differences in the variable importance of very small value.

5 Conclusion and Outlook

After executing the three dimensional analysis, a few conclusion can be drawn:

Regarding the feature extraction process, none of the amplitude-based features performed really well in total. All of them have a mean survival score of 12 rounds an less, which does not indicate great performance. While the information in the z-axis acceleration frequency based on those wavelengths might be still valuable for extracting features, the method described in 3.1 might not be the best one. Further research on how other methods, e.g. Power Spectral Density, could extract better features could be done in the future.

The feature selection algorithm worked quite well and improved at least for the OLS and the Random Forest Regressions the R^2 performance by some value. Yet, through the step size of $M = 5$ the global maxima of some models might have been overstepped and not identified. Setting $M = 1$ or using one of the more computationally demanding feature selection methods could bring even more interesting results.

Applying the feature selection algorithm to Lasso regression was an unnecessary computational load. As described in 3.3.1 and 4.2.1, the method already includes feature selection by variable shrinkage. Comparing the best lasso regression model to the best OLS model, the performance difference is small, but lasso simply uses all features that were used as input and uses those that are relevant to it. The OLS model needs to iteratively approach the best selection of features for its peak performance. Therefore, the lasso regression is a very convenient way to train a model without knowing much about the value of input variable

The selected training methods resulted in mixed results. The Random Forest performed best across all models, followed by the OLS and lasso regressions. All Neural Networks disappointed in a performance below $R^2 = 0.5$. It would be interesting to train multi-layered Neural Networks in future analysis and vary more tuning parameters as weight decay or number of neurons.

Searching for a training method that works best for the prediction of IRI using all the input features, the Random Forest Regression should be first choice due to its high coefficient of determination and small errors.

References

[1] Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H., 2008. The pothole patrol: using a mobile sensor network for road surface monitoring.

[2] Mohan, P., Padmanabhan, V. N., and Ramjee, R., 2008. "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones". *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*, p. 323.

[3] Perttunen, M., Mazhelis, O., Cong, F., Ristaniemi, T., and Riekki, J., 2011. "Distributed Road Surface Condition Monitoring". *Lecture Notes in Computer Science*, pp. 64–78.

[4] Michael W Sayers, Thomas D. Gillespie, and Paterson, W. D. ., 1986. *Guidelines for Conducting and Calibrating Road Roughness Measurements*. No. 46.

[5] Sayers, M. W., and Karamihas, S. M., 1998. "The little book of profiling". *Basic Information about Measuring and Interpreting Road Profiles*(September), p. 100.

[6] Seraj, F., Meratnia, N., and Havinga, P. J., 2017. "RoVi: Continuous transport infrastructure monitoring framework for preventive maintenance". In 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE, pp. 217–226.

[7] Sueur, J., Aubin, T., and Simonis, C., 2008. "Seewave: a free modular tool for sound analysis and synthesis". *Bioacoustics,* **18**, pp. 213–226.

[8] Laubis, K., Simko, V., and Schuller, A., 2016. "Road Condition Measurement and Assessment: A Crowd Based Sensing Approach". *Thirty Seventh International Conference on Information Systems*, pp. 1–10.

[9] Hastie, T., Tibshirani, R., and Friedman, J., 2009. "The Elements of Statistical Learning". *Elements,* **1**, pp. 337–387.