Machine Learning: regression, dimensionality reduction and data visualization (2016-2017)
Université Catholique de Louvain

Airfoil Noise Prediction

Alejandro Simón Rodríguez

21 de diciembre de 2017

Índice

1 Introduction **3**

2 Exploration and Analyzing Data **3**
2.1 Feature selection 3
2.2 Outliers 4
2.3 Normalization 5

3 Multilinear Regression **5**

4 Neural Network **6**

5 K-Nearest-Neighbors **8**

6 Conclusion **10**

7 Annex **11**
7.1 Annex A 11
7.2 Annex B 11
7.3 Annex C 12
7.4 Annex D 12

Índice de figuras

2.1. Graph of relation between frequency and period of the noise 3
2.2. Result of boruta algorithm 4
2.3. Boxplot of X1 5
3.1. Result of k-Cross Validation Variying the train and validation sets 6
4.1. Generalization Error depending on k per Number of Neurons 7
4.2. Generalization Error depending on k with two layers 8
5.1. Generalization Error per Number of Neighbors 9
5.2. Result of k-Cross Validation Variying the train and validation sets 10

Índice de tablas

3.1. Result of RMSE taking five differents test sets 6
4.1. Result of RMSE taking five differents test sets 8
5.1. Result of RMSE taking five differents test sets 10
6.1. RMSE from the three model 11
7.1. Result of k-Cross Validation Variying the train and validation sets 12
7.2. Result of k-Cross Validation Variying the train and validation sets 12

1. Introduction

In this project I'm going to justify theoretically and empirically which model will be the best to predict the noise of an airfoil.I have developed three models a multilinear regression model, a multilayer-perceptron (neural network) and the k-nearest-neighbors algorithm.
To implement practically the models I have used R as programming language and the librarys Boruta,Neuralnet and FNN.

2. Exploration and Analyzing Data

2.1. Feature selection

First of all, before apply a feature selection methods I have done an univariable and bivariable analysis of the data and I have seen that Noise Frequency and Noise Period are related. I have used a scatter plot function to see the relation and then I have found the strength of the relationship with correlation.They have a correlation of -0.57, in fact, they are inversely proportional.

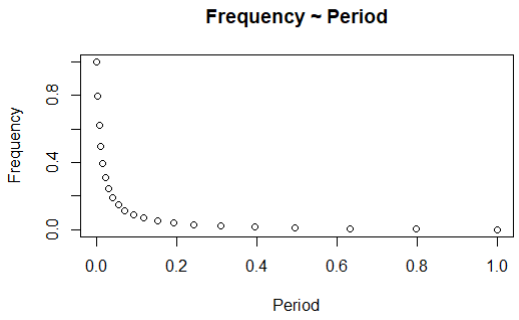

Figura 2.1: Graph of relation between frequency and period of the noise

In brief,those variables are redudants and they contribute with the same information to the model.Hence,I have removed Noise Frequency from the model.

After this analysis, I have used Boruta algorithm for this part [3].Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest.I have chosen boruta because boruta finds all features which are either strongly or weakly relevant to the decision variable.This makes it well suited for our model where one might be interested to determine which variables are more strongly connected with the target variable.

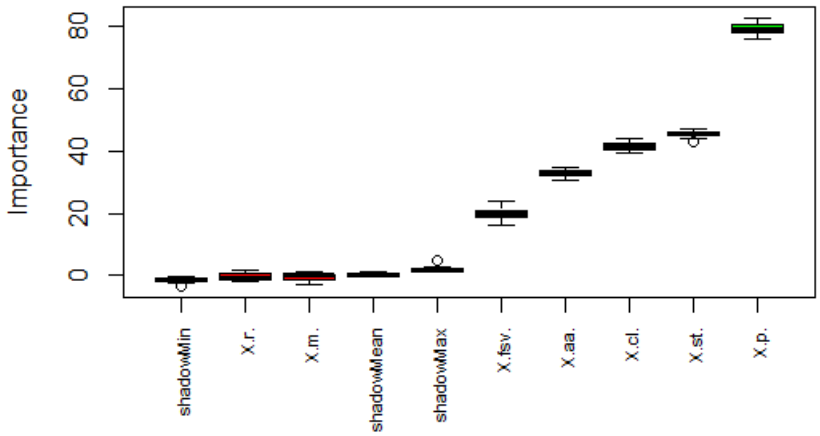

Figura 2.2: Result of boruta algorithm

The shadow values are creating by shuffled copies of all features.Boruta iteratively compares importances of attributes with importances of shadow attributes, created by shuffling original ones. Attributes that have significantly worst importance than shadow ones are being consecutively dropped. On the other hand, attributes that are significantly better than shadows are admitted to be Confirmed. Shadows are re-created in each iteration [4].

To conclude,if we observe the results we can see that the variables manufacturer (X.m.) and normalized roughness coefficient (X.r.) should be dropped.On the other hand,we should use for the model the rest of the variables.

2.2. Outliers

After feature selection I have studied the outlaiers of the data set.To study this I have represented the boxplot of the X1-features and X2-features.

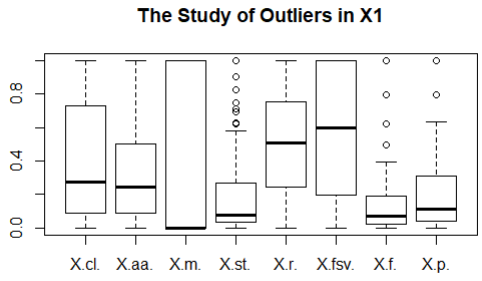

Figura 2.3: Boxplot of X1

I have seen that there are outliers in the same features in both models.Futhermore,there are a lot of outliers in both models,thus i have decided not to remove the outliers because if I remove these values from X1 set I will not be able to predict these values for the target variable (taking X2 set with the outliers). These outliers can be values very difficult to reach but can be values possibles to obtain.

2.3. Normalization

I have chosen Normalize data for many reasons.First of all,the features of the models have differents scales and it's more easy for interpret result with scaled data.The others reasons are: normalize the data is a good idea for the algorithms Knn or MLP,because it can be good for euclidean distance used in knn and to improve the time to learn in a neural network (based in gradient techniques).

I have used Min-Max normalization and the formule is:

$$X_{Normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3. Multilinear Regression

I have implemented Multilinear Regression with *lm* function.I have wanted study well if the overfitting in my model. Because of that I have divided the model like I explain in the section 7.1.

In brief,I have applied five times k-cross-validation and I have estimated five RMSE. I have done these divisions to have a more generalized and real idea of the values calculated. The results are:

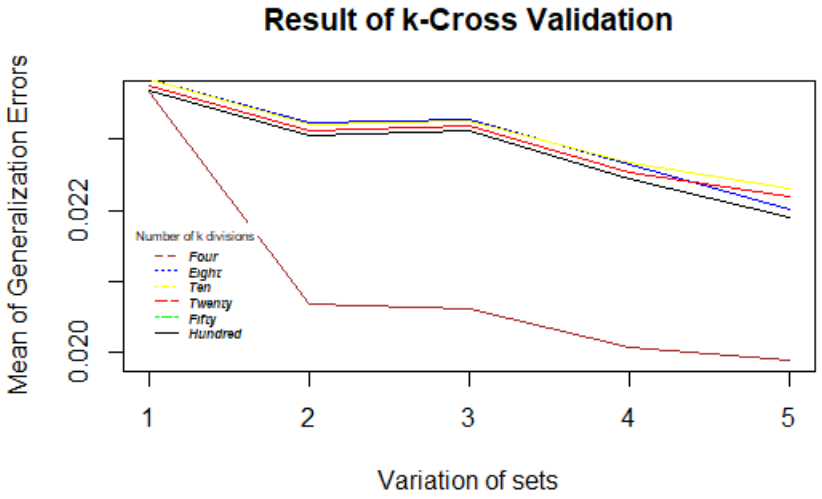

Figura 3.1: Result of k-Cross Validation Variying the train and validation sets

Seeing the graph,we can notice that the variation of averages of generalization errors is small. Thus, We can observe that the model does a good estimation of the values to every Cross Validation Test and the Generalization Errors are quite stable for each iteration. Hence, I can conclude that model doesn't have overfitting and will do a good prediction of the test set.

After notice that model doesn't have overfitting,I have estimated the Root Mean Square Error (RMSE) for every test set built.The results are:

Error	**First**	**Second**	**Third**	**Fourth**	**Fith**	**Means**
RMSE	0.139051	0.148131	0.146747	0.156359	0.161986	0.150455

Tabla 3.1: Result of RMSE taking five differents test sets

In the section 6 I will discuss which model I will use observing the RMSE of each model.

4. Neural Network

I have implemented the Neural Network with *NeuralNet* package.This package allows me to choose all the parameters required like the number of layers,the number of neurons per layer and the learning algorithm.

Numero de capas:Justificacion con resultados
I have decided to use the Resilient Back Propragation(rprop) algorithm for learning algorithm step.This algorithm is better than Back Propragation because it's faster and

more efficiency.In fact, you don't have to decide a learning rate but it's more difficult to implement [2].

I have used k-Cross-Validation for estimating which is the optimal number of neurons per layer and how many hidden-layers need for my model.I have implement the same system of division on the previous cross validation step but this time with less variation of k a cause of the time to run a neural network.

Firstly,i have researched the numbers of neurons per layer.The next graph shows the relation between the means of the generalization errors depending on k and the number of neurons:

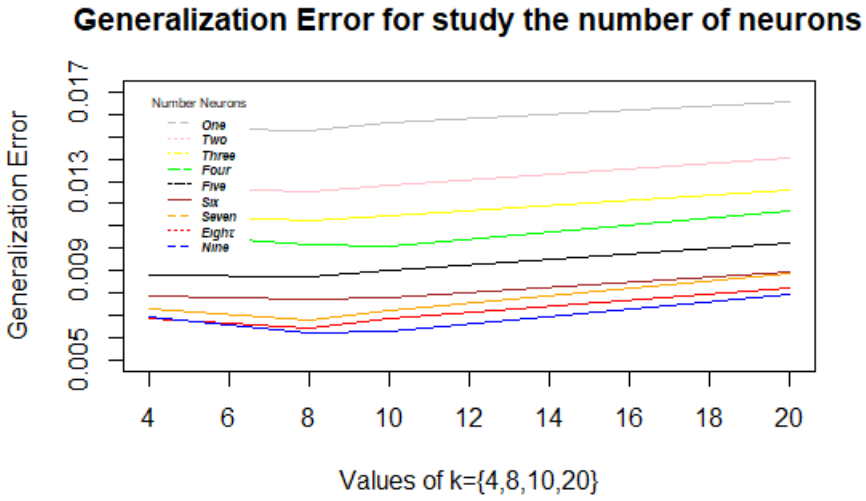

Figura 4.1: Generalization Error depending on k per Number of Neurons

As we can see in the graph above,there is a relation between decrease of errors and numbers of neurons is inversly proportional.Nevertheless,this also augment the time for run the neurals networks.Hence, I have had to find a balance among accuracy and time.

Secondly, I have researched the numbers of layers.To answer this question,I have performed cross validation with two layers and variation of the number the neurons per layer. For example,in the next graph I vary the number of neurons in the first layer:

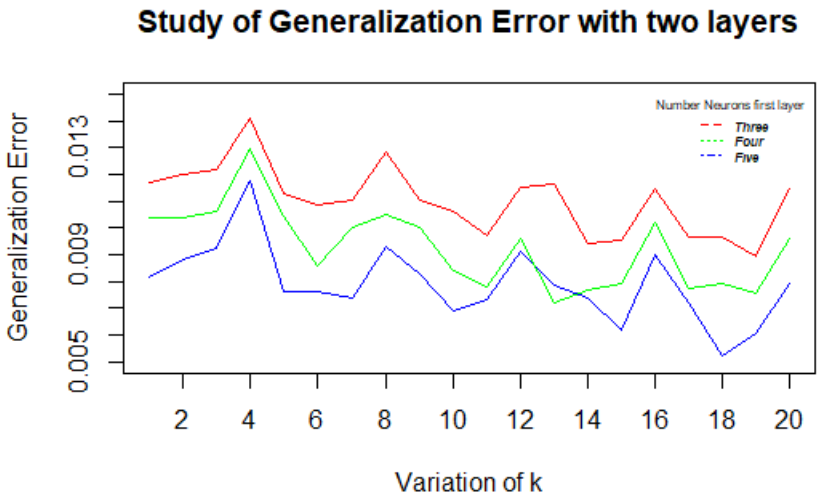

Figura 4.2: Generalization Error depending on k with two layers

I have noticed seeing the results that the model with two layer has high accuracy but the generalization errors weren't so stable.This meant that the model had very high precision for some models and for others not. Hence, i can conclude that there was some overfitting with two layers in the model.

Futhermore, before this study i have investigated about The Universal Aproximation Theorem For Neural Network [1].In brief, the problem of the model satisfy the hypothesis, thus to use one layer is justified for the model.

Finally,I'm going to choose one layer with 15 neurons for the representation from the neural network.The results are:

Error	**First**	**Second**	**Third**	**Fourth**	**Fith**	**Means**
RMSE	0.075824	0.061547	0.066576	0.060430	0.056878	0.06425

Tabla 4.1: Result of RMSE taking five differents test sets

In the section 6 i will discuss which model i will choose for the predicting X2.

5. K-Nearest-Neighbors

I have implemented K-Nearest-Neighbors with *FNN* package. One time inside to this package, I have used *knn.reg* function because the most typical use of knn is clasifications problems.

For k-cross-validation I have done the same division of the data as Multilinear Regression and also the same loops for varying the test,train and validation sets for every loop.I have used k-cross-validation to decide the numbers of neighbours that i have to choose for the model and to study the overfitting from the model.

To know the numbers of neighbors I have tested the model from 1 to 300 neighbors and in each of this test I have applied the cross validation technique. I have made a graph where we can see the mean of every Generalization Errors per Number of Neighbors.

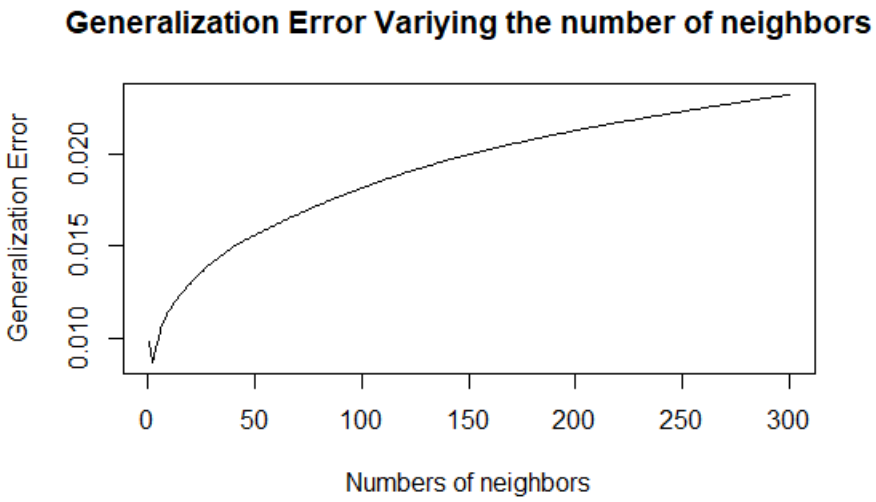

Figura 5.1: Generalization Error per Number of Neighbors

As we can see, the model has the lowest Generalization Error when we have only one neighbor.Then the model increases directly proportional the Generalization Error with the numbers of neighbors.

To study the overfitting of the model i have collected, the results of K-Cross-Validation using one neighbor. The results are:

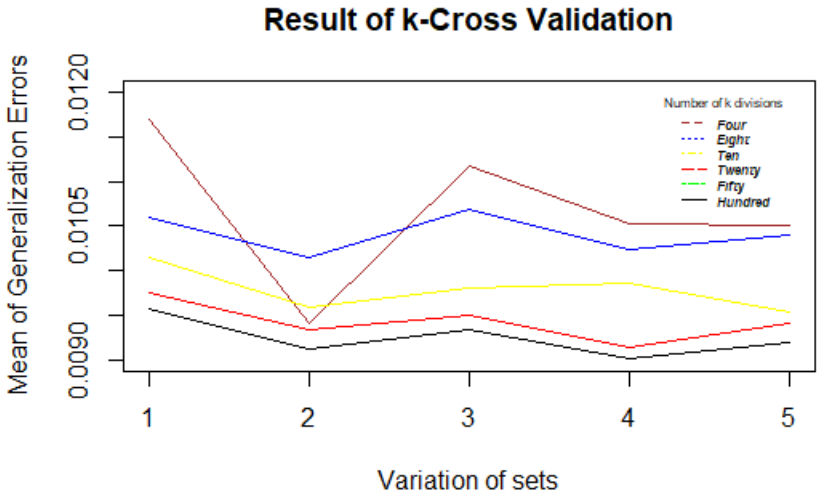

Figura 5.2: Result of k-Cross Validation Variying the train and validation sets

Seeing the graph,we can observe that the variation of averages of generalization errors is low.We can conclude that the model doesn't have overfitting because the Generalization Errors are enough stable. In consecuence, the model will estimate a prediction of the test set with high accuracy.

Finally, I have taken the results on the test set. I have calculated the Root Mean Squared Error and these are the results:

Error	**First**	**Second**	**Third**	**Fourth**	**Fith**	**Means**
RMSE	0.085102	0.094009	0.095682	0.088058	0.097177	0.092006

Tabla 5.1: Result of RMSE taking five differents test sets

In the section 6 i will discuss the final election of the model for predicting X2.

6. Conclusion

In the project I have done a total analysis of the problem.Firstly, I have realized a pre-process of the data where I have discussed the feature selection,the ouliers and the normalization. Secondly, I have tested three models: Multilinear Regression, Multilayer Perceptron and K-Nearest-Neighbors. After test them with cros validation I have calculated the RMSE. The summary of these RMSE of each model is:

Error	ML-R	MLP	KNN
RMSE	0.150455	0.06425	0.092006

Tabla 6.1: RMSE from the three model

Finally, I can conclude that the best model for predicting the result is a neural network model 7.4 with one layer and the more neurons as possible, for 6.1 I have chosen a model with one layer and 15 neurons. Nevertheless,KNN is also a very good option for the prediction and too much faster than MLP.

7. Annex

7.1. Annex A

In this section I'm going to explain the method that I have used for k-cross validation. Firstly,for adding more randomness , I create a loop with 5 cycles and in each cycle the subsets are differents. Secondly, I have divided the set into three subsets: train, validation and test sets. The size of the test set is 20 % of the data set because this set has to have enough size to be representative and estimate the error. For example:

One cycle could be:

Test set: the first 200 data

Validation and train = the rest of the data

Second cycle could be:

Test set: the 200-400 data

Validation and train = the rest of the data

And so on.....

After, I have divided the validation and train sets following k-Cross-Validation.The values of k have been 4,8,10,20,50,100 for Multilinear regression and for K-Nearest-Neighbors. Although, the values of k have been 4,8,10,20 for neural network a cause of time to run Neural Network.

7.2. Annex B

Table of k-cross Validations results in Multilinear Regression algorithm

K	First	Second	Third	Fourth	Fith	Means
4	0.024671	0.022259	0.022285	0.021679	0.021344	0.022447
8	0.024864	0.024352	0.024476	0.023720	0.022854	0.024053
10	0.024962	0.024462	0.024569	0.023832	0.023134	0.024192
20	0.024780	0.0242549	0.024397	0.023634	0.023036	0.024020
50	0.024665	0.0241602	0.024301	0.023534	0.022712	0.023874
100	0.024504	0.023972	0.024123	0.023324	0.022835	0.023752

Tabla 7.1: Result of k-Cross Validation Variying the train and validation sets

7.3. Annex C

Table of k-cross Validation algorithm in knn

K	First	Second	Third	Fourth	Fith	Means
4	0.011695	0.009409	0.011170	0.010531	0.010510	0.010663
8	0.010603	0.010144	0.010681	0.010239	0.010400	0.010413
10	0.010145	0.009587	0.009799	0.009855	0.009534	0.009784
20	0.009752	0.009333	0.009509	0.009142	0.009411	0.009429
50	0.009569	0.009130	0.009340	0.009022	0.009191	0.009250
100	0.009348	0.008885	0.009182	0.008921	0.009246	0.009117

Tabla 7.2: Result of k-Cross Validation Variying the train and validation sets

7.4. Annex D

To execute the code to obtain the result, you should normalize by min-max criterium the target of X2 to calculate de RMSE. Once you have normalized the target of X2,you have to use two last lines of script **neuralnetwork.r** which will be commented to calculate RMSE.

Referencias

[1] `https://www.researchgate.net/profile/George_Cybenko/publication/226439292_Approximation_by_superpositions_of_a_sigmoidal_function_Math_Cont_Sig_Syst_MCSS_2303-314/links/551d50c90cf23e2801fe12cf/Approximation-by-superpositions-of-a-sigmoidal-function-Math-Cont-Sig-Syst-MCSS-2303-3pdf`, 1989.

[2] `https://pdfs.semanticscholar.org/916c/eefae4b11dadc3ee754ce590381c568c90de.pdf`, 1993.

[3] `https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/` 2016.

[4] https://cran.r-project.org/web/packages/Boruta/Boruta.pdf, 2017.