A case study : Influence of Dimension Reduction on regression trees-based Algorithms -Predicting Aeronautics Loads of a Derivative Aircraft

Edouard Fournier, Stéphane Grihon, Thierry Klein

A case study : Influence of Dimension Reduction on regression trees-based Algorithms - Predicting Aeronautics Loads of a Derivative Aircraft

Edouard Fournier*[1,2,3], Stéphane Grihon†[2], and Thierry Klein‡ [1,3]

[1]Institut de mathématique, UMR5219; Université de Toulouse; CNRS, UPS IMT, F-31062 Toulouse Cedex 9, France
[2]Airbus France 316, Route de Bayonne, Toulouse France.
[3]ENAC - Ecole Nationale de l'Aviation Civile, Université de Toulouse, France

Abstract

In aircraft industry, market needs evolve quickly in a high competitiveness context. This requires adapting a given aircraft model in minimum time considering for example an increase of range or the number of passengers (cf A330 NEO family). The computation of loads and stress to resize the airframe is on the critical path of this aircraft variant definition: this is a consuming and costly process, one of the reason being the high dimensionality and the large amount of data. This is why Airbus has invested since a couple of years in Big Data approaches (statistic methods up to machine learning) to improve the speed, the data value extraction and the responsiveness of this process. This paper presents recent advances in this work made in cooperation between Airbus, ENAC and Institut de Mathématiques de Toulouse in the framework of a proof of value sprint project. It compares the influence of three dimensional reduction techniques (PCA, polynomial fitting, combined) on the extrapolation capabilities of Regression Trees based algorithms for loads prediction. It shows that AdaBoost with Random Forest offers promising results in average in terms of accuracy and computational time to estimate loads on which a PCA is applied only on the outputs.

Keywords:
Regression trees-Aeronautics-Dimensional reduction-Extrapolation *MSC Classification:*
62J02, 62-07, 63P30

*edouard.fournier@airbus.com
†stephane.grihon@airbus.com
‡thierry.klein@math.univ-toulouse.fr

1 Introduction

In aircraft industry, market needs evolve quickly in a high competitiveness context. This requires adapting a given aircraft model in minimum time considering for example an increase of range or of the number of passengers such as the A330 family in Airbus (2017). In our case study, variants concern the maximum take-off weight of a given aircraft model. Depending on the configuration, the computation of loads and stress, as defined in Hoblit (1988); Hjelmstad (2005), to resize the airframe is on the critical path of this aircraft variant definition: this is a time consuming (approximately a year for a new aircraft variant) and costly process, one of the reason being the high dimensionality and the large amount of data. Big Data approaches such as defined by Gandomi and Haider (2015) is mandatory to improve the speed, the data value extraction and the responsiveness of the overall process. This study has been realized during a proof of value sprint project within Airbus to demonstrate the usefulness of statistics and machine learning approaches in the Engineering field. In a previous internal project, it has been shown that the family of regression trees Breiman et al. (1984) works well to predict loads for different aircraft missions in an interpolation context. Thus, we can formulate our problem in this way: is it possible to use dimensional reduction and regression trees-based algorithms to predict loads in an extrapolation context (i.e outside the design space of a certain weight variant) to improve the actual process?

1.1 Industrial context

An airframe structure is a complex system and its design is a complex task involving today many simulation activities generating massive amounts of data. Such is the case of the process of loads and stress computations for an aircraft (that is to say the calculations of the forces and the mechanical strains suffered by the structure) and can be represented as follow:

This overall process is run to identify load cases (i.e aircraft mission and configurations: maneuvers, speed, loading, stiffness...), that are critical in terms of stress endured by the structure and, of course, the parameters which make them critical. The final aim is to callibrate the load to reduce the weight of the structure. Typically for an overall aircraft structure, millions of load cases can be generated and for each of these load cases millions of structural responses (i.e how structural elements react under such conditions) have to be computed. As a consequence, computational times can be significant.

For a derivative aircraft, we can give some rough order of magnitudes in terms of quantities of produced data: External loads (10^6 of bytes); Weights: number of elements (10^4 of bytes); Internal loads: number of components by the number of external loads by the number of elements (10^{11} of bytes); Reserve Factors: number of internal loads by the number of failure modes (10^{12}

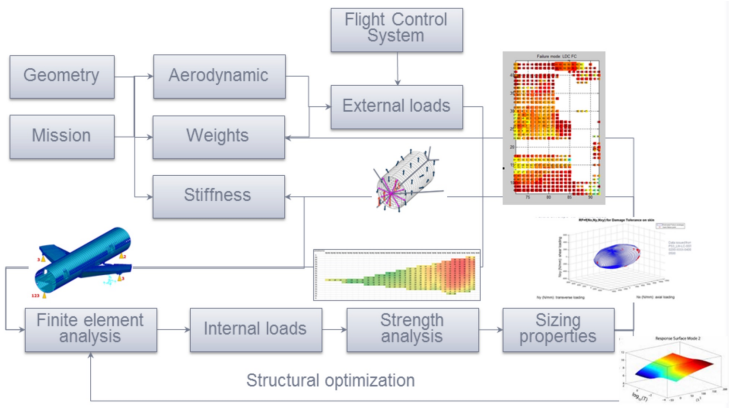

Figure 1: Flowchart for loads and stress analysis process

of bytes). Hence, we easily reach 10^{18} to 10^{21} of bytes for a single derivative aircraft.

In an effort to continuously improve methods, tools and ways-of-working, Airbus has invested a lot in digital transformation and the development of infrastructures allowing to treat data (newly or already produced). The objective here is to exploit and adapt Machine Learning and optimization tools in the right places of the computational process. As pointed by Manyika and al. (2011), these techniques cover a large number of fields such as Internet and Business Intelligence but they can also benefit to the manufacturing industry (here aeronautics). The main industrial challenge for Airbus is to reduce lead time in the computation of loads and preliminary sizing of an airframe.

1.2 A simplistic load and stress model computation process example

In order to illustrate the process exposed in the previous subsection, let consider a simplistic load model completed with equations calculating thickness used to correct the weight distribution of a wing structure similar to Doherty (2009) (Figure 2).

The structure contains a fuel tank at the wing tip with the dimensions specified above. The length of the wing is L, the chord length at wing root is Co and at the tip Ct. As a consequence, there are three different types of loads which affect the wing: the aerodynamic lift Q_{lift} (i.e the force which allows the aircraft to lift off and to maintain altitude) which depends on the length of the wing, the load factor and the total weight of the aircraft; the loads concerning the fuel & fuel tank weight Q_{fuel} depending on the fuel weight and the dimension of the

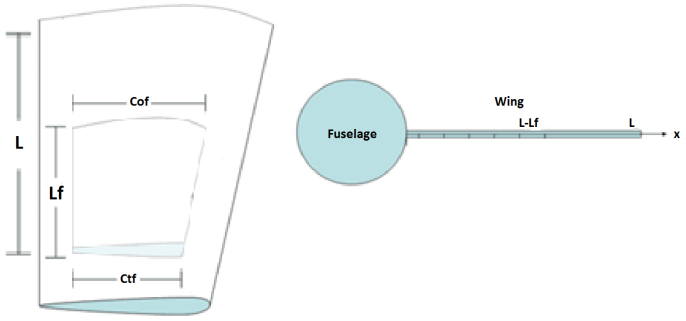

Figure 2: Scheme of the wing structure considered in the load model

fuel tank; and the loads due to the wing structure $Q_{wingstructure}$ weight and the dimension of the wing. By adding these three types of loads, and providing the weight of the wing structure, the weights of the tank and the fuel contained, as well as the total weight of the aircraft and the load factor; Q_{total} provides the basis for calculating the shear force V (unaligned and opposed forces internal to the material) and bending moment M (the reaction induced (bending) when an external force or moment is applied) of the wing. The relation between these quantities is :

$$V(x) = -\int_0^L Q(x)dx$$

$$M(x) = \int_0^L V(x)dx$$

where x is the position along the wing. We consider that the wing is represented by a box of the form as shown in Figure 3.

We can complete equations calculating thickness. Indeed, by considering the box has height $h(x)$ supposed linearly decreasing along the span, considering we must not exceed an allowable of σ_{max} tension and compression. Considering the fluxes in the wing covers are given by $N(x) = \frac{M(x)}{h(x)C(x)}$ thus we have the thickness distribution defined by:

$$t(x) = \frac{M(x)}{h(x)C(x)\sigma_{max}}$$

And by integrating we get the weight of the cover given by:

$$W_{cover} = 2\int_0^L \frac{M(x)}{h(x)C(x)\sigma_{max}}dx$$

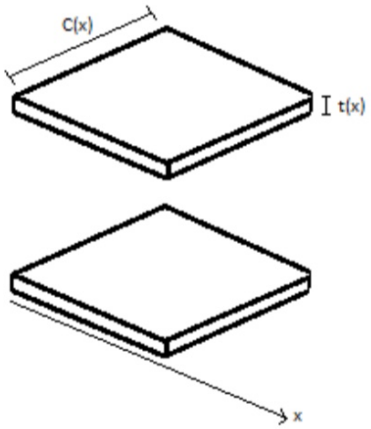

Figure 3: Form of the box

Indeed, by considering that the box takes the form above, by integrating $t(x)C(x)$ along x and by multiplying by 2ρ, where ρ is the density of the material used to fabricate wing panels, we get the weight of the wing cover. More precisely, we obtain the minimum weight of the wing cover able to resist an allowable σ_{max} tension and compression. We assume that $W_{cover} = 70\%W_{wing}$, then we can extract the minimum weight of the wing structure able to resist an allowable σ_{max} tension and compression.

1.3 Data presentation

The data we have at our disposal are the aircraft parameters (features) which are used in the computing chain for calculating loads (outputs which correspond to moments and forces). We have data coming from the weight variant 238 tons (aircraft parameters and loads distribution along the wing); and we would like to predict those of the 242t and other weight variants (247t and 251t).

24 aircraft (A.C.) parameters play the role of features (lying in $\mathbb{R}$) and we would like to predict loads (outputs) which are in $\mathbb{R}^k$. To simplify, we will focus on predicting bending moment along the wing which is, in our data, represented by a vector of size $k = 29$. The data base is mainly constituted by gusts (90% of all load cases) and we will focus on them. To begin, we shall focus on the 238t and 242t data before generalizing our results to other weight variants. Before going any further, follows a quick summary of the size of our different datasets:

	238t(Train&Test)	242t(Validation)
Dimension data features	28391 rows x 24 col.	28391 rows x 24 col.
Dimension data outputs	28391 rows x 29 col.	28391 rows x 29 col.

Table 1: Description of the datasets

In a more formal way, let be the 238t database of features defined by $\mathbf{X} = (X^1, ..., X^{24})$ where X^j are quantitative variables (i.e a A.C. parameter), and $X^j = (x_1^j, ..., x_{28391}^j)^T$. The 238t database of outputs is then defined by $\mathbf{Y} = (Y^1, ..., Y^{29})$ and $Y^j = (y_1^j, ..., y_{28391}^j)^T$. Here follows a description of the aircraft parameters $\mathbf{X}$ (inputs) we have at our disposal in the training data base 238t:

Description	Distribution type	Mean	Std	Min	Max
Defl. Left inboard Elevator	Gaussian	0.015	0.034	-0.116	0.108
Stabilizer Setting	Mixture of Gaussian (2 modes)	-0.033	0.023	-0.093	0.0033
Defl. Spoiler 1 Left Wing	Bi Modal	-0.221	0.218	-0.436	0
Defl. Spoiler 2 Left Wing	Mixture of Gaussian (2 modes)	-0.266	0.262	-0.755	0.230
Defl. Spoiler 3 Left Wing	Mixture of Gaussian (2 modes)	-0.266	0.262	-0.755	0.230
Defl. Spoiler 4 Left Wing	Mixture of Gaussian (2 modes)	-0.266	0.262	-0.755	0.230
Defl. Spoiler 5 Left Wing	Mixture of Gaussian (2 modes)	-0.266	0.262	-0.755	0.230
Defl. Spoiler 6 Left Wing	Mixture of Gaussian (2 modes)	-0.266	0.262	-0.755	0.230
Defl. Low speed outer Aileron	quadrimodal	-0.028	0.053	-0.157	0
Lower part Rudder Deflection	Gaussian	0	0.011	-0.072	0.072
Total A.C. Mass	Multimodal	195738	35428	135093	238000
Mach Number	Multimodal	0.716	0.19	0.372	0.93
True Airspeed	Multimodal	223	50	126	282
Altitude	Multimodal	6270	4519	0	12634
x-location of cg in % amc	Multimodal	0.297	0.114	0.140	0.42
Thrust(calculated)	Multimodal	131442	157160	0	415495
X-Load Factor	Gaussian	-0.020	0.107	-0.3	0.261
Y-Load Factor	Gaussian	0	0.08	-0.306	0.307
Z-Load Factor	Gaussian	1.024	0.43	-0.701	2.643
Fuel Tank mass TANK1L	Multimodal	392	1030	0	4341
Fuel Tank mass TANK2L	Multimodal	13008	12721	0	36295
Fuel Tank mass TANK3L	Multimodal	1883	1377	0	3087
Fuel Tank mass TANK1L	Multimodal	945	1029	0	2592
Left inner engine thrust	Multimodal	65721	78579	0	207747

Table 2: Description of the 238t dataset

The bending moment is calculated at 29 points along the wing - each point represents a station and stations are not equidistant (two more stations are located in the center wing box; we prefer to focus here on stations of the wing only). Thus Y^k represents the values of the bending moment taken at the k^{th} station. Through a change of coordinate system (aircraft system to wing system), we can easily plot bending moments as follow:

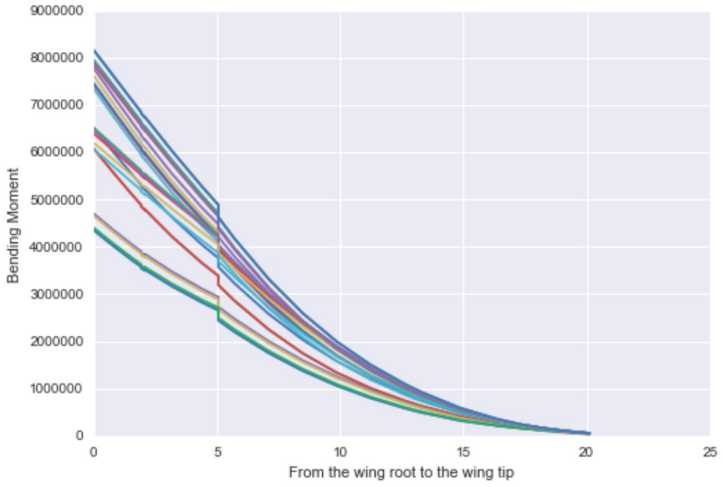

Figure 4: Bending moments along the wing for different load cases

1.4 Industrial problem

Aircrafts (A.C.) have been developed for different maximum take-off weight (which is one of the many aircraft parameters used in the computing chain to calculate the loads). Because the computation process exposed above for a new aircraft variant (a new weight variant in our case) can reach easily a year, the use of meta-models, optimization and statistic approaches such defined by Gandomi and Haider (2015) is mandatory to improve the speed and responsiveness of the overall process.

From this standpoint, we can expose the following problem: for each combination of A.C. parameters corresponding to a load case, and each load case being categorized into a load condition (family of load cases – gusts or maneuvers), can we give an estimation of the loads for different A.C. parameters for new weight variants (242t, 247t and 251t) knowing the loads of the weight variant 238t?

The mathematical problem of this project is an extrapolation problem. Is it possible to 'extrapolate' loads of the 242 tons, 247t and 251t knowing loads of the 238t by using machine learning? To be more precise, can we find a function depending on aircraft parameters that allows us to estimate/extrapolate to 242t and other weight variants by learning from those of the 238t? In a previous project concerning loads, it has been shown that the family of regression trees works well on the data we have to deal with. As a consequence, different algorithms based on decision trees will be investigated. Besides, because of the dimension of our outputs, how do dimensional reduction techniques affect the

capability of extrapolation of machine learning algorithms based on regression trees?

This paper is organized as follow: the following part relates to the three dimensional reduction techniques we have used, the third exposes the different algorithms based on regression trees and the last part outlines our results before we conclude.

2 Three Dimensional Reduction Techniques

Because the outputs are vectors, it is mandatory to test several dimensional reduction techniques to improve the efficiency and speed of the modeling process. In the first part, we shall discuss about the principal components analysis. In the second part, we will consider a polynomial fitting, and in the last part we will mix both methods. These dimensional reduction techniques will reduce the dimension of the output space. Each technique has been used on the 238t, and these allow us to reverse the technique to come back to the original output space easily.

2.1 Principal Components Analysis

In few words, the Principal Components Analysis (PCA), developed by Pearson (1901) and formalized by Hotelling (1993) is a statistical method used to compress a matrix n x p of quantitative variables into a smaller rank matrix. This method uses the variance-covariance matrix (or correlation matrix) to extract important factors (few in general) to represent observations in a smaller subspace. As a consequence, each observation is represented by coordinates into new components linked to these factors (this approach is similar to the SVD decomposition).

We apply the PCA in the space defined by the outputs (centered and reduced), and here is the decline of the variance explained by each component as well as the cumulative percentage of the explained variance:

Few components (6) explain 99.99% of the total variance. When we look closer at the correlation of the original variables with the principal components, we see that all features have a similar correlation coefficient with the two first principal components.

2.2 Polynomial fitting

Because the outputs we are trying to predict are vectors/curves, we can summarize the information of each curve by fitting a polynomial function instead. Indeed, each curve represents the bending moment along the wing and we can

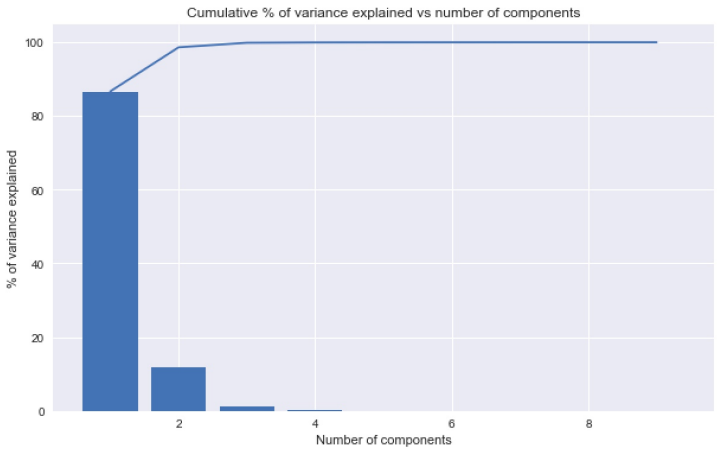

Figure 5: Cumulative percentage of the explained variance

clearly see that the curve has a quadratic form.

As a consequence, we consider that it exists a polynomial function p of degree d such as:

$$p(x) = a_0 x^d + ... + a_d$$

The coefficients $a_0, ..., a_d$ are obtained by minimizing the squared error by the least squares method.

Nevertheless, a discontinuity always appears at the 12th station along the wing. Consequently, we should fit a polynomial on the first part of the curve and another on the second. In order to choose properly the degree of each polynomial, we must assess the quality of the fit.

The optimal couple of degrees would be 2 for the first polynomial and 2 for the second polynomial as well. The dimension of the output space would be 6 instead of 29.

2.3 Polynomial fitting & Principal Components Analysis

By first applying polynomial fitting on the curves and then applying a PCA on the coefficients of the polynomials, we can decrease one more time the dimension of the output space from 6 to 4.

By keeping 4 principal components, the output space goes from 6 to the 4 dimensions and the precision is greater than 99.9% for at least 99% of the

observations. Here follows the decline of the explained variance per component as well as the cumulative percentage of the explained variance:

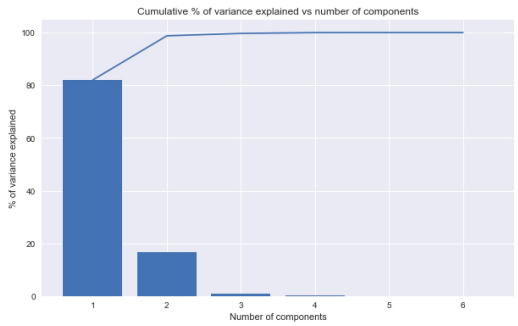

Figure 6: Cumulative percentage of the explained variance

In the following, we shall test the different dimensional reduction techniques above which will be compared to no dimensional reduction.

3 Regression based on Trees

In this section, different algorithms based on decision trees will be investigated. More precisely, the Classification and Regression Trees have been the source of numerous ensemble methods such as Bagging, Random Forest, the Gradient Boosting and AdaBoost and we explain how they work on the data we deal with. Recall we have at our disposal the 238t database of inputs which contains $\mathbf{X} = (X^1, ..., X^{24})$ where X^j are quantitative variables (i.e a A.C. parameter), and outputs are defined by $\mathbf{Y} = (Y^1, ..., Y^{29})$. For each individual, we observe a couple $Z_i = (X_i, Y_i)$ where $X_i = (X_i^1, ..., X_i^{24})$ and $Y_i = (Y_i^1, ..., Y_i^{29})$. We have thus a sample of observations of size $n = 28391$. The aim is to explain $\mathbf{Y}$ by a function of $\mathbf{X}$. For the sake of simplicity, we will consider the univariate regression $\mathbf{Y}^k$ (that is to say the value of the bending moment on the k^{th} station) by a function of $\mathbf{X}$.

3.1 Classification and Regression Trees (CART)

Classification and Regression Trees have been formalized by Breiman et al. (1984) and are decision trees. They consist of approximating a function F such as $F : \mathbf{X} \to \mathbf{Y}^k$. This algorithm considers all of 28391 observations and all of the 24 inputs. Let us recall how the method works (see Wikistat (2016)):

"*The construction of a tree consists in determining a sequence of nodes: a node is defined by the choice of one of the inputs (also called features) X^j and a division criteria which induces the partitionning of the sample linked to the node into two classes; a division criteria is defined by a threshold of the feature X^j; the root (also called initial node) corresponds to the full sample. As a consequence, the algorithm needs: a definition of a division criteria; a rule which tells when a node is terminal (it becomes thus a leaf); and the association of each leaf to a value of Y^k*".

"*A division (i.e the choice of the feature X^j and its threshold) is said to be acceptable if none of the two descending nodes is empty. The division criteria is based on the definition of a function of heterogeneity (the variance of the sub-sample), the objective being to split the sample into two groups as homogeneous as possible. The division of the node creates two nodes (left and right). Among all acceptable divisions, the algorithm keeps the one which minimizes the sum of the heterogeneity function of the two nodes*".

"*A tree stops growing to a certain node, which thus becomes a leaf, when it is homogeneous, or when there is no more acceptable division or when the number of observations of the sub-sample linked to the node is smaller than a predefined threshold. Here follows an example of construction of a tree*":

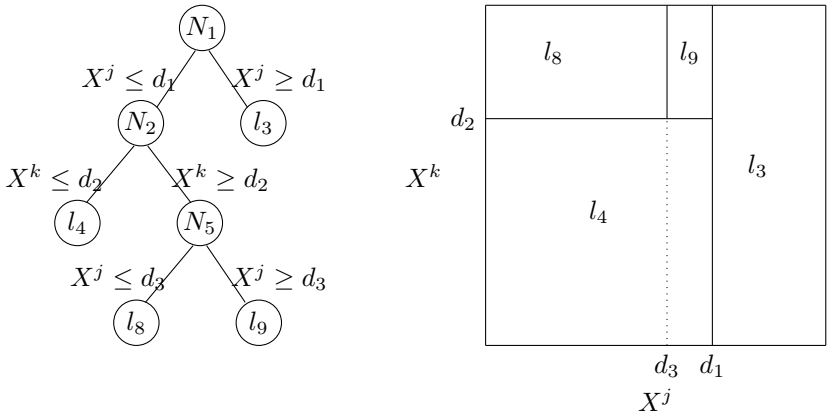

Figure 7: Example of construction of a tree Wikistat (2016) : Nodes are designed by N, and leaves by l

N_1 is the node containing all observations of $\mathbf{X}$, and other nodes or leaves contain a subsample of $\mathbf{X}$. Let called $\mathbf{X}_{l_j}$ the subsample of $\mathbf{X}$ that is contained by l_j. As a consequence, the value of $\mathbf{Y}^k$ associated to l_j is defined by :

$$\mathbf{Y}_{l_j}^k = \frac{1}{\#\{Y_i^k \in l_j\}} \sum_{i=1}^n Y_i^k \mathbb{1}_{\{Y_i^k \in l_j\}}$$

The value of Y^k associated to each leaf is then the average value of Y^ks associated to the sub-sample of the leaf.

At the end, this algorithm provides a huge tree with many leaves which can lead to over fitting. To avoid this effect, the tree must be pruned: we have to extract a sub-tree minimizing a generalization risk from the tree.

3.2 Bagging with regression trees

Bagging is an ensemble method to be applied if one wants to improve the qualities and capabilities of prediction of any algorithm. The following comes from Breiman (1996). Let us consider the full sample $\mathbf{X}$ of size $n = 28391$. From this n-sample, we draw randomly with replacement t n-sized samples from $\mathbf{X}$, called $\mathbf{X}_t$ and, for each $\mathbf{X}_t$, we train a predictor p_t (in our case, a tree-based algorithm). $\{p_1, ..., p_k\}$ is therefore an ensemble of predictors, predictors defined on different samples and the predictor on the whole training set is then defined by $\mathbb{E}[p(x, \mathbf{X}_t)]$, that is to say the average value of all predictions done by the predictors defined above. Sampling with replacement is most of the time associated to boosting sampling. The method explained above is named Bagging (stands for Boosting AGGregatING). Bagging improves predictions capabilities because it introduces differences between training samples which lead to variability of predictors. Breiman has shown that good candidates to boosting are classification and regression trees and neural networks.

3.3 Random Forest

Random Forests, introduced by Breiman (2001), are based on bootstrap sampling and CART. From the training set $\mathbf{X}$, t n-sized samples $\mathbf{X}_t$ are drawn with replacement (bootstrap technique used in the Bagging algorithm as well) and CARTs are fitted on each $\mathbf{X}_t$. When a tree is built, at each node of the tree, we draw randomly m inputs out of 24 (independently) and the optimal splitting criteria is defined through these m drawn variables. Trees grow to the maximal size and are not necessarily pruned.

Each tree is an estimator of the underlying function and built on a variation of the training set. As a consequence, each estimator leads to different results. Nevertheless, because of the numbers of estimators, the ensemble of trees (the forest), leads to a stable model. For a new observation, the prediction is then the average value of all the predictions of all predictors as in Bagging.

3.4 Gradient Boosting

The gradient boosting, intuited by Breiman (1997) and developed by Friedman (1999), is like every other boosting method: it combines weak learners. The goal stays the same, to explain $\mathbf{Y}^k$ by a function of $\mathbf{X}$ and instead of tuning parameters of this model, we iteratively add a model to the previous one to increase its capabilities. The name of "gradient" comes from the fact that the gradient of the squared error is the negative residual (see Friedman (1999) and Li (2016)). In our case, we use regression trees (CART). Here follows a simplified version of the Gradient Boosting Machine algorithm (for more details, see Friedman (1999)):

Algorithm 1 Simplified Gradient Boosting Machine

procedure GBM
    Fit a decision tree F_1 on $\mathbf{X}$ (resp. $\mathbf{Y}^k$)
    Compute the error residuals $e_1 = \mathbf{Y}^k - F_1(\mathbf{X})$
    for $t = 2, ..., T$ **do**:
        Fit a decision tree F on $\mathbf{X}$ (resp. e_{t-1})
        $F_t(\mathbf{X}) = F_{t-1}(\mathbf{X}) + F(\mathbf{X})$
        Compute the error residuals $e_t = \mathbf{Y}^k - F_t(\mathbf{X})$
    The model is then the sum of all fitted trees

3.5 AdaBoost

One thing that Bagging does not take into account is that each observation is not equally susceptible to be drawn randomly from the training set. Most of the time, we cannot assure this condition. As explained by Drucker (1997); 'in boosting, the probability of a particular example being in the training set of a particular machine depends on the performance of the prior machines on that example". In other words, if machine (a model) is able to predict and learn properly an observation, we do not need to learn more about it, but on observations which are difficult to learn on. Thus, these last ones will be more likely to be picked in a boosting sample. Adaboost was first introduced by Freund and Shapire (1995, 1996), and the following is a slightly modified version by Drucker (1997) called AdaBoost.R2:

Initially, each observation is assigned by a weight $w_i = 1$, $i = 1, ..., n$. The algorithm is defined this way and continues till the average loss $\overline{L}$ goes under 0.5:

Algorithm 2 AdaBoost.R2

procedure ADB
  for $t = 1, ..., T$ **do**:
    The probability that the observation i is in the training set is directly obtain by $p_i = \frac{w_i}{\sum w_i}$. Draw with replacement a n-sized sample $\mathbf{X}_t$ (and its corresponding output $\mathbf{Y}_t^k$) from the training set $\mathbf{X}$ (and $\mathbf{Y}^k$).
    Build a model F_t on $\mathbf{X}_t$ (resp. $\mathbf{Y}_t^k$) by making a weak hypothesis $h_t : \mathbf{X}_t \to \mathbf{Y}_t^k$
    Pass $\mathbf{X}$ to the model to get each predictions $F_t(\mathbf{X}_i), i = 1, ..., n$
    Calculate a loss for each observation. The loss may be of any form as long as $L \in [0, 1]$
    Calculate the average loss:$\overline{L} = \sum_{i=1}^{n} L_i p_i$
    Assessment of the confidence in the predictor by calculating $\beta = \frac{\overline{L}}{1-\overline{L}}$
    Update the weights $w_i \to w_i \beta^{1-L_i}$
  Outputs of each machine F_t are then weighted, and the predictor is the (weighted) median

Although this algorithm is noise and outliers sensitive, it does not need to be calibrated. This ensemble technique can be used with Random Forest and Decision Trees Regressors.

4 Prediction of loads for a new weight variant

4.1 Data preparation

Several options are possible to improve the capability of predictions of machine learning. For example, some of them are sensible to the homogeneousness of the data they learn from, or the number of input variables, as well as outliers. Concerning the last case, we cannot consider outliers because every load cases have been validated thus we must consider all of them. In the first part, we will focus on clustering of our load cases of gusts to improve the ML performance. In the second part, we shall analyze the influence of different dimensional reduction techniques on the generalization capabilities of several algorithms based on regression trees.

To improve the capability of machine learning algorithms, clustering has been performed on the gust cases. To take into account the form of each curve in the clustering process, the K-means algorithm has been performed on the data concerning the A.C. parameters as well as the coefficients of polynomials (respectively of 2nd and 2nd degree). To choose efficiently the number of clusters, a PCA has been performed and in the two first components, two clusters can be distinguished precisely. In the following, these two clusters will be referred as Cluster 0 and Cluster 1.

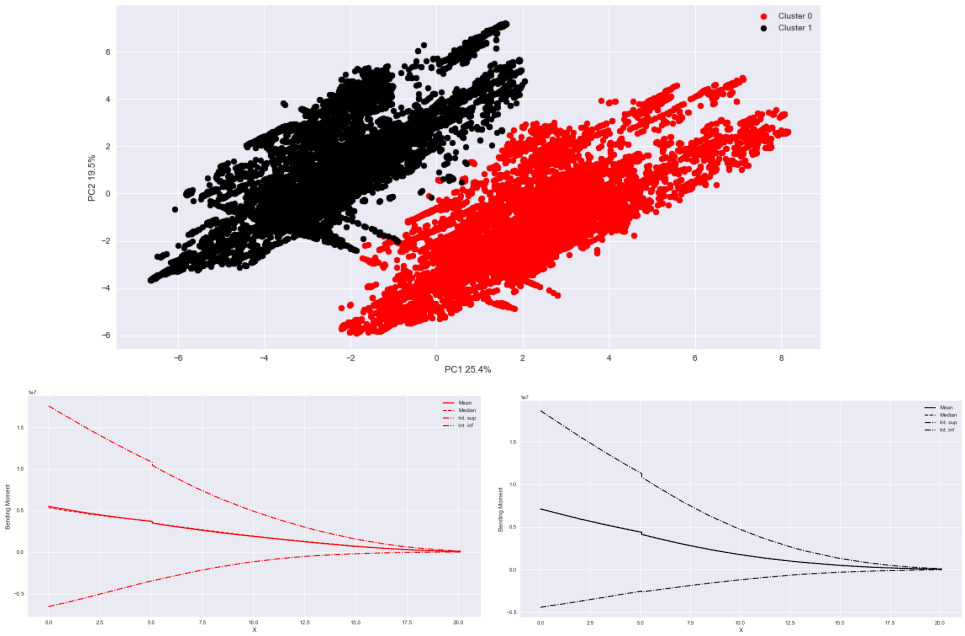

Figure 8: (a) Scatter plot of individuals in the two PC; (b)(c): Average, median, and confidence intervals of bending moments of clusters number 0 and number 1

As we can see above, the average bending moment of the Cluster 0 is more linear than the one of Cluster 1. Besides, the cluster 1 is constituted by bending moment which are mainly positive and with higher value at the wing root. By looking closer at the A.C. parameters, we can see that most of variables have the same distribution with a slightly different mean value. Nevertheless, some of them are really different: this is the case for DQ_DEGL1 (Deflection left inboard Elevator), DSP_DEG1L (Deflection Spoiler 1 Left Wing), DP_DEGIL (Deflection all speed Inner Aileron), DP_DEGOL (Deflection low speed Outer Aileron) and even more for ENXF (X-Load Factor Body Axis), especially the distribution:

	DQ_DEGL1	DSP_DEG1L	DP_DEGIL	DP_DEGOL	ENXF
Cluster 1	0.0043	-0.00025	-0.0082	-0.0079	-0.0587
Cluster 0	0.0258	-0.4363	-0.0495	-0.0488	0.0173

Table 3: Comparison of variables means in the two clusters

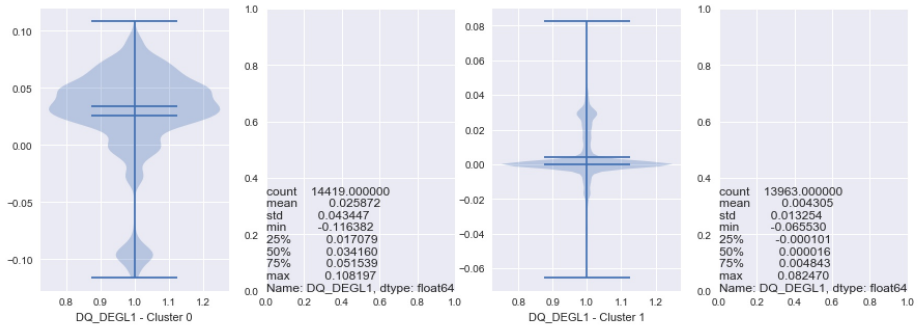

Figure 9: Comparison of DQ_DEGL1 for the two clusters

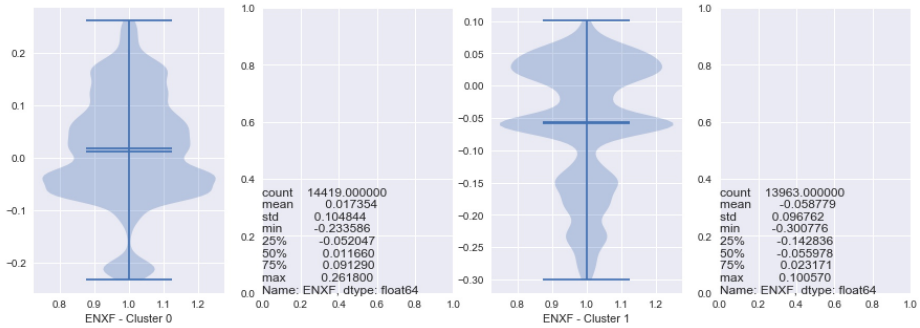

Figure 10: Comparison of ENXF for the two clusters

4.2 From 238t to 242t

Before presenting the results, it is important to explain more the R-squared score we have used in this project and why it is relevant in an engineering context. The R-squared, or also known as coefficient of determination, is a number that shows how well predictions are with respect to the explained variance. In other words, it is a measure of how well the model fits the data:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y})^2}{\sum_i (y_i - \overline{y})^2}$$

In our case, we calculate a R^2 at each station of the wing. Indeed, by doing so, we maintain the engineering sense of accuracy of a curve. Because the variance for one curve can be extremely high - for example, we have at the root a value of 8 000 000 and at the wing tip it is closed to 0 - calculating a R^2 on all the values at the same times would lead to over-estimate the accuracy of our models because the total variance is higher and thus, the ratio between the squared error and the variance is really low.

By calculating a R-squared at each station of the wing, we consider the variance only of the same kind of values. This has the disadvantage of being stricter concerning the results but is more accurate on how well our models are. The R-squared score given is then the average value of all R- squared calculated at each station.

To compare properly the results, from the 238t data set, we have drawn randomly a sample representing 80% of the observations, the last 20% represent the test set, and the 242t is our validation dataset, and we have repeated several time to see if a modification of the training set leads to unstable results in forecasting and generalizing.

To perform the comparison of algorithms presented above, we have used the scikit-learn library. Unfortunately, because we are trying to predict a field of vectors, just Random Forest is naturally implemented to do so and to take advantage of links which could exist between them. Then we used the MultiOutputRegressor for the other algorithms which, basically, create a regressor per output vector. As a recall, here are the algorithms we have tested the generalization capabilities. Adaboost based on decision trees regressors (ADB-DT); Adaboost based on Random Forest regressors (ADB-RF), Random Forest (RF), Bagging (BG) and Gradient Boosting (GB).

First, before checking the influence of dimensional reduction techniques we check which algorithms work the best on raw data:

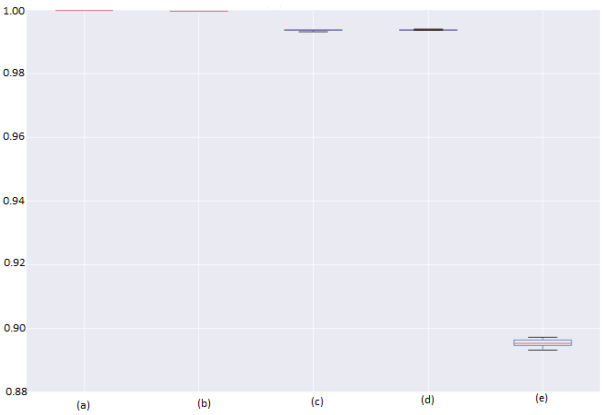

Figure 11: Boxplot of scores on random samples concerning Cluster 0 - Learning scores 238t-80% : (a) ADB-DT, (b) ADB-RF, (c) RF, (d) Bagging, (e) Gradient Boosting

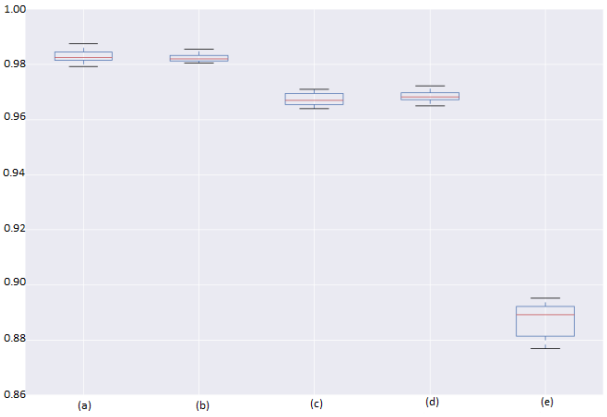

Figure 12: Boxplot of scores on random samples concerning Cluster 0 - Test scores 238t-20% : (a) ADB-DT, (b) ADB-RF, (c) RF, (d) Bagging, (e) Gradient Boosting

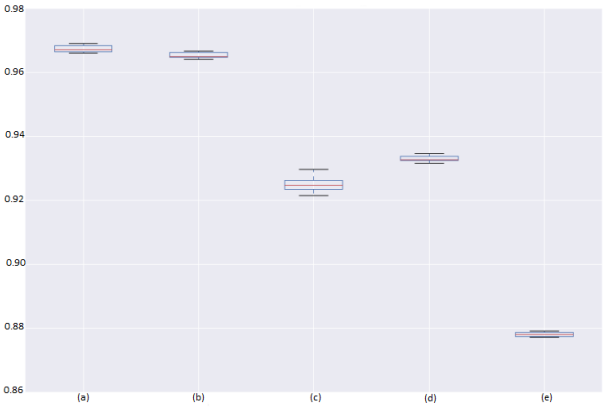

Figure 13: Boxplot of scores on random samples concerning Cluster 0 - Validation scores 242t : (a) ADB-DT, (b) ADB-RF, (c) RF, (d) Bagging, (e) Gradient Boosting

As we can see, even if AdaBoost is not able to predict and take into account several outputs, the one based on decision tree regressors gets the better results with a lower variability. Random Forest combined with AdaBoost has 4% higher scores with a lower variability than RandomForest only. It is important to notice that Adaboost (based on decision trees or Random Forest) is the method which has the less degrowth from the test score to the validation score (from 98.2% to 96.7%). As a consequence, we will focus now on the two first algorithms to see

the impact of dimensional reduction techniques: AdaBoost with decision tree regressors (ADB-DT) and AdaBoost with RandomForest (ADB-RF).

To quantify the influence of dimensional reduction techniques on extrapolation capabilities, here follows the different configurations we need to compare:

- (1) Raw inputs + raw outputs: no data transformation.
- (2) Raw inputs + PCA outputs: we keep the original input space and we perform a PCA on the output space.
- (3) Raw inputs + polynomial fitting: we keep the original input space and replace the outputs by polynomial coefficients.
- (4) Raw inputs + polynomial fitting and PCA: we keep the original input space and replace the outputs by polynomial coefficients on which we perform a PCA.
- (5) PCA inputs + Raw outputs: we keep the original bending moment and we perform a PCA on the input space.
- (6) PCA inputs + PCA outputs: we perform a PCA on the design space, and another on the output space.
- (7) PCA inputs + polynomial fitting: we perform a PCA on the design space and replace the outputs by polynomial coefficients.
- (8) PCA inputs + polynomial fitting and PCA: we perform a PCA on the design space and replace the outputs by polynomial coefficients on which we perform a PCA.

Concerning AdaBoost with Decision Trees regressors, we have:

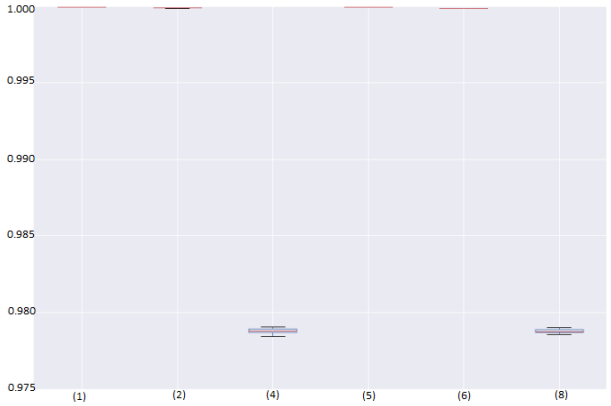

Figure 14: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - decision trees (ADB-DT) - Learning scores 238t-80% (indices are the number of the tested configurations)

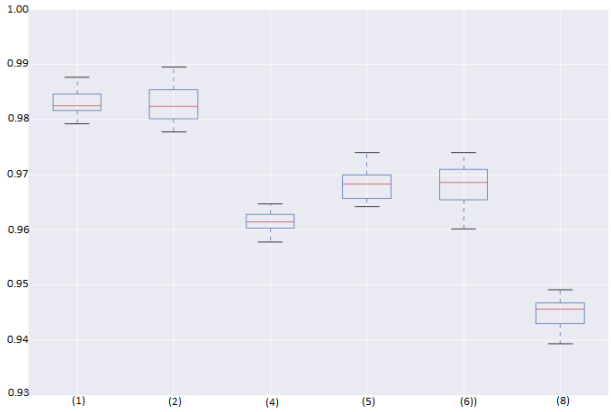

Figure 15: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - decision trees (ADB-DT) - Test scores 238t-20% (indices are the number of the tested configurations)

The polynomial fitting alone is not shown because Adaboost has revealed to be unable to learn properly on this kind of data. A PCA performed on the inputs does not improve results but reduce their variability. Nevertheless, we can see that the ADB-DT with a PCA on the outputs improves results when predicting the 242t.

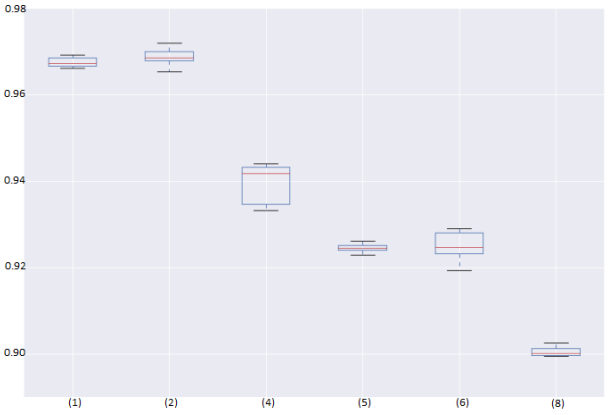

Figure 16: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - decision trees (ADB-DT) - Validation scores 242t (indices are the number of the tested configurations)

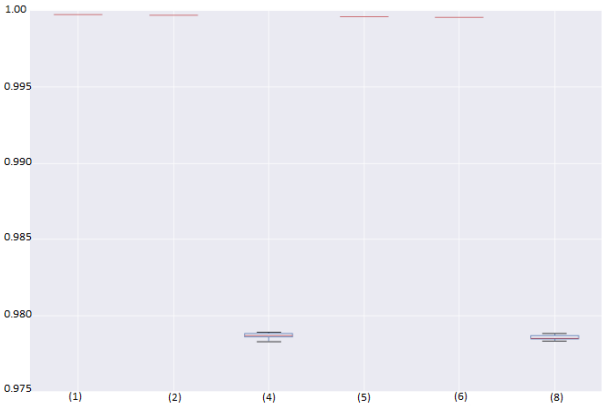

Figure 17: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - Random Forest (ADB- RF) - Learning scores 238t-80% (indices are the number of the tested configurations)

The results of ADB-RF are similar to ADB-DT. One major difference is the variability concerning the validation scores which is reduced against the others methods. From a cluster to an others, results concerning the variability and the type of algorithms are the same; just the scores change:

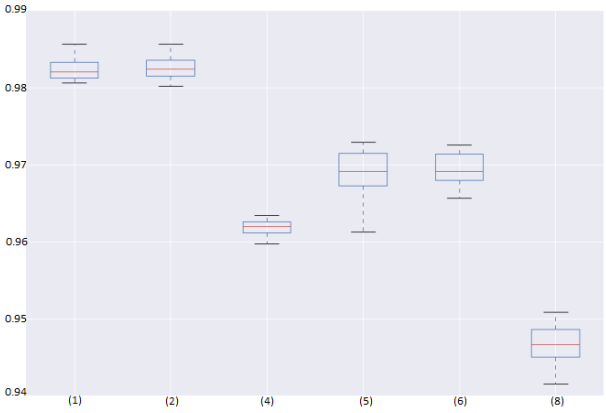

Figure 18: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - Random Forest (ADB- RF) - Test scores 238t-20% (indices are the number of the tested configurations)

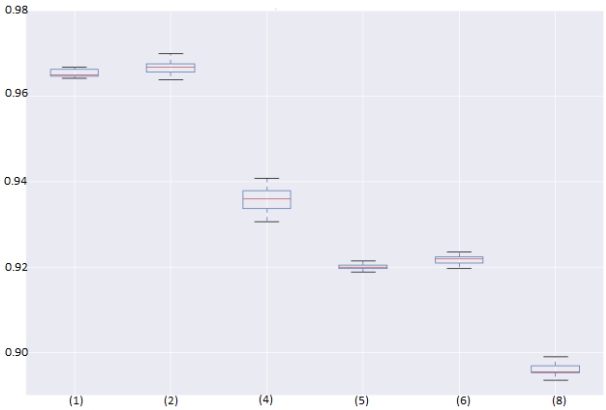

Figure 19: Boxplot of scores on random samples concerning Cluster 0 and AdaBoost - Random Forest (ADB- RF) - Validation scores 242t (indices are the number of the tested configurations)

	Cluster 0			Cluster 1		
	Learning	Test	Validation	Learning	Test	Validation
(1) ADB-DT	99.99/0	98.35/0.3	96.74/0.1	99.99/0	97.44/0.4	95.6/0.09
(2) ADB-DT	99.98/0	98.30/0.4	96.85/0.2	99.98/0	97.51/0.5	95.72/0.2
(1) ADB-RF	99.97/0	98.29/0.2	96.54/0.09	99.97/0	97.51/0.3	95.56/0.08
(2) ADB-RF	99.96/0	98.3/0.2	96.66/0.17	99.96/0	97.53/0.4	95.62/0.09
(1) RF	99.37/0.03	96.74/0.2	92.51/0.2	99.17/0.04	95.7/0.4	92.17/0.4
(2) RF	99.38/0.03	96.91/0.2	92.78/0.18	99.21/0	95.88/0.3	91.77/0.6

Table 4: Mean/standard deviation of scores after cross-validation

AdaBoost with Random Forest or Decision Trees are similar but the variability. Nevertheless, we can assume now that a PCA on the outputs improves the results and from now, we shall investigate how are the error distributed to understand better the lack of generalization capabilities of our model. In the following, just AdaBoost with Random Forest will be investigated.

To calculate the error ratio per curve of bending moment, the following formula has been used:

$$error = \frac{\sqrt{\int_0^L (f(x) - \hat{f}(x))^2}}{\sqrt{\int_0^L (f(x))^2}\sqrt{\int_0^L (\hat{f}(x))^2}}$$

Where L is the length of the wing. By doing so it allows us to have a physical idea of how far our predictions are.

90% of the predictions of the 242t have an error lower than 7%, 88% with an error lower than 5%, and 78% with an error lower than 2%; nevertheless, the average error of the whole 242t dataset is 2.79%, that means that in average the score is 97.21% for Cluster 0.

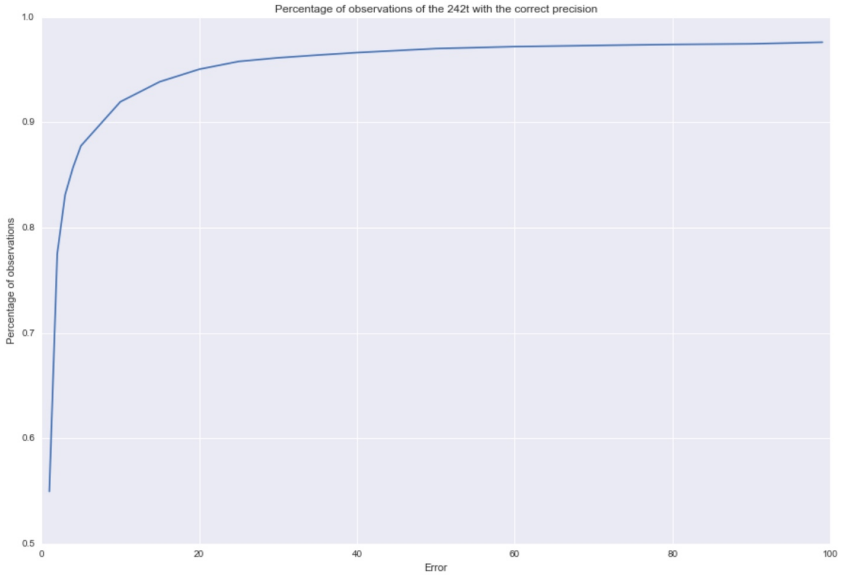

Figure 20: Percentage of observations of the 242t vs. Error Rate

It is interesting to understand why some points are really bad predicted. In the first place, we shall look at the mean curve whose prediction error is above 10% and we can see that the curve which are not well predicted belong to the whole space and does not occupy a certain subspace; as a consequence, the mean curve is almost null:

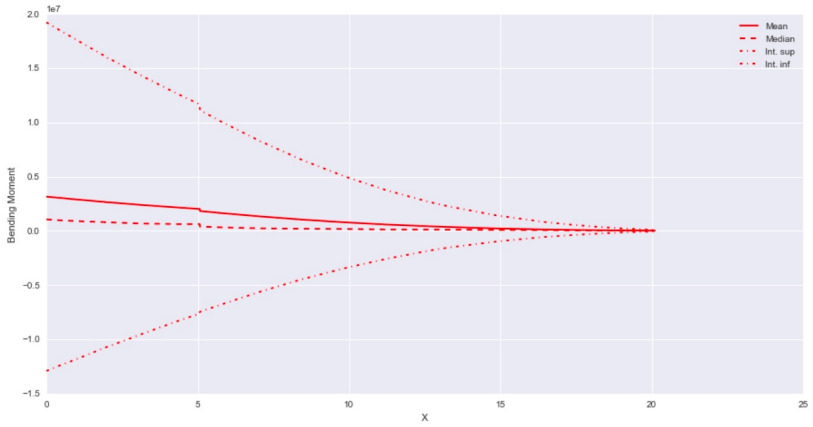

Figure 21: Mean curve of badly predicted bending moments of the 242t

By performing a PCA and plotting the A.C. parameters of the training set (238t), as well as projecting the A.C. parameters of the 242t onto the first components, we can see that the two first components do not help to understand better the space distribution of the input parameters. Nevertheless, we can see that the error grows in subspace were the density of points on the training set is high. It is even more flagrant when you look at the projection in the second and third component:

This high density value but high error ratio can be explained by the fact that slight changes in the A.C. parameters (especially total weight) leads to very different bending moments and by the fact that the design space is not well distributed. Some points have a lot of weight in the PCA as we can see above, and because we can not consider them as outliers, it makes difficult the model to learn properly and have a better distributed design space. Besides, the total weight of the aircraft (which is the crucial key of the analysis) has not as much importance as it should be in the model and in the PCA; as a consequence, load cases estimation leads to hazardous results for some dense zone.

4.3 From 238t to 251t

In order to know if decision trees algorithms are able, in our case, to generalize properly the problem we are dealing with, we shall focus on testing the best

Figure 22: (238t (blue) and 242t (black) A.C. in the first and second PC - red points correspond to observation whose error is above 5%

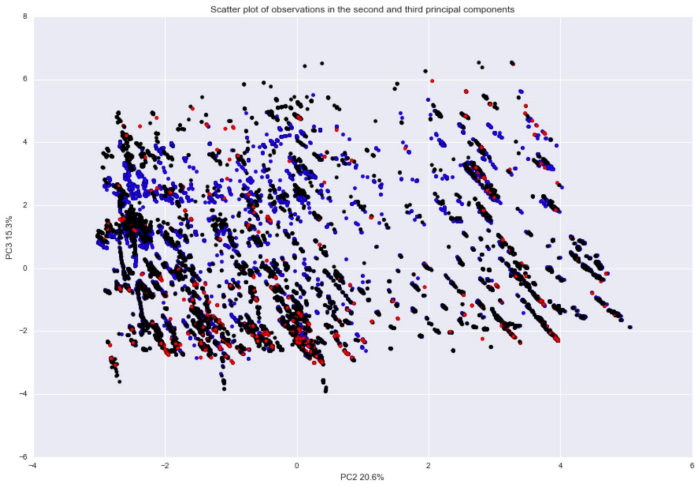

Figure 23: (a) 238t (blue) and 242t (black) A.C. in the second and third PC – red points correspond to observation whose error is above 5%

two algorithms, that is to say Adaboost based on decision trees and based on Random Forest with both combining with a PCA on the outputs. As previously explained, we shall, for a random sample of the 238t data set, create a model, test it on the rest of the 238t dataset, and predict the 242t, 247t, and 251t cases. This process is repeated several times for a different sample to show the

variability and the stability of our model. The shape of the data set of the 247t and 251t are similar to the 238t and 242t.

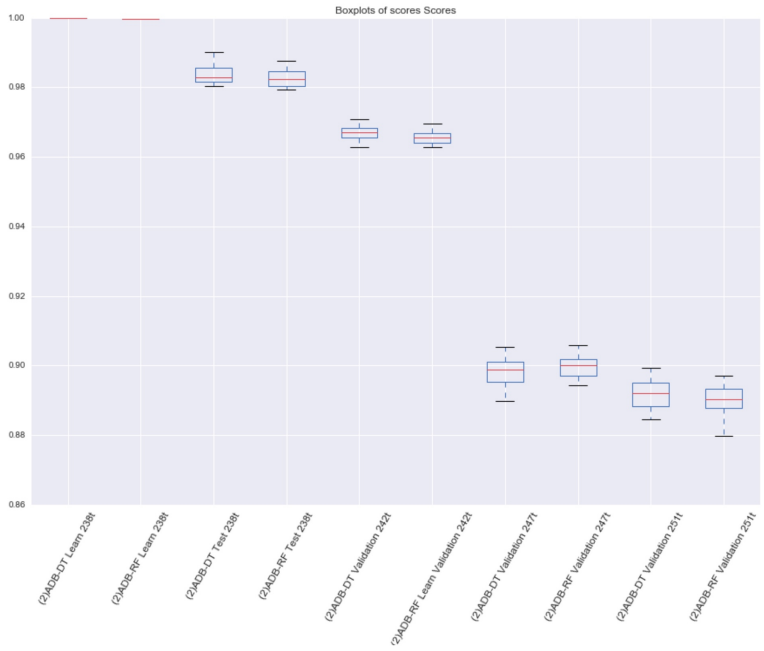

Figure 24: Results of generalization (boxplots of scores) concerning AdaBoost with Decision Tree Regressors and Random Forest Regressors

The quality of generalization drops for points far from the training data set and also for some really dense zones. As a consequence, concerning the 247t data set, only 49% of the predicted observations have an error smaller than 2%, 78% for an error under 5% and 95% of the observations have an error smaller than 35%. For the 251t, only 48% of the predicted observations have an error smaller than 2%, 78% for an error under 5% and 95% of the observations have an error smaller than 30%.

5 Conclusion

Let us highlight now the contribution of this case study. As mentioned above, AdaBoost associated with Random Forest gives excellent results for observations which are not far from the training set (78% with an error less than 2%) and close to 95% of load cases have an error under 10% and in average we get a score above 97%. This is even more accurate when the outputs have similar

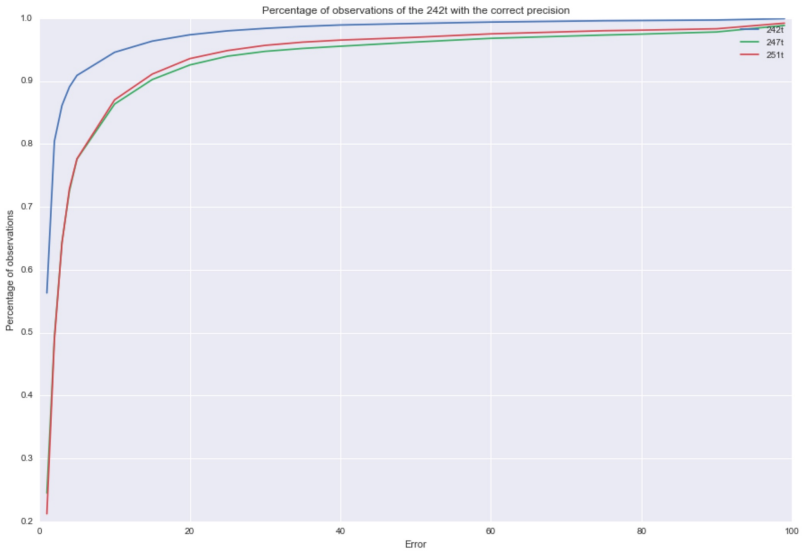

Figure 25: Percentage of observations of the 242t, 247t, 251t vs Error rate

forms for close design points and for load cases that are not impacted by the weight change roughly. As soon as we try to generalize the results for observations far from the learning data set or for load cases which leads to different behaviour, results drop. If we control the design space at the starting point, or add information concerning the form of the load to predict, or place us in an interpolation context, results would be even better.

A PCA on the outputs improves the results in average, and this can be explained because of the high co linearity of the outputs. Because of the presence of outliers and especially because all inputs matter, a PCA on the input space does not improve our results in average.

By trying to predict a vector (the shape of our training matrix is 28931x53) and not a point (it would have been 838 999x25), the speed of learning is exponentially decreased, and we keep the engineering information of the mathematical object.

Upcoming works concerning this project should investigate the following point: define a reliable method for extrapolation; test other dimensional reduction techniques as the shape invariant model approach such as defined by Sergienko et al. (2012) which has been used in the petroleum industry; produce data in sub-spaces where there is a lack of information; investigate the fact that the optimal parameters obtained are maybe not optimal in term of

generalization; consider other machine learning algorithms than those based on regression trees because they are known to be not optimal in a generalization problem, because they are considered as "black-boxes" and because they do not give uncertainties; considering on-line learning: as soon as a new observation is available, the model should keep learning sequentially.

Airbus pursues the increasing knowledge capitalization and the development of new methods and tools for Research and Engineering through Big Data initiatives and the promising results of the sprint project, in which this case study has been achieved, are part of the root of upcoming bigger projects about Machine Learning in the load and stress process.

References

Airbus, C. A. (2017). A330 family. Available from : http://www.aircraft.airbus.com/aircraftfamilies/passengeraircraft/a330family/.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Breiman, L. (1997). Arcing the edge. *Technical Report*, (486). Statistics Department, University of California.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Doherty, D. (2009). Analytical modeling of aircraft wing loads using matlab and symbolic math toolbox.

Drucker, H. (1997). Improving regressors using boosting techniques. *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 107–115.

Freund, Y. and Shapire, R. (1995). A decision-theoretic generalization of on-line learning and application to boosting. *Proceedings of the second European Conference on Computational Learning Theory*, pages 23–37.

Freund, Y. and Shapire, R. (1996). Experiments with a new boosting algorithm, machine learning. *Proceedings of the Thirteenth Conference*, pages 148–156.

Friedman, J. H. (1999). Greedy function approximation: A gradient boosting machine.

Gandomi, A. and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *Internation Journal of Information Management*, 35:137–144.

Hjelmstad, K. D. (2005). *Fundamentals of Structural Mechanics*. Springer US.

Hoblit, F. M. (1988). *Gust Loads on Aircraft: Concepts and Applications*. AIAA Education Series, AIAA.

Hotelling, H. (1993). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 23:417–441 and 498–520.

Li, C. (2016). A gentle introduction to gradient boosting. College of Computer and Information Science, Northeastern University. Available from: http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf.

Manyika, J. and al. (2011). Big data: the next frontier for innovation, competition and productivity. Mc Kinsley Global Institute.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572.

Sergienko, E., Gamboa, F., and Busby, F. (2012). Shape invariant model approach for functional data analysis in uncertainty and sensitivity studies.

Wikistat (2016). Arbres binaires de décision — wikistat. Available from: http://wikistat.fr/pdf/st-m-app-cart.pdf.