Machine Learning for Predicting Flight Ticket Prices

André Cibils,
AI-Lab, Ecole Polytechnique Fédérale de Lausanne

ABSTRACT **:** By using machine learning and in particular linear and non linear regression, we predict flight tickets prices. For this purpose we are using data collected during the past 6 months. We explore multiple features such as the time before the flight, the month of the flight and route of the flight to arrive to an acceptable prediction. The trend is that ticket prices are minimum roughly two months and one year before the day of the flight.

1 . INTRODUCTION

The main purpose of this paper is to find a common trend in the evolution of the prices of flight tickets. In this context, the use of machine learning is highly appropriate,so a lot of data has been crawled during the past six months, as data is one of the most important ressource in order to have valid results.

The mechanisms used by the companies to change the prices of flights tickets over time are not transparent. Many parameters are not available, such as the number of remaining places in the airplane. We can expect the price to be high when we reach the date of the flight or when the flight just got available, but also that the ticket will cost more in the summer or in December (due to the vacations) than in October or November.

For this problem, the start was to use simple models such as linear regression and multilinear regression to find a common trend as a first solution and have a first impression of the result, and in a second time more complex models like the ridge regression can be used in order to improve the quality of the results.

Multiple features will be explored to improve the model, the principal and most important one being the time remaining before the flight. The month of the flight, the route (ie from where to where is the flight, for example Budapest to Barcelone), the duration of the flight are some features we will explore.

2 . MATERIALS AND METHODS

2.1 Data Collection

The datas are separated in two sets. For each of these sets, everyday at 10:00, a crawler extract data from https://wizzair.com/en-GB/TimeTable. For the first set, the crawling began the 9th of November 2015 and it gathered informations of eight different routes. The second set has data only from the 4th of January 2016 but has information about twelve others routes. The dataset is merged once the useful data have been parsed. For both set, the crawling ended the 8th of Mai 2016.

Each day, the crawler will go through the entire timetable and look for each flight availabe to gather informations about those flight in a JSON format. The information hold in this JSON is multiple, and the first step is to extract the useful data from these.

The parser extracted relevant information from the JSON and wrote them as csv. Depending on the stage of the project, the information is for instance the time before the flight (translated in Unix time), the precise time of the flight, the route, the month. Once the parsing is done, we end up with a total of 1'031'219 features vectors, ie the number of line in our csv.

However, if two flights go off the same day, the site doesn't give us enough information : we can only know the lowest price of the two flights. We decided to drop one of the two flight, and we won't considered the hour of the flight as a feature.

No identification number has been assigned to a flight since there are not two plane going off at the exact same time. The time when the flight go off (in Unix time) could be considered as an identification number.

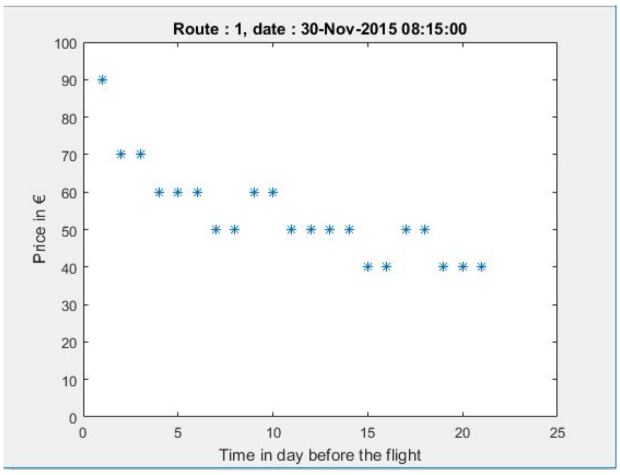


Fig1 : Example of data for a given flight

2.2 Linear & Multilinear Regression

The goal here is to find a valid estimation of the common trend by using linear & multilinear regression. Our hypothesis is really simple at first : it takes the date before the flight as input and give the expected price as output, more formally hθ(x) = θ * x, where θ is a vector representing the parameters discovered during the training phase and x is the vector representing the features (Only the date before the flight here, but more features are added as the project goes on).

The complexity of the x vector is defined as the number of degrees of the polynomial equation of our hypothesis. We can indeed create new features by taking the power of 2,3 or even 4 of our main feature : the time before the flight. This allow us to have different shape of curves. However, having a high complexity is dangerous for the validity of our results as it can leads to overfitting.

To start, we divide our data set into two sets, the training set and the test set (respectively 70% and 30%). The first one will be used to train our program and the second one to verify the validity of the hypothesis on new, fresh, unused data. This is primordial to validate our hypothesis.

In order to proceed, we will define a cost function J(θ) :

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

where m is the number of our training examples, x is our features and y our observations. For this simple model, x is the time before the flight and y is the price.

This cost function is the least-square estimation and force the hypothesis h(x) to be as close as possible to our training example [1]. Indeed, the cost function punishes heavily big differences.

Our aim is to find the minimum of this function ie. the parameters θ such that J(θ) is as small as possible. To find this minimum, we use the gradient descend algorithm :

```
Repeat {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$
} (simultaneously update for every $j = 0, \ldots, n$)
```

Let α is the learning step, carefully choosen. A too big learning step will lead to non convergence of the algorithm, but a too small one will make the program really slow.

The gradient descend algorithm does not give the absolut minimum but only a local one, which is fine in practice. We stop the algorithm when |θ_new – θ_old|<threshold.

For the training, our data are normalized between zero and one.

Once our parameters are settled and clear we proceed to test our hypothesis with the test set.

2.3 Ridge Regression & Regularization

The next step is now to improve our model. To do so, we will use a technique called regularization. This is used to reduce the variance, ie the overfitting problem.

Overfitting appears when our hypothesis fits to well to our training data, which can mean that our complexity is too high. To add regularization in our model, we need to add a term to our cost function :

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

This poses a problem, which is finding the right λ. For this, we will redivide our data in 3 sets this time. To make the model more robust, the k-fold method be used (Fig 2) [2].

20% of the data will be put aside and called, once again, test set. We will loop 10 times on the 80% data remaining, each time we select 80% of this data to be our training set, and 20% to be our cross validation set.

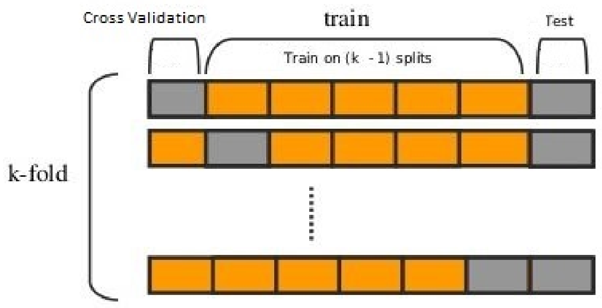


Fig 2 :K-fold technique

The cross validation set will be used to select λ : for each λ, we find the optimized θ and we compute the cost associated with those parameters (the data used to calculate this cost are the data from the cross validation set : like the test set, this data haven't been used to train our program).

Then, we select the better one by simply taking those who have the lowest cost.

We end up with ten sets of parameters, which we can compare with one another to find some unexpected behavior (see Appendice 2). Our parameters will be the average of the ten parameters found by the k-fold method.

2.5 K-mean Clustering of Multilinear Regression Models

An other interesting thing to look over would be to see if we can separete the routes into two clusters.

For this, we apply the k-mean algorithm [3] too the datas. To start, we randomly assign each x to a cluster. Then, we apply both our multilinear regression models to the set of data, and calculate for each x the corresponding error (using the cost function defined earlier). We reassign each x to the cluster whose parameters give the smallest training error. This is done multiple time (until convergence, ie the number of x assigned to a different cluster is non significant).

After that, the idea is to look for each x to which cluster and which route it belongs, and then assign one trend to a cluster of routes and the other to the rest of the routes.

We obviously don't take the routes as features here.

2.5 Explored Features

In order to have a strong and complete model, we will have to choose the right features.

The most obvious and important one is the delay before the flight. By augmenting the complexity of the hypothesis, we can change the shape of our trend, but we need to be careful not to have a too big complexity as it would lead to overfitting. We can use the test set or the cross validation set to select the right number of degrees for the polynomial equation.

Other features considered are the month were the flight takes off, the duration of the flight, the route of the flight, if the flight takes off during the week or the weekend. After that, we need to analyse the impact of these parameters.

The month and the route are so called dummy variables, ie boolean. They will note impact the shape of the curve but only make the general price higher or lower.

3 . RESULTS

3.1 Useful Features

As we can see in Fig 3, too much complexity will lead to a drastic augmentation of our error. The right number seems to be three.

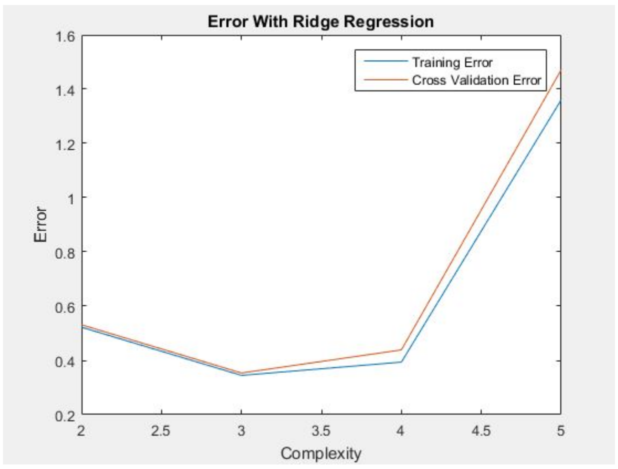


Fig 3 : TrainingError and CrossValidationError depending on the complexity

The month of the flight and the route are very impactant parameters, but as said before, they can not change the shape of the curve (see in Fig 4).

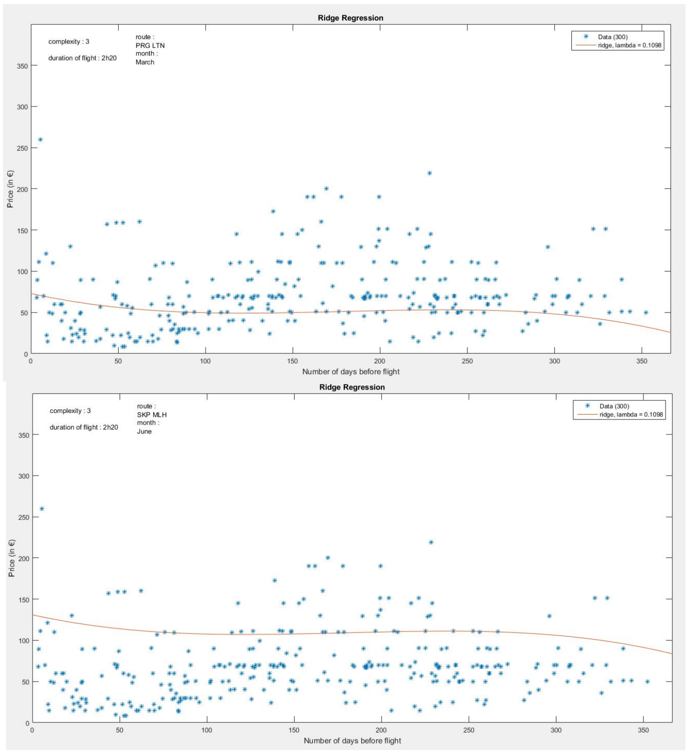


Fig 4 : Influence of the month & route parameters

3.2 Results Of The Linear & Multilinear Regression

The results of this methods are roughly correct and doesn't suffer too much of overfitting. It points that the two key moment to buy a ticket seems to be either two months or one year before the date, without considering anything else than the number of days remaining before the flight (see Fig 5)

After the data got normalized between zero and one, the training error is 0.0589 and the test error is 0.0632, which is 7% more. Our model overfits a little to our data but the regularization fix it.

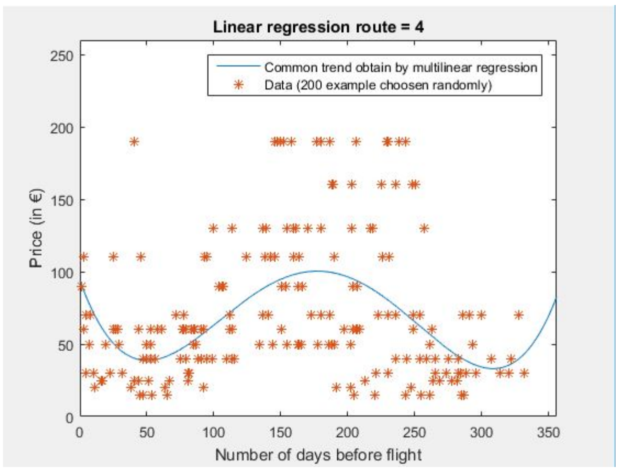


Fig 5 :Results of the multilinear regression,

h(x) = $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

3.3 Result of Ridge Regression & Regularization

Ridge regression offers us a lot more of parameters to choose, and the results depends highly of this parameters. We used some tools to help us to select those (see Fig 6)

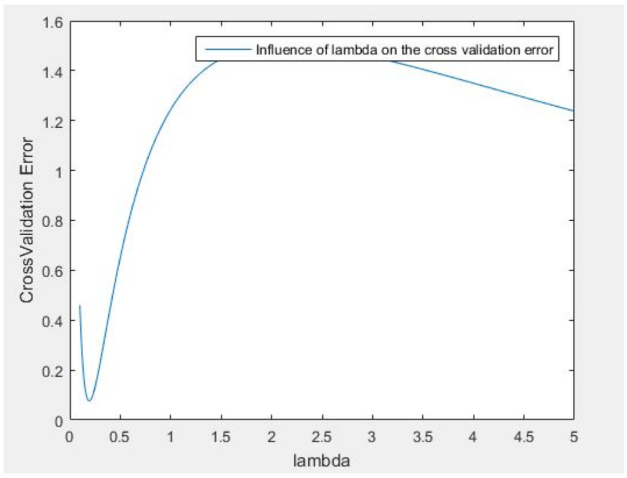


Fig 6 : Influence of λ *in Cross Validation Error, with the optimized* θ*.*

The trend obtained by the ridge regression is better than the one from the multilinear regression, the shape is a lot less eratic (see Fig 7).

The value of our parameters are much closer to zero than the parameters of the multilinear regression (compare Appendice 1 and 2).

The test error is lower using the same features : 0.015915 for the ridge regression against 0.0665 for the multilinear regression, which is 4.18 times more. The regularization worked well and reduce a lot the test error.

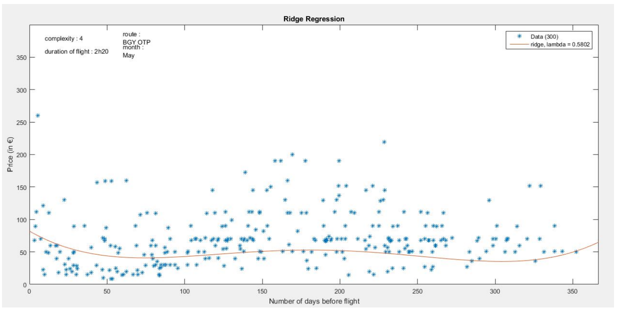


Fig 7 : Results of the ridge regression

3.4 Result of K-mean Clustering of Multilinear Regression Models

The clustering of the routes doesn't offer valid results. The clusters are changing every time we compute, so can't find an absolut convergence. There is no two trends in the data that we can find with certitude this way, as they highly depends of the random starting clusters.

The number of route in the clusters are moreover often one or two routes in a cluster, and the rest in the other cluster.

Moreover, the shape of the curves seems often irrealist, we can see in Fig 8 that for the route belonging to the yellow trend, the best moment to buy a ticket would be the day of the flight, at the very last minute.

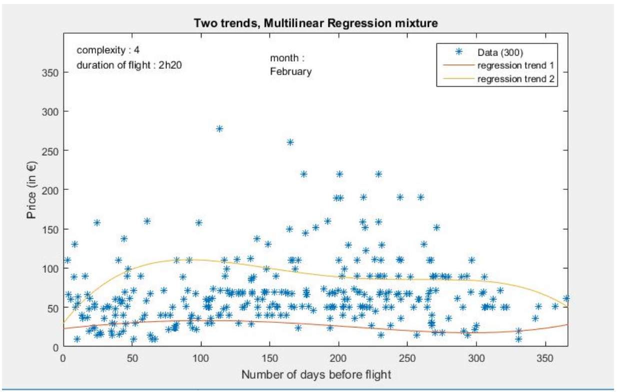


Fig 8 : A result of the clustering of the routes by the k-mean algorithm

4 . DISCUSSION

The results seems to validate our hypothesis, the trend found is correct. The best moment to buy a ticket is two months or one year before the date of the flight. We notice that the route or the month of the flight we are choosing have a huge influence on the price.

The parameters are interesting (see Appendice 2) : The important feature is definetly the time before the flight.

In order to improve our result, we should take more parameters in sight, such as if the flight is during the week or during the week end.

We tried to associated the routes to two different clusters, but the method used didn't gave us robust results.

6 . FUTUR WORK

The method we used earlier to separete the routes into 2 clusters gave us results whom highly depends of the random start, but the idea seems promising.

In the futur, we can try to use others models, like the overlapping mixtures of Gaussian processes [4]. Gaussian processes are models who discover θ parameters by using more probabilistic methods. The ovelapping mixtures of this processes is a complicate process that we can explore another time.

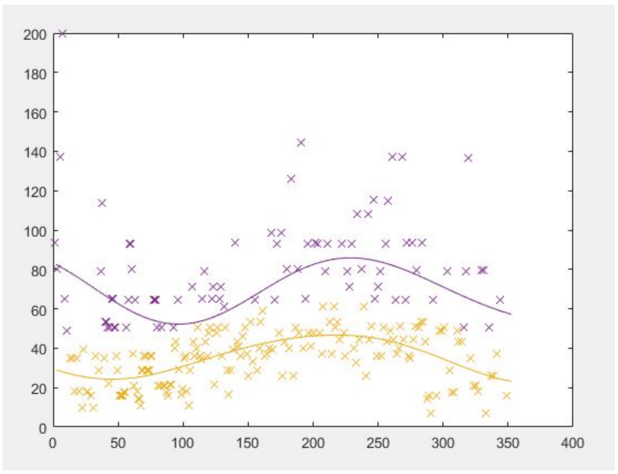


Fig 9 : Two different trend found in the data with the Ovelapping Mixture of Gaussian Processes

5 . ACKNOLEDGMENTS

I would like to thanks my project responsible, Mr. Igor Kulev for his precious advices and for his guidance. He also provided me the data, I couldn't have make it without him.

I would also like to thanks Mr Boi Faltings, director of the Artificial Intelligence Laboratory, who gave me the infrastructure to do this project, and for his course where I learned a lot about k-mean and others ia topics which have been useful here.

6. LITERATURE CITE

[1] : *Least Squares Estimation*
http://stat.ethz.ch/~geer/bsa199_o.pdf
[2] : *Cross validation*
https://en.wikipedia.org/wiki/Cross-validation_%28statistics%29
[3] : *K-mean clustering*
https://en.wikipedia.org/wiki/K-means_clustering
[4] : *Ovelapping Mixtures of Gaussian Processes*
http://www.squobble.com/academic/doc/lazarogredilla2012_omgp.pdf

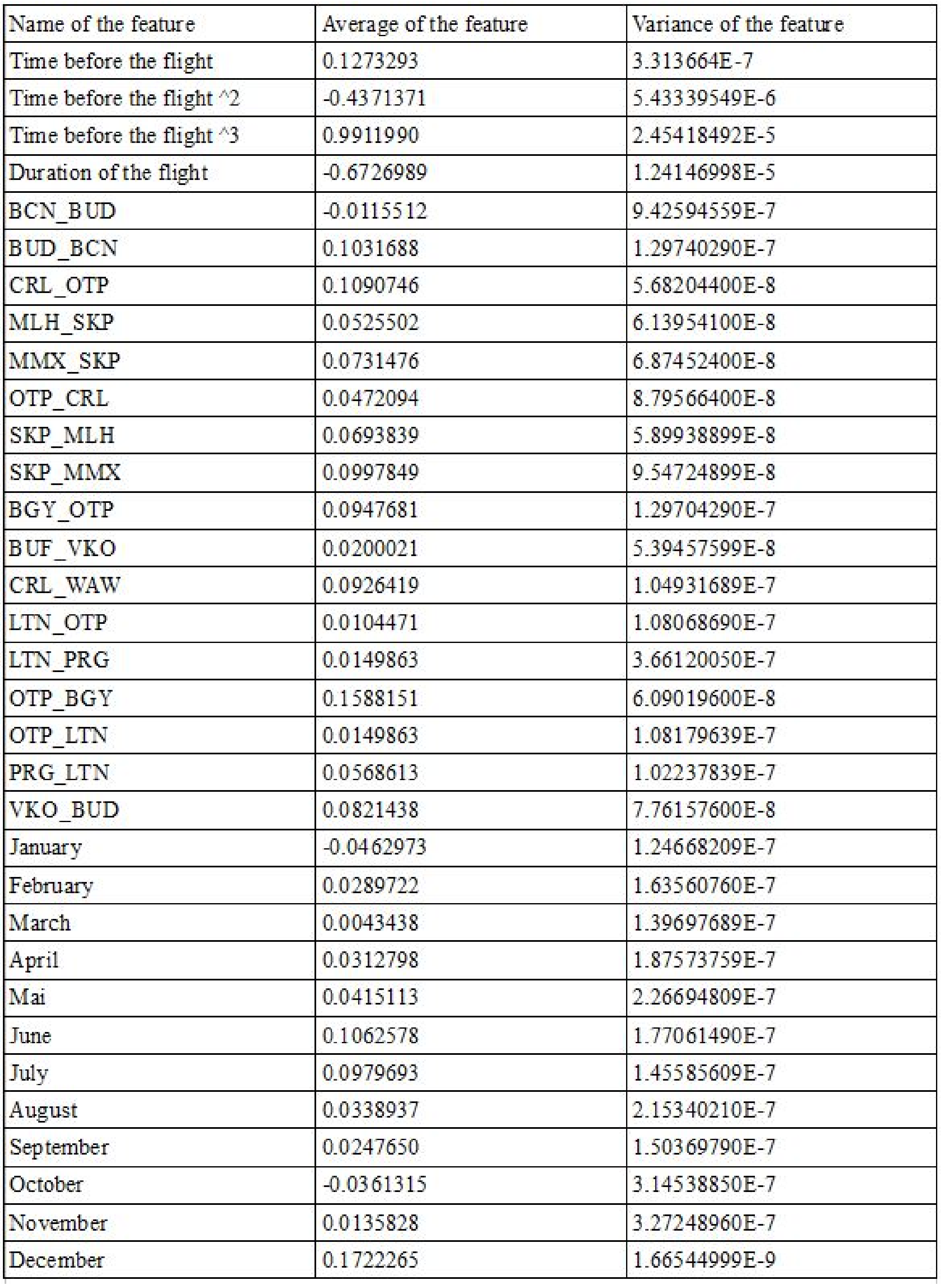
7 . APPENDICES

Appendice 1 : parameters value for the multilinear regression

Name of the feature	Value of the feature	Lower bound of the 95% confidence inteval	Upper bound of the 95% confidence interval
Date before the flight, degree 1	-3,44	-3,63	-3,24
Date before the flight, degree 2	17,48	16,67	18,3
Date before the flight, degree 3	-27,97	-29,25	-26,7
Date before the flight, degree 4	13,87	13,2	14,53

Appendice 2 : Value and variance of our parameters for the ridge regression

This parameters have been found with normalized data (0-1), and validate by the k-fold cross-validation method.



Name of the feature	Average of the feature	Variance of the feature
Time before the flight	0.1273293	3.313664E-7
Time before the flight ^2	-0.4371371	5.43339549E-6
Time before the flight ^3	0.9911990	2.45418492E-5
Duration of the flight	-0.6726989	1.24146998E-5
BCN_BUD	-0.0115512	9.42594559E-7
BUD_BCN	0.1031688	1.29740290E-7
CRL_OTP	0.1090746	5.68204400E-8
MLH_SKP	0.0525502	6.13954100E-8
MMX_SKP	0.0731476	6.87452400E-8
OTP_CRL	0.0472094	8.79566400E-8
SKP_MLH	0.0693839	5.89938899E-8
SKP_MMX	0.0997849	9.54724899E-8
BGY_OTP	0.0947681	1.29704290E-7
BUF_VKO	0.0200021	5.39457599E-8
CRL_WAW	0.0926419	1.04931689E-7
LTN_OTP	0.0104471	1.08068690E-7
LTN_PRG	0.0149863	3.66120050E-7
OTP_BGY	0.1588151	6.09019600E-8
OTP_LTN	0.0149863	1.08179639E-7
PRG_LTN	0.0568613	1.02237839E-7
VKO_BUD	0.0821438	7.76157600E-8
January	-0.0462973	1.24668209E-7
February	0.0289722	1.63560760E-7
March	0.0043438	1.39697689E-7
April	0.0312798	1.87573759E-7
Mai	0.0415113	2.26694809E-7
June	0.1062578	1.77061490E-7
July	0.0979693	1.45585609E-7
August	0.0338937	2.15340210E-7
September	0.0247650	1.50369790E-7
October	-0.0361315	3.14538850E-7
November	0.0135828	3.27248960E-7
December	0.1722265	1.66544999E-9