Applied Data Analysis and Machine Learning

Report Project – 1

Paul Giraud

Table des matières

Introduction:

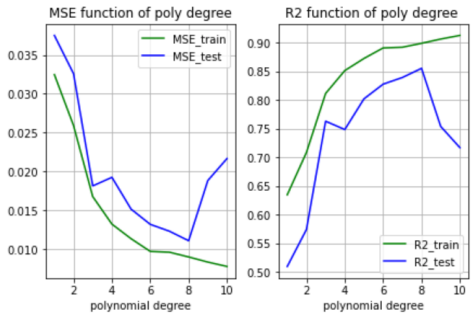
In this project we will look at different methods of regression, OLS, Ridge and Lasso and using with them bootstraps and Cross-Validation.

During the five first exercises we will apply them on the Franck Function, defined for x in [0, 1] and y in [0, 1].

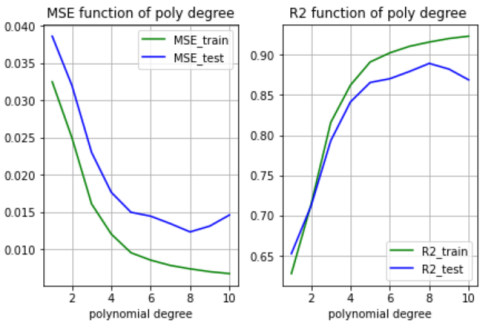
In the sixth exercise, we will apply all of those methods on a real set of data, which represents the topology in a Norwegian region.

Exercise 1:

In exercise 1, we are using the ordinary least square on the Franck Function. This function is represented for x and y between 0 and 1. Hence, our data are already normalized. That is why in the last version of my code I do not scale my data.



Without scaling



With scaling

When I scale the data, I use the Sklearn function, nevertheless we just have to subtract the mean of our input and divide by the variance.

In my code I plot all the regressions for each polynomial degree to see how our solutions seem to be close to the Franck function.

Moreover, I just keep in a matrix all the different coefficients beta for each polynomial degree regression to see how each coefficient of beta are changing when we increase the degree of the regression. Hence, we see more clearly if we are over-fitting or not. In fact, we are next to a good regression when our coefficients do not change a lot when we are increasing the degree, and we often over-fitting when it's varying a lot.

---------- beta coefficient for each polynomial degree ----------

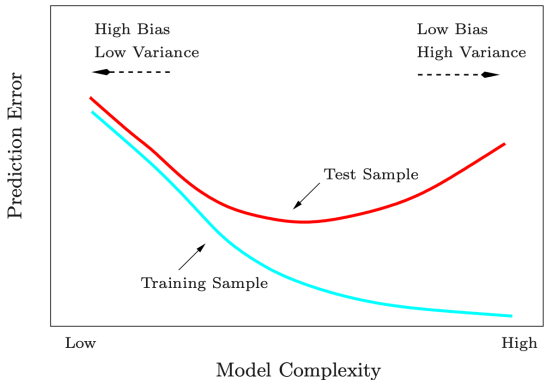
	1	X^1	Y^1	X^2	XY	Y^2	X^3	X^2Y	XY^2	Y^3	...	X^9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
1	0.960742	-0.485191	-0.636651	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
2	1.116584	-0.909314	-0.694031	-0.025443	0.878268	-0.377316	0.000000	0.000000	0.000000	0.000000	...	0.00000
3	0.948178	-0.433298	1.380690	-1.461931	1.604734	-6.283239	0.889678	0.208517	-0.942351	4.242011	...	0.00000
4	0.713399	2.916810	2.876087	-15.022051	-1.587229	-10.913895	20.319117	4.231107	3.324544	9.478897	...	0.00000
5	0.623438	5.643032	1.996252	-27.933164	-8.697588	-0.239973	43.533130	27.675955	11.685067	-24.893000	...	0.00000
6	0.738386	0.054209	1.732153	22.754672	1.153198	2.508840	-143.893915	-31.993201	21.500458	-49.333000	...	0.00000
7	0.913384	-6.522423	-0.656071	78.544850	53.178271	9.219732	-350.002833	-283.826142	-146.559369	-30.592329	...	0.00000
8	0.887184	-0.232186	-3.650523	-34.000055	64.537703	50.281326	380.184628	-126.047584	-373.171438	-244.370353	...	0.00000
9	0.801101	6.292777	0.644591	-134.845215	-46.212643	1.950290	1103.824295	716.817782	312.698167	27.726965	...	0.00000
10	0.749746	7.771606	5.356557	-115.933772	-216.024434	-41.006960	667.264064	1926.752750	1846.163548	120.300839	...	-3271.45744

We can see that we are over-fitting around the 6/7 degree which is in line with the MSE plot.

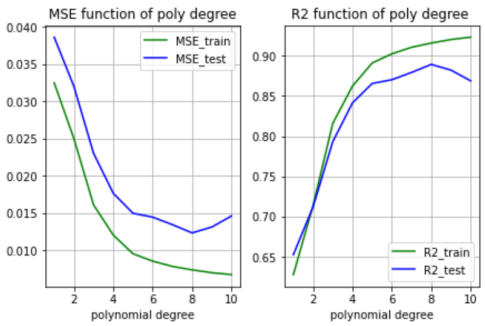
Exercise 2:

In this exercise we will study the bias-variance trade-off by implementing the bootstrap resampling techniques.

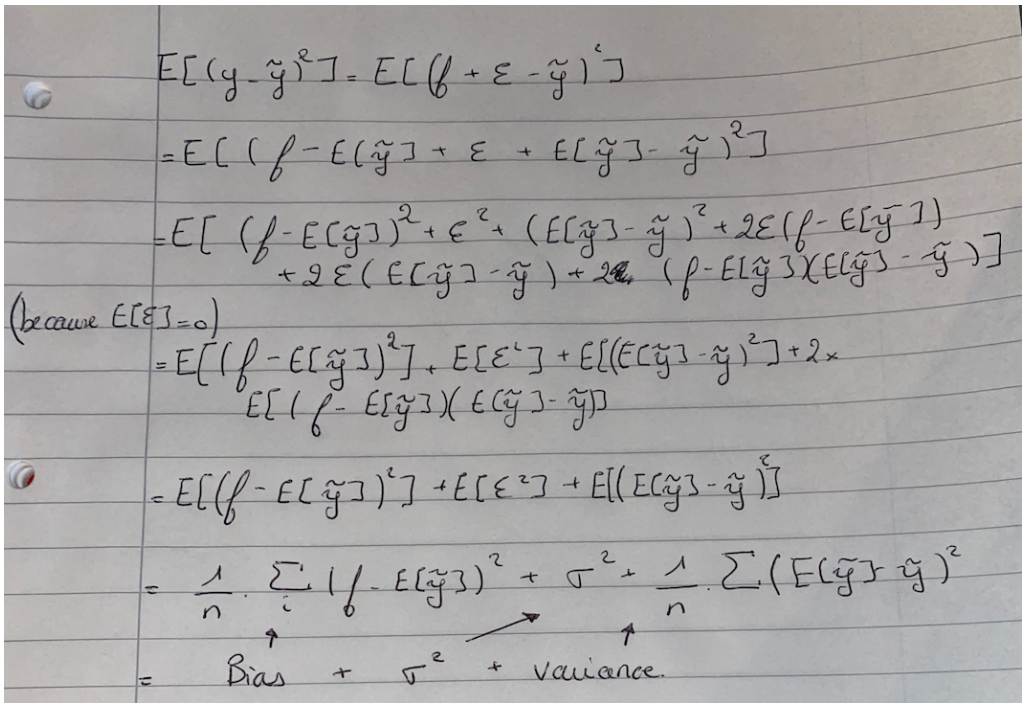
First, there is the figure 2.11 of Hastie:



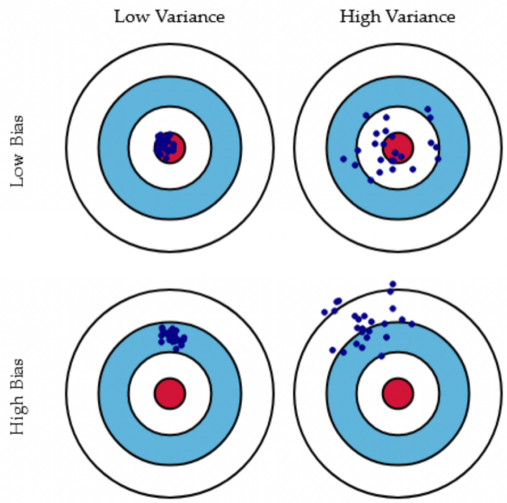
According to the exercise 1, here is the MSE of my different regressions:



Therefore, we can see that a higher polynomial degree refers to a low bias and a high variance and a small polynomial degree refers to a high bias and a low variance.

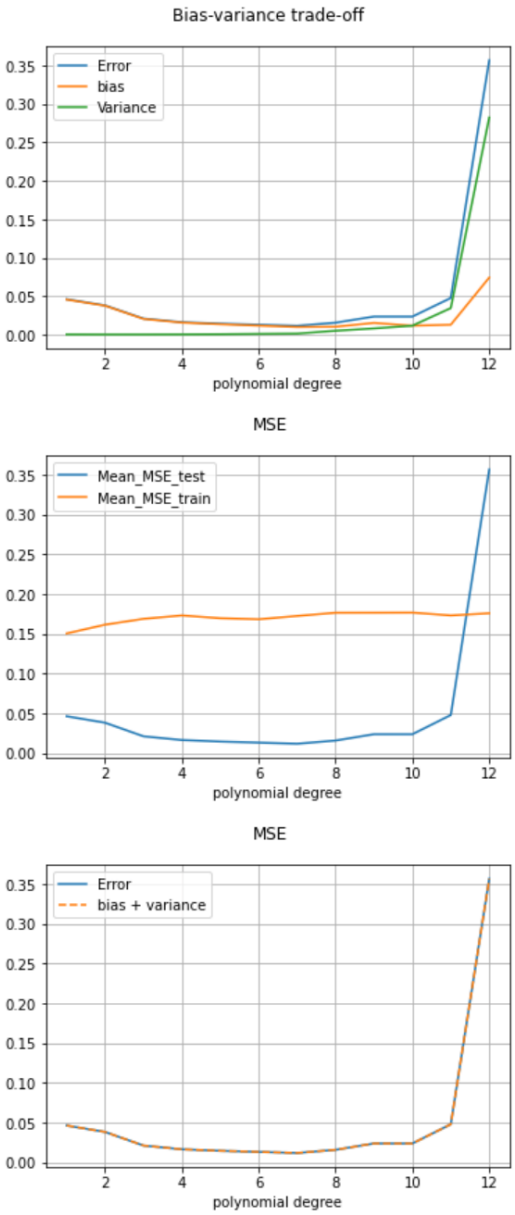


The aim of the bias-variance trade-off is to reduce variance considerably relative to the ordinary least squares (OLS) solution. The lower variance solutions produced provide superior MSE performance.
We can see equations tell us that in order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias.



Source : https://compphysics.github.io/MachineLearning/doc/pub/week37/html/week37.html

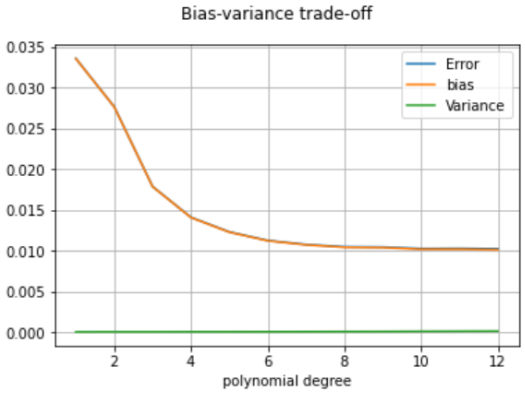
```
################ Parameters ################

n = 20  #number of data
n_bootstraps = 20 #number of boostraps
polynomial = 12 #polynomial degree

############################################
```

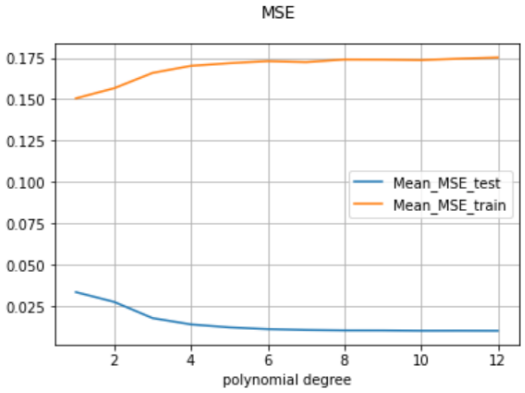


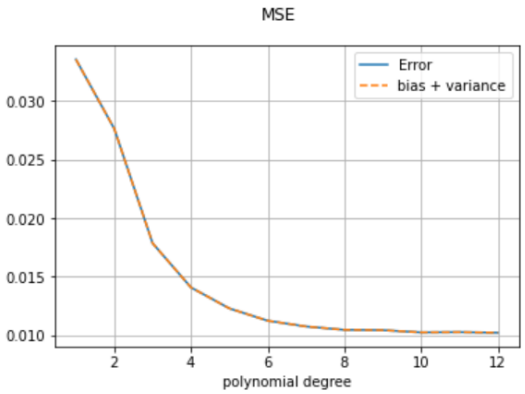
We can see that there is a degree from which the variance increases greatly which implies that the MSE increases. We can see however that the bias decreases. Now we want to minimize the variance rather than the bias. This shows in agreement with exercise 1 that we over-fit from degree 6/7.

However, we can see that depending on the number of data we do not get the same thing. Here we have increased the number of data, and we can see that a polynomial of degree 12 still makes a good regression.

```
################ Parameters ################

n = 100  #number of data
n_bootstraps = 50 #number of boostraps
polynomial = 12 #polynomial degree

############################################
```



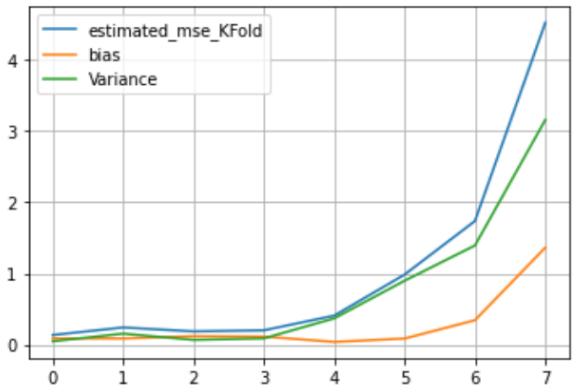




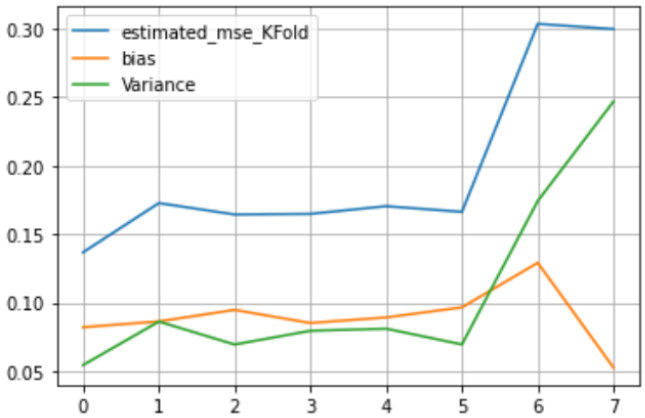
Exercise 3:

In this exercise, we are using the cross-validation with OLS.
We can observe, that for different values of k the graphs are not identical, but in each case, the MSE is higher than with the bootstrap method.

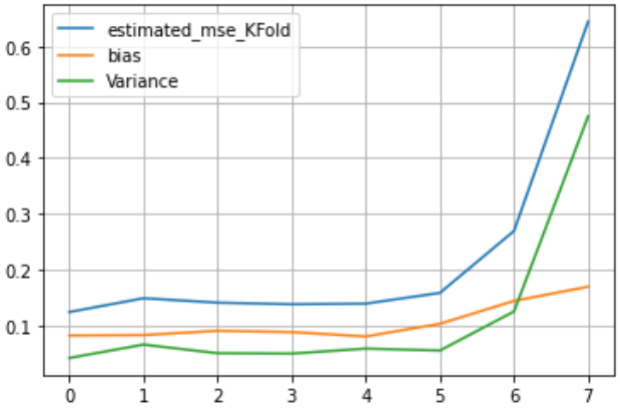
Kfold = 5



Kfold = 8



Kfold = 10



Exercise 4:

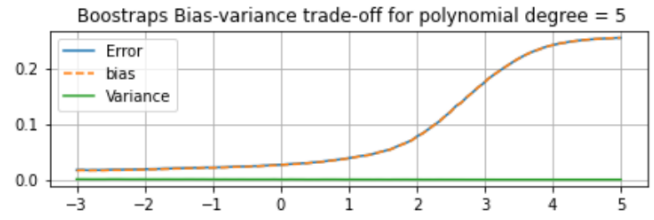
In this exercise we will use the Ridge regression. With this method I will use bootstrap and cross-validation. I code the bootstrap with my own code, and for cross-validation I will use my own code and Sklearn. Ridge regression is based on this equation: (cost function that we want to minimize)

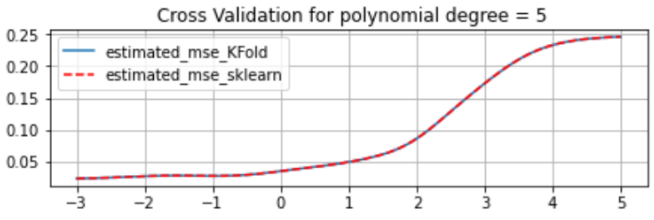
$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

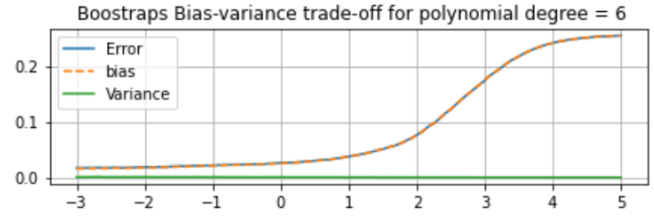
So, for a list of polynomial degree, I plot a graph of the bias-variance trade-off in function of lambda, and a graph of the MSE obtained by the cross-validation in function of lambda also.

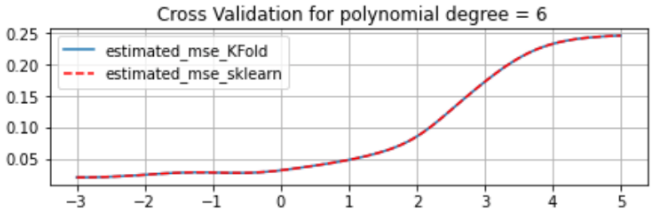
According to previous exercises, the best polynomial degree is around the fifth, so I will run my code for degree from 3 to 7.

```
################ Parameters ################

n = 20  #number of data
polynomial = 10 #polynomial degree
list_degree = [3, 4, 5, 6, 7]
k = 10 #number of splits
n_bootstraps = 20
nlambdas = 100
lambdas = np.logspace(-3, 5, nlambdas)

############################################
```



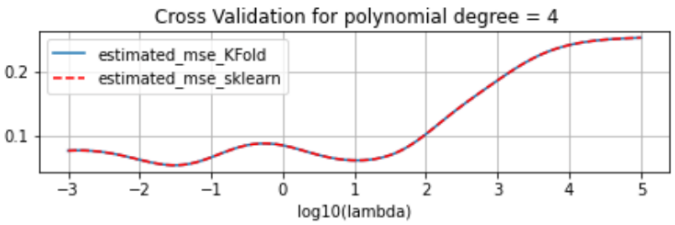


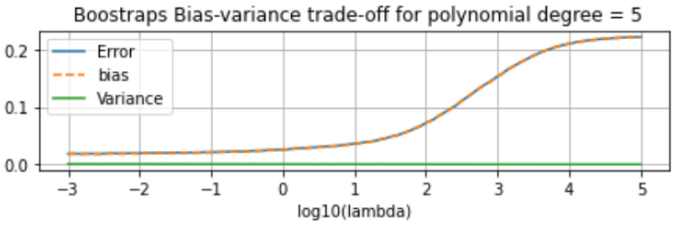


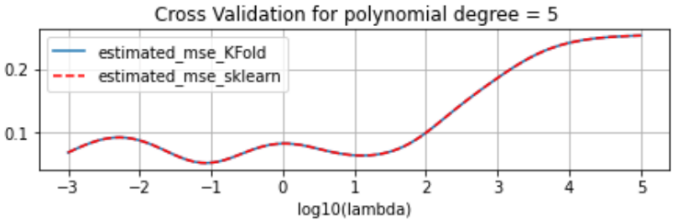


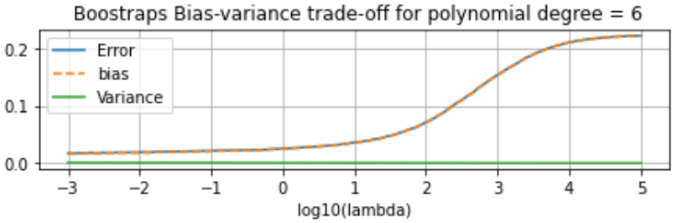
But we can see for a different number of Kfold, 5 for instance, the MSE oscillates and the MSE is no longer minimum for a lambda tending towards 0:

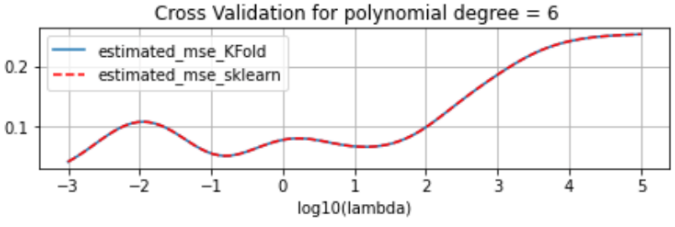
```
################ Parameters ################

n = 20  #number of data
polynomial = 10 #polynomial degree
list_degree = [3, 4, 5, 6, 7]
k = 5 #number of splits
n_bootstraps = 20
nlambdas = 100
lambdas = np.logspace(-3, 5, nlambdas)

############################################
```











Nevertheless, even if the MSE is no longer minimum for a lambda tending towards 0, it is still higher than for kfold = 10 and a lambda tending towards 0. We can observe that we are very near the OLS solution obtained with bootstrap.

Exercise 5:

This exercise is very similar to the previous one, but we will no longer use Ridge regression, but we will use Lasso regression. Lasso regression is based on this equation: (cost function that we want to minimize)

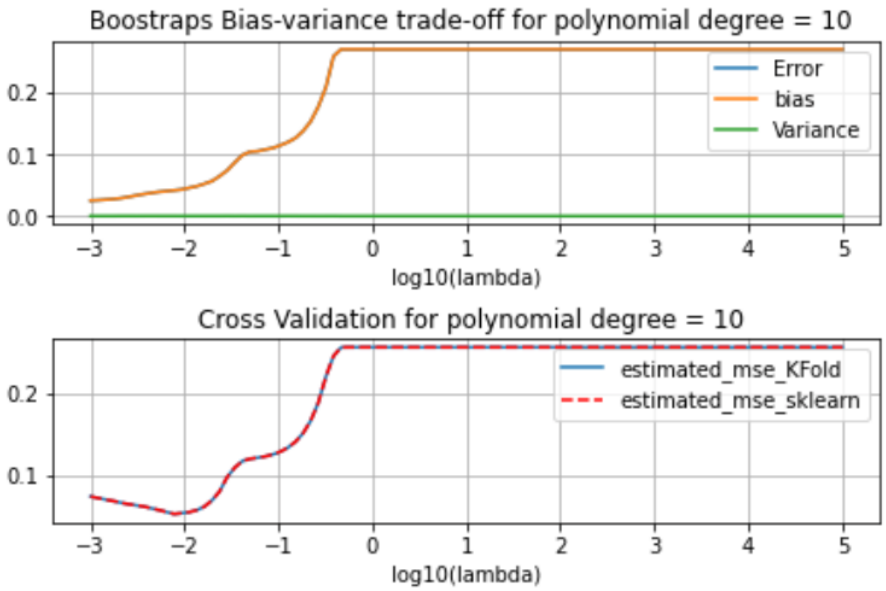
$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||y - X\beta||_2^2 + \lambda ||\beta||_1$$

However, unlike Ridge regression, I use the methods contained in Sklearn.
So, in the first part I define my data and I initialize my parameters, which are here the different degrees of polynomial, the list of lambdas, the number of bootstraps, and of course my amount of data.

Then, for each degree of polynomial I compute for different lambda the error obtained by bootstrap and that obtained by cross-validation.

Here is what we get for a polynomial degree of 5. We can see that the two curves have similar shapes. However, there is a small difference, the MSE obtained by bootstrap only increases when we increase lambdas, while the one obtained by cross-validation has a minimum obtained for a lambda different from 0.

In view of the values, we also observe that the bootstrap gives us a lower MSE than that obtained by cross-validation. Another particularity that we can see from the curves is the threshold just before lambda = 1 from which the MSE remains constant.



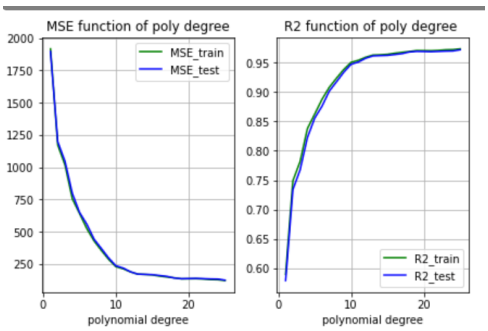
Exercise 6:

We will now use all the techniques used in the previous exercises with real data. I will use these different methods on the data in the SRTM_data_Norway_2 tif file. This data represents the topology in a Norwegian region.

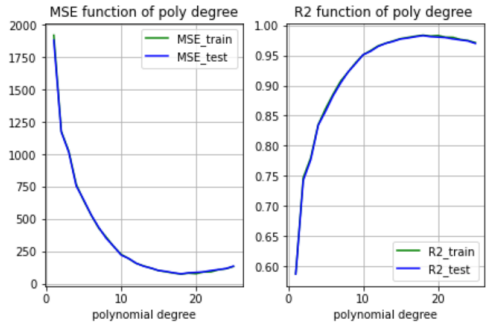
So, first I start to set up the data. We take the longitude and latitude values (x and y) and we get the altitude value (z). We observe that x and y are between 0 and 1.

In a first part I will perform the OLS regression on our dataset. Then, still with the OLS regression I will perform the bootstrap and cross-validation method. After that I will do the same thing with Ridge regression and Lasso regression.

OLS regression:



Without scaling



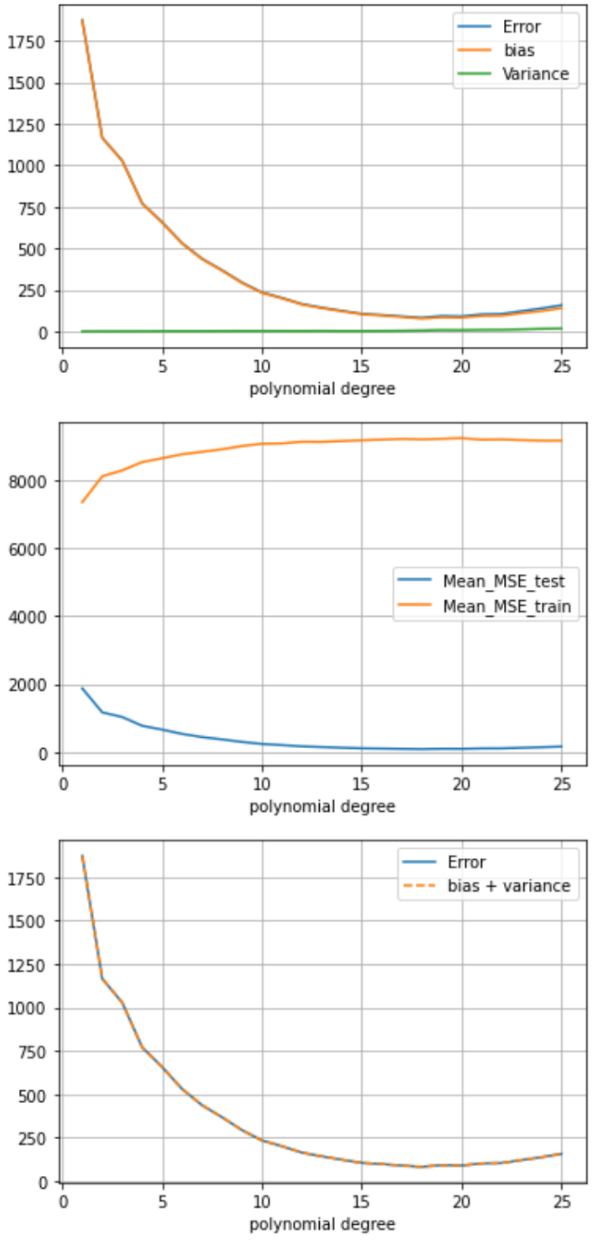
With scaling

We can see that when we normalize our data, we get a smaller error than when we do not scale our data. Moreover, when we scale our data, we can see that we reach a minimum of the MSE for a degree equal to 18 and that without scaling the MSE only decreases until the degree 25 with always a value higher than our regression with scaling.

We can also see that we do not have the same figure as the figure 2.11 of Hastie.In fact, the values of the errors are very high and does not allow us to see the figure. But when we look directly at the values, we can see that the MSE test is higher than the MSE train.

OLS bootstrap:

Here is what we get when we use the bootstrap method:



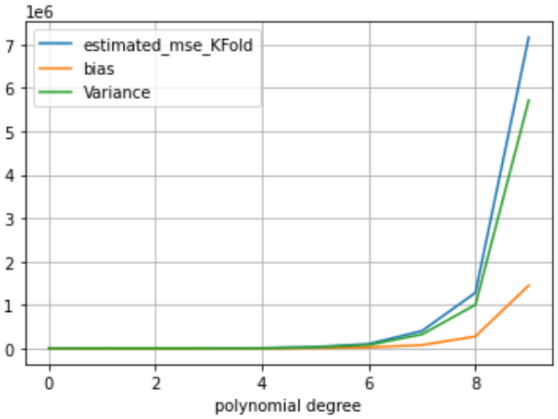
We see that it is different of what we get with the regression of Franke Function. In fact, even if we applied noise to the Franke Function, there was still much less gradient than in the data we are currently processing. Our data represent mountains, there is a lot of elevation change and since there are fjords, the elevation difference can be very high for a small move. So, it is more difficult to find a polynomial degree that satisfies a good regression of our data.

We can observe that the optimal polynomial degree is 18 and this is in agreement with what we had found previously.

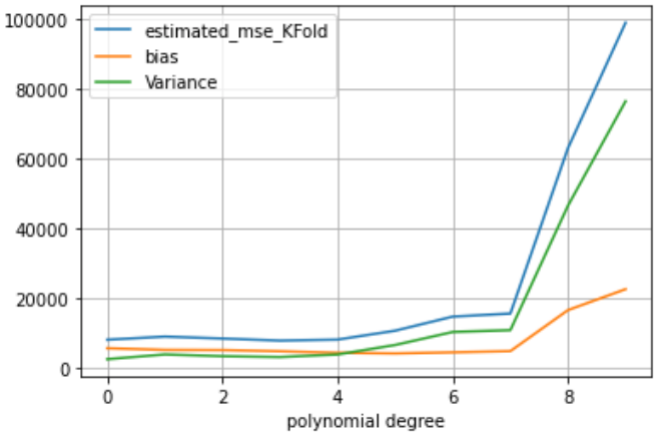
OLS Cross-validation:

Let's calculate the MSE for different values of kfold for different values of lambda (according to what I presented in exercise3, cost function) and all this for different polynomial degree.

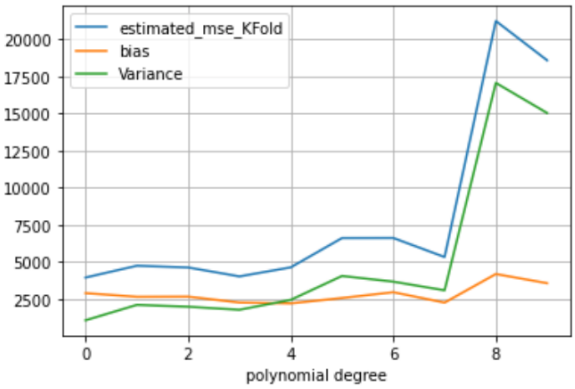
For kfold = 5



Kfold = 8

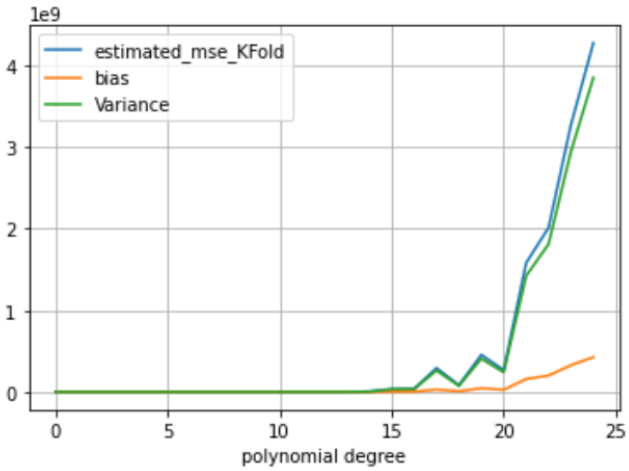


Kfold = 10

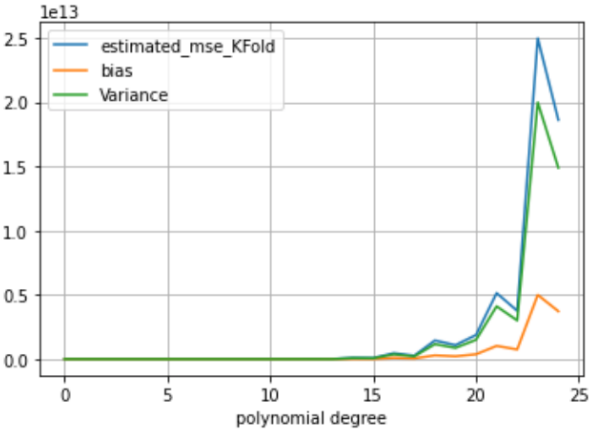


We observe that the MSE is very quickly high, that is why I just stop until polynomial degree 10. However, even if we want to have the smallest MSE, we can see that for the polynomial degree 18 the variance decreases and so the MSE too:

For kfold = 10



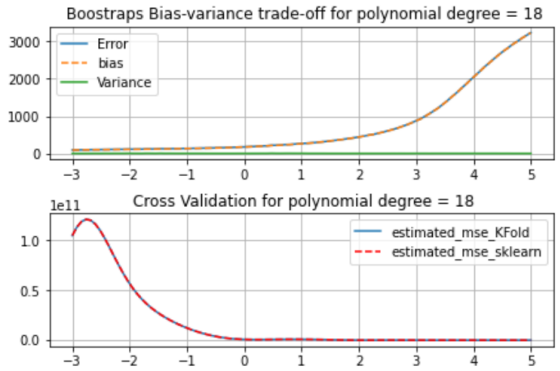
For kfold = 5



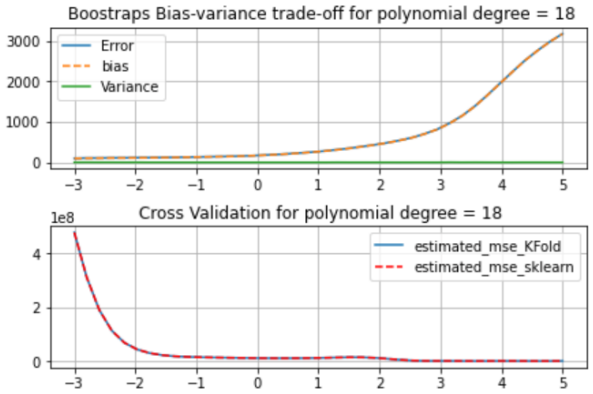
Ridge regression – bootstrap – cross-validation:

The higher the degree of the regression the longer the computation time for these methods will be. So even though the code can perform these methods on a degree list, I will simply run these methods on a polynomial of degree 18 because we have previously decided that this is the optimal degree for our data (and parameters, number of data...).

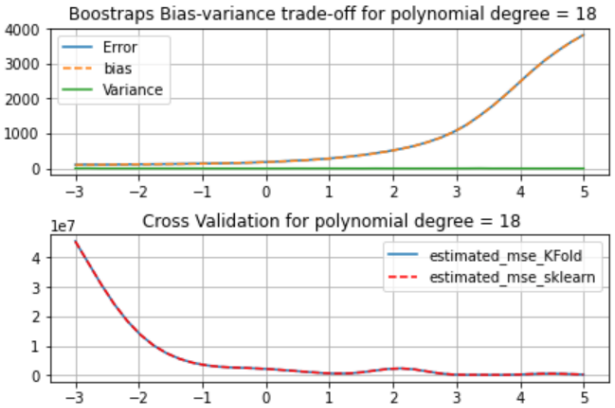
For kfold = 5



For kfold = 8



For kfold = 10



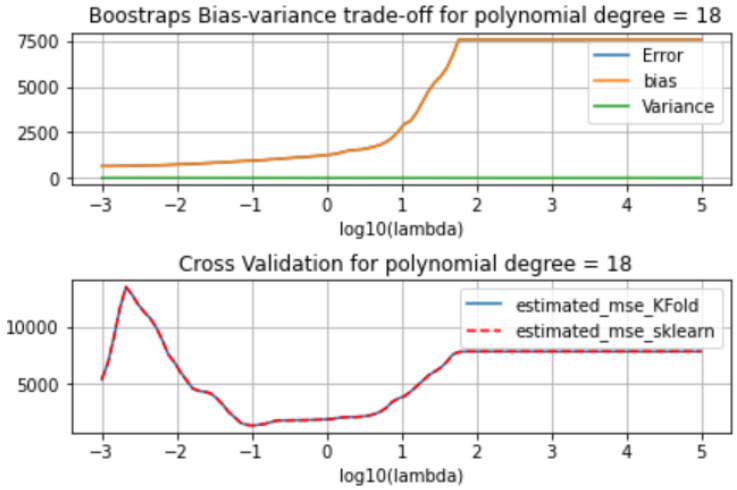
We can observe that contrary to the boostraps method, the MSE from Cross-Validation is decreasing when lambda is increasing. When we increase the number of kfold it seems to have a minimum for lambda near 10 but the error still be very high.

Lasso regression – bootstrap – cross-validation:

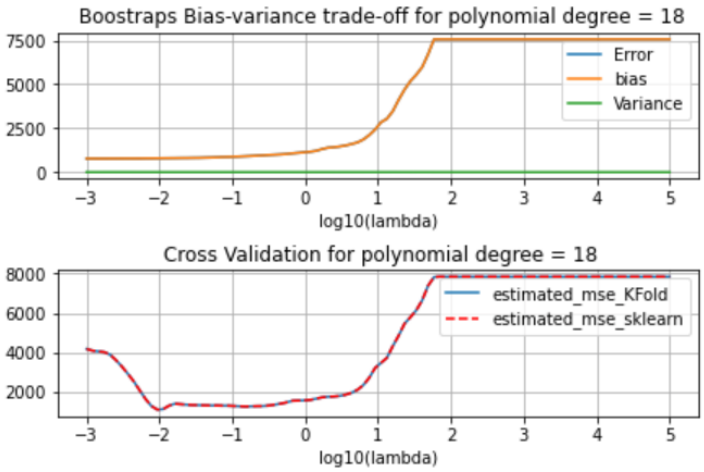
As for the Ridge regression, I will perform these methods with Lasso regression only on a polynomial degree 18.

```
################ Parameters ################

N = 100  #number of data
list_degree = [18]
k = 5 #number of splits
n_bootstraps = 10
nlambdas = 100
lambdas = np.logspace(-3, 5, nlambdas)

############################################
```

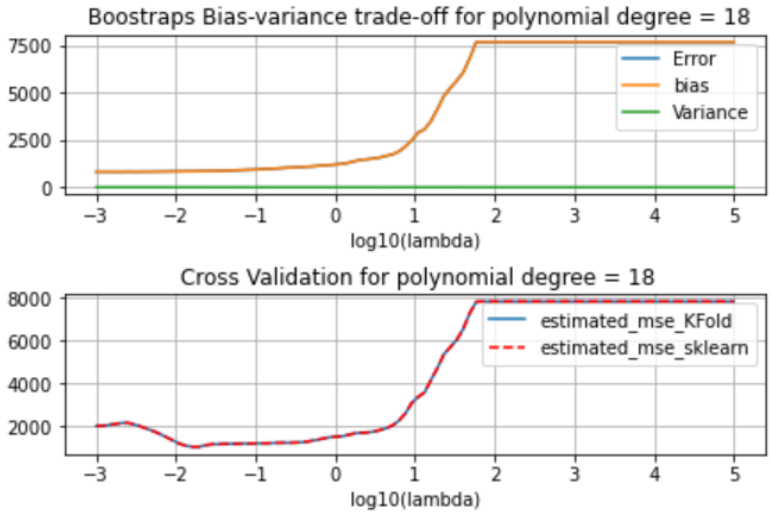
For kfold = 5



For kfold = 8



For kfold = 10



We can observe that contrary to Ridge regression, here the MSE from Cross-Validation. Here the optimal parameter lambda is not when lambda is the smallest as for bootstrap. The MSE is smaller for lambda around 10 to the power -2 (When we are looking at kfold = 8 or 10, even If for kfold = 5 lambda is near 10 to the power -1 but the MSE is still bigger).

Conclusion:

To conclude, in this project we have applied OLS, Ridge and Lasso regression with and without Bootstrap and Cross-Validation.
We have observed that for the Frank Function, the optimal polynomial degree is near 6. We have also noticed that for Ridge and Lasso regression, when we are using the Cross-Validation the optimal parameter lambda is not the smallest but there are oscillations of the MSE in function of lambda.
We can notice the same thing where we are using the real data set.

My feelings about the project:

- I learn a lot of methods during this project, I have never followed a Machine Learning course before.
- The subject was interesting, not too short and not too long
- A little bit hard to do it on my own (Even if I asked for a group, we decided to don't work together so I will look for other teammates for the next project).