Project 1 FYS-STK4155 Autumn 2017

Jon Audun Baar & Mikael Ravndal

7. oktober 2018

1 Introduction

A common problem in inferential statistics is to investigate the relationship between two stocastic variables, in particular how one of them depends on the other. A widely used method to develop a model for this relationship is technique called linear regression. In this project we aim to evaluate the performance of three different types of linear regression (OLS-, ridge- and lassoregression) using a polynomial as our model.

First we evaluate our methods on a known function, that is we know the relationship between our variables. Then we apply the same methods to evaluate the performance of our methods on a real data set where we don't know the underlying relationship (if there is any at all).

Most of our code is in the GitHub repository. We have chosen to not use the code in our report. We have, however, made a python file, project01.py, that provides most of our code in sequence, so that it is possible to follow the code that produce our testresults and plots in order.

2 Method

The general idea of linear regression is to assume that the relationship between your independent and dependent variables is given by some function, and then try to approximate this function, also called a model, by minimizing some error term. In our particular case we have two independent variables, also called predictors, and one dependent variable, also called response. We also assume that this relationship is given by a polynomial.

something about variance vs bias

2.1 Ordinary least squares regression

First we looked at fitting our model using what is called ordinary least squares regression. In this case the error that we wish to minimize, also called the cost function, is defined by $C(\boldsymbol{\beta}) = \sum_{i=1}^{N} (\boldsymbol{y_i} - \boldsymbol{y'_i})^2$. One great benefit of this method as opposed to for instance the later explained lasso method, is that C is a convex twice differentiable function and thus by calculus we can solve the minimization problem analytically. One can show that the solution is given by (1). For a thorough explanation of this identity consult [**?**]

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \boldsymbol{y} \tag{1}$$

2.2 Ridge regression

One major weakness of OLS regression is that noisy sample data can cause large coefficients, that is a model with large variance. One way to approach this problem is to impose a restriction to the magnitude of our coefficients $\boldsymbol{\beta}$. Ridge regression does this by adding a term, also called a penality, to the cost function of OLS regression an in that way penalizing large betas. The Ridge cost function is then given by

$$C(\boldsymbol{\beta}) = \sum_{i=1}^{N} (\boldsymbol{y_i} - \boldsymbol{y_i'})^2 + \sum_{i=1}^{N} \beta_i^2 \tag{2}$$

As with the case of OLS, we are in a good position with respect to solving this minimization problem. By Lagranges method of multiplication and the same techniques as for OLS one can show that the solution for β is given by

$$\boldsymbol{\beta} \;=\; (X^T X + \lambda I)^{-1} X^T \boldsymbol{y} \tag{3}$$

where λ is a real number which for high values emphazises restricting the magnitude of the coefficients and for low values emphazises minimizing the error of the model compared to the training data.

2.3 Lasso regression

Just as the ridge method, lasso solves the problem of high variance by adding a penality to the cost function. The difference is that lasso adds the 1-norm of the betas to C instead of the 2-norm. Thus we get

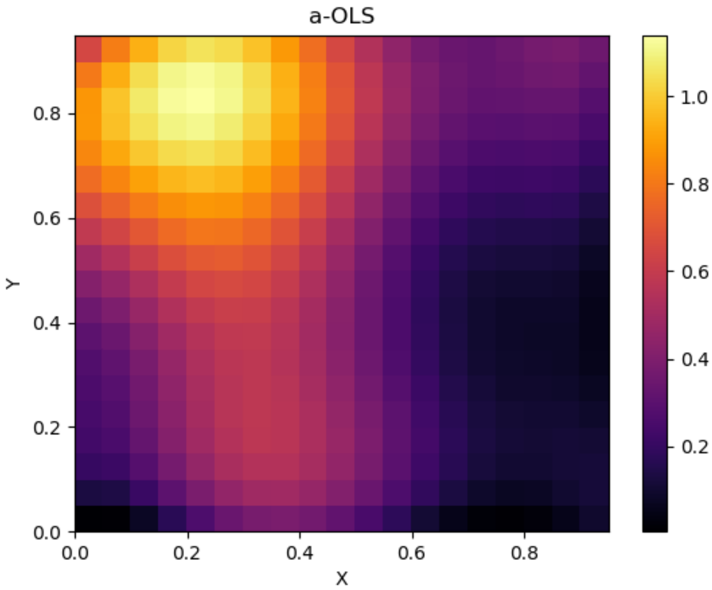
$$C(\boldsymbol{\beta}) = \sum_{i=1}^{N} (\boldsymbol{y_i} - \boldsymbol{y_i'})^2 + \sum_{i=1}^{N} |\beta_i| \tag{4}$$

The advantage of this approach is that it tends to set coefficinets that corresponds to "corellatingpredictors to zero. For a thoroug derivation of this consult [?]

3 Implementation

3.1 Ordinary least square on the Franke function

First we have generated some test data, with the noise being very little, which we have plotted to get a more intuitive feel for how it looks. We have done this for all the models.
Ordinary Least square of the Frankefunction(the same as Ridge with $\lambda = 0$):



As you can tell from our code, we haven't made an own function for ordinary least square, since it is the same as Ridge, just with the λ set to 0. This will also show later that when the λ gets low it is very similar to OLS.
Here is the values for calculating the five first degrees and their MSE and R2-score:
OLS Test Data We can tell that our model is fitting our test data better and

k	MSE	R2
1	0.020836568820240136	0.7159598591230791
2	0.015631136506458913	0.7869193218104034
3	0.007175355204257789	0.9021869233537347
4	0.003960217232650187	0.9460150723293508
5	0.0018602479352178107	0.9746414541595732

better with a higher degree. Now we have to check with bootstrap as well to see if our model fits the data good.
Before the resampling we have also calculated the betas of $k = 5$:
Var of Beta, degree 5

$$\begin{matrix}[9.47238055e-03 & 8.01217226e-01 & 4.12183842e-01 & 9.89887798e+00 \\ 8.39455751e+00 & 4.14211294e+00 & 2.71771303e+01 & 3.28888083e+01 \\ 1.78285914e+01 & 1.31093669e+01 & 1.91368860e+01 & 2.67004422e+01 \\ 1.91391193e+01 & 9.34737951e+00 & 1.13215581e+01 & 2.55450522e+00 \\ 4.40599813e+00 & 3.60735967e+00 & 1.80107762e+00 & 1.00979408e+00 \\ & & & 1.50021893e+00]\end{matrix}$$

Also the 95-percentage confidence interval of the betas: 95-percentage CI of betas, degree 5

$$
\begin{matrix}
[[7.25873511e-02 & 4.54098870e-01] \\
[7.23335269e+00 & 1.07421092e+01] \\
[3.44378188e+00 & 5.96043621e+00] \\
[-4.41211552e+01 & -3.17880887e+01] \\
[-2.42828028e+01 & -1.29254533e+01] \\
[-1.57022805e+01 & -7.72437196e+00] \\
[4.22571363e+01 & 6.26923830e+01] \\
[4.18269196e+01 & 6.43072224e+01] \\
[1.67482166e+01 & 3.32996879e+01] \\
[-1.02120404e+01 & 3.98078784e+00] \\
[-3.30453851e+01 & -1.58973753e+01] \\
[-7.33667324e+01 & -5.31114961e+01] \\
[-1.94596236e+01 & -2.31061330e+00] \\
[-3.98768180e+01 & -2.78922323e+01] \\
[1.90552365e+01 & 3.22448231e+01] \\
[-2.43921072e+00 & 3.82593943e+00] \\
[1.88142647e+01 & 2.70423776e+01] \\
[8.68290825e+00 & 1.61280472e+01] \\
[-7.84195698e+00 & -2.58124771e+00] \\
[1.66275865e+01 & 2.05666637e+01] \\
[-1.80168037e+01 & -1.32155417e+01]]
\end{matrix}
$$

The code and commenting for the calculations is to be found in python-file project01.py

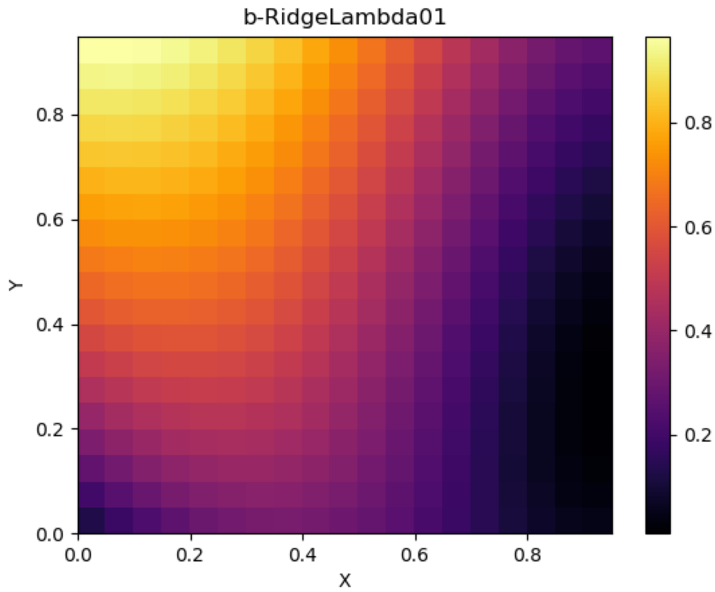
3.1.1 Resampling

Using our bootstrapping algorithm with a resampling of 100, degree of five, we get these values:
VAR: 0.000052
BIAS: 0.001933
Bootstrap mean of MSE: 0.0020
Bootstrap mean of r2Score: 0.9757

The bootsrap values aligns pretty well with our original ones.

3.2 Ridge regression

Ridge Regression with $\lambda = 0.1$
Graphic plot of how it looks:



Ridge Test Data

k	MSE	R2
1	0.025965982389446095	0.6906471643146342
2	0.018247163430545398	0.7826074259086047
3	0.010258352759015437	0.8777843076427536
4	0.009382588378732645	0.8882179662029949
5	0.009143926633340667	0.8910613282063765

Compared to OLS, we can tell that Ridge does significantly worse then OLS.
Var of Beta, degree 5

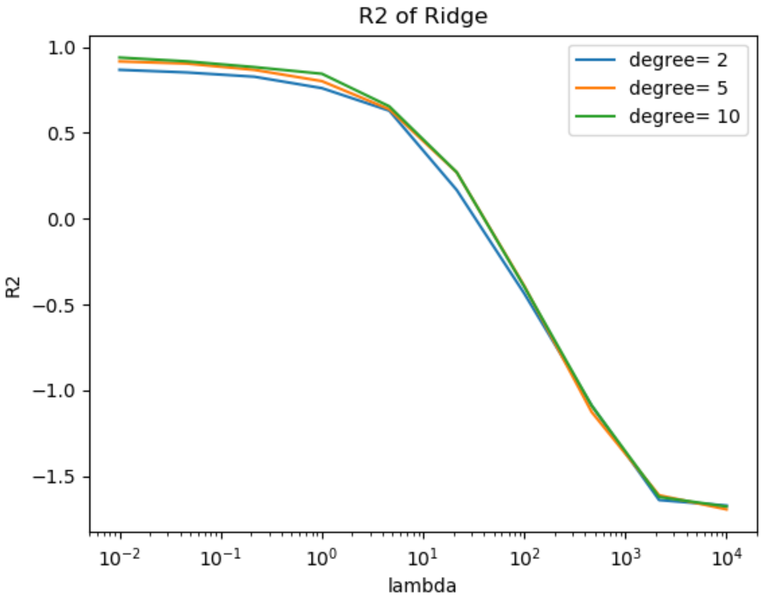
[0.00077004 0.01729839 0.01384995 0.06816008 0.06826285 0.05696662
0.0417508 0.02034709 0.06671475 0.03500783 0.02294549 0.01073849
0.03470761 0.01261125 0.02127092 0.03349824 0.01062464 0.03548007
0.03422875 0.04204926 0.02520149]

Also the 95-percentage confidence interval of the betas: 95-percentage CI of betas, degree 5

$$
\begin{aligned}
&[[8.39518095e-01 \quad 9.45603016e-01]\\
&[6.00189901e-01 \quad 1.10406786e+00]\\
&[9.22637923e-01 \quad 1.52633094e+00]\\
&[-6.51421032e+00 \quad -5.30387095e+00]\\
&[5.84279933e-01 \quad 1.67511842e+00]\\
&[-5.82384066e+00 \quad -4.85326698e+00]\\
&[3.42317950e+00 \quad 4.16050815e+00]\\
&[1.31502300e+00 \quad 2.10193829e+00]\\
&[-2.00335018e+00 \quad -1.07460992e+00]\\
&[1.38884378e+00 \quad 1.92131292e+00]\\
&[3.21253289e+00 \quad 3.98989176e+00]\\
&[5.79118852e-01 \quad 1.17557901e+00]\\
&[-3.41043372e-02 \quad 7.77207310e-01]\\
&[-8.02285903e-01 \quad -2.23570539e-01]\\
&[3.31692208e+00 \quad 3.80174824e+00]\\
&[-3.69457810e+00 \quad -2.99905374e+00]\\
&[-2.10226995e+00 \quad -1.54360569e+00]\\
&[-5.64130918e-03 \quad 8.94344017e-01]\\
&[-1.07674065e+00 \quad -1.87377295e-01]\\
&[3.14047856e-01 \quad 8.79022520e-01]\\
&[-1.87908328e+00 \quad -1.35609332e+00]]
\end{aligned}
$$

3.2.1 Resampling

We can take a look at how different lambdas and different degrees of the polynomial makes a change in the R2-score and the MSE.
Here is a plot to show how they develop as a function of lambda.



We can tell pretty easily that the degree of the predictions doesn't matter much compared to how much the choice of lambda do. We can still tell that a lower degree function does worse then the other functions.

Some interesting values from bootstrap:

Bootstrap-values from degree of 5, lmb = 0.1 and 100 bootstrap-samples
VAR: 0.000067
BIAS: 0.008640
Bootstrap mean of MSE: 0.0087
Bootstrap mean of r2Score: 0.8980

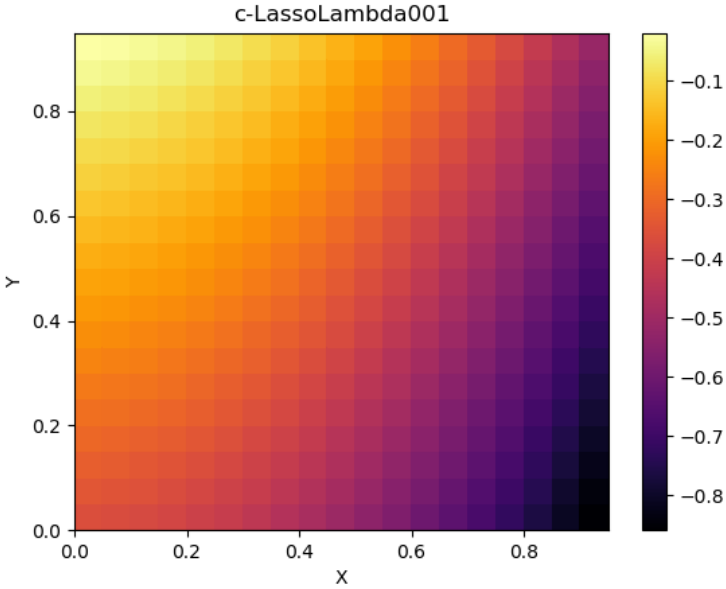
Bootstrap-values from degree of 5, lmb = 1 and 100 bootstrap-samples
VAR: 0.000059
BIAS: 0.011915
Bootstrap mean of MSE: 0.0120
Bootstrap mean of r2Score: 0.8597

Bootstrap-values from degree of 5, lmb = 10 and 100 bootstrap-samples
VAR: 0.000066
BIAS: 0.020274
Bootstrap mean of MSE: 0.0203
Bootstrap mean of r2Score: 0.7617

Bootstrap-values from degree of 2, lmb = 10 and 100 bootstrap-samples
VAR: 0.000057
BIAS: 0.022754
Bootstrap mean of MSE: 0.0228
Bootstrap mean of r2Score: 0.7327

3.3 Part c)

Lasso Regression with $\lambda = 0.01$



Ridge Test Data

k	MSE	R2
1	0.25272407945476655	-2.01089746779934852
2	0.25272407945476655	-2.0108974677993485
3	0.25272407945476655	-2.0108974677993485
4	0.25272407945476655	-2.0108974677993485
5	0.25272407945476655	-2.0108974677993485

Var of Beta

$$[0. \quad 0.00025829 \quad 0.00394575 \quad 0. \quad 0. \quad 0.00324571$$
$$0. \quad 0. \quad 0. \quad 0. \quad 0. \quad 0. \quad 0.$$
$$0. \quad 0. \quad 0. \quad 0. \quad 0. \quad 0. \quad 0.$$
$$0.]$$

95-percentage CI of betas

$$
\begin{array}{r}
[[0. \quad 0.] \\
[-0.40937802 \quad -0.34637908] \\
[-0.17825992 \quad 0.06797132] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0.58986796 \quad -0.36654512] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.] \\
[-0. \quad 0.]]
\end{array}
$$

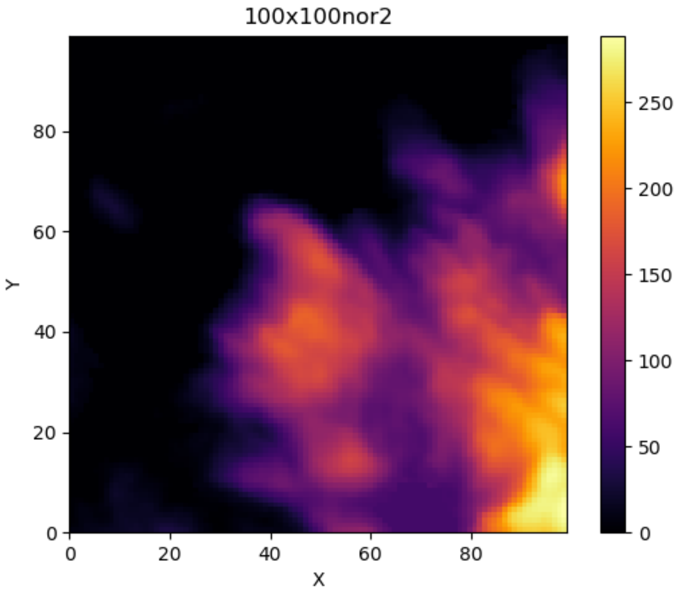
There is obviusly something happening with Lasso that doesn't work. Our Lasso method works great with the real data.

3.3.1 Resampling

Bootstrap-values from degree of 1, lmb = 0.1 and 100 bootstrap-samples
VAR: 0.000000
BIAS: 0.239704
Bootstrap mean of MSE: 0.2397
Bootstrap mean of r2Score: -2.0278
This was the same for all degrees and values of lambda, so I only included this.

3.4 Part d)

Imports 100x100 chunk of real data from top left corner of dataset nr.1.
Plot of real data

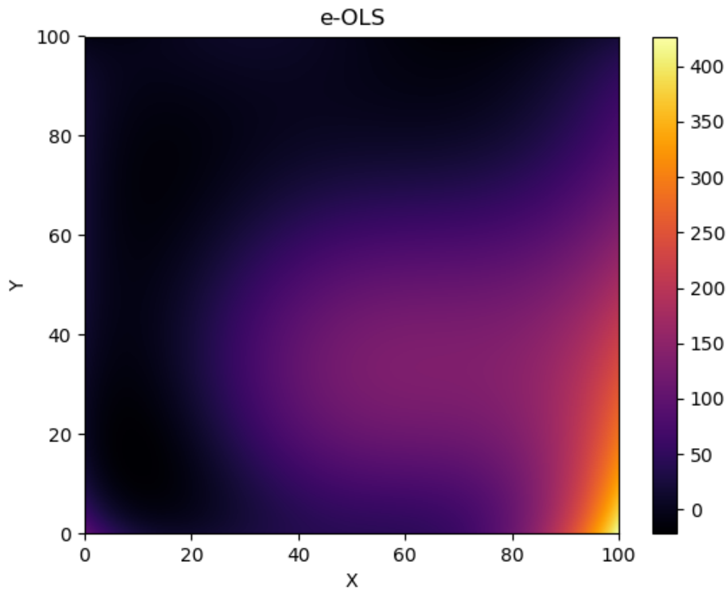


3.5 Real Data

Repeat of the previous method and data, but with real data.

3.6 OLS

Plot of the data with the OLS-method of degree 5:



OLS Score of the Real Data:

k	MSE	R2
1	1912.190996	0.587579
2	1177.397306	0.746059
3	1023.042624	0.779351
4	824.587589	0.822153
5	791.268452	0.829340

Var of Beta

$$
\begin{matrix}
[2.32350868e-17 & 3.59021583e-14 & 9.26320612e-14 & 6.67623760e-12 \\
2.76179532e-12 & 2.44594128e-11 & 1.00899287e-09 & 2.82477031e-08 \\
3.26102575e-08 & 4.03014136e-09 & 7.93068781e-13 & 8.29815731e-12 \\
2.72417498e-12 & 8.99036913e-12 & 2.47909085e-12 & 4.07929803e-17 \\
2.67723887e-16 & 5.18767547e-16 & 4.23509770e-16 & 1.84571052e-16 \\
 & & & 9.42227510e-17]
\end{matrix}
$$

3.6.1 Resampling

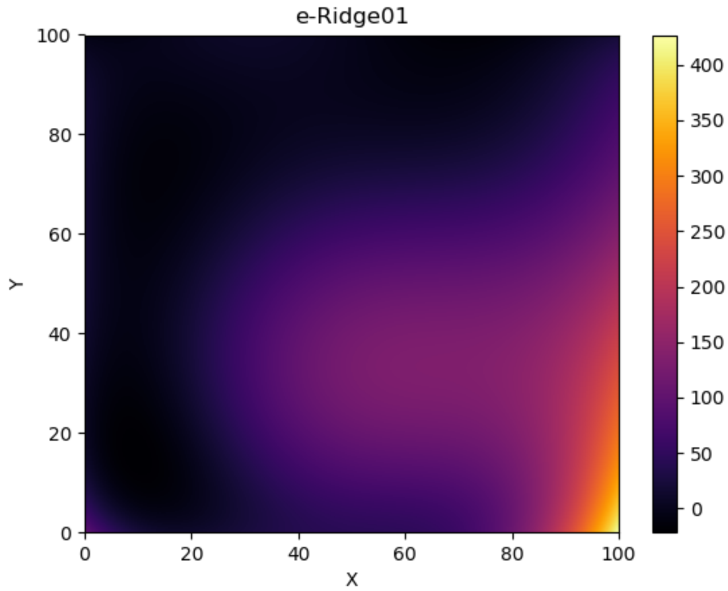
Bootstrap-values from degree of 5 and 100 bootstrap-samples
VAR: 1.287057

BIAS: 791.273865
Bootstrap mean of MSE: 792.5609
Bootstrap mean of r2Score: 0.8291

The MSE and R2-score are really close to our values for MSE and R2.

3.7 Ridge

Plot of the data with the Ridge-method of degree 5, lambda = 0.1:



Ridge Score of the Real Data with lambda = 0.1:

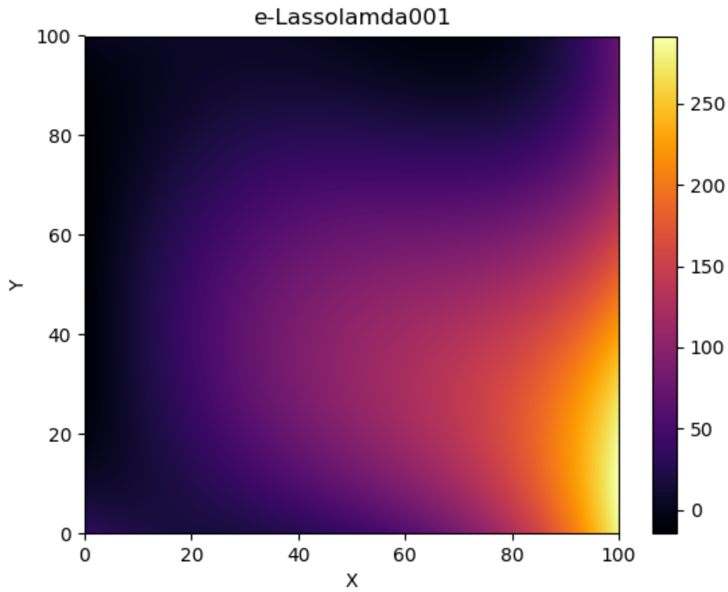
k	MSE	R2
1	1912.190999	0.587579
2	1177.397307	0.746059
3	1023.042624	0.779351
4	824.635824	0.822143
5	791.268452	0.829340

3.7.1 Resampling

Bootstrap-values from degree of 5, lmb = 10 and 100 bootstrap-samples
VAR: 1.297002
BIAS: 791.285081
Bootstrap mean of MSE: 792.5821
Bootstrap mean of r2Score: 0.8291
Can tell that MSE and R2-score is pretty similar to our computations.

3.8 Lasso

Plot of the data with the Lasso-method of degree 5, lambda = 0.01:



Lasso Score of the Real Data with lambda= 0.1:

k	MSE	R2
1	6601.647721	-0.423841
2	1564.545187	0.662560
3	1189.660192	0.743415
4	1155.338990	0.750817
5	974.391256	0.789844

3.8.1 Resampling

Bootstrap-values from degree of 5, lmb = 10 and 100 bootstrap-samples
VAR: 1.689048
BIAS: 1239.400793
Bootstrap mean of MSE: 1241.0898
Bootstrap mean of r2Score: 0.7323

Here the MSE and R2score is a pretty long way from our scores when we use all our data. So the model is probably a little overfitted and we should expect our R2 to actually be lower than it is.

4 Conclusion