# Assignment 1

Computational Intelligence, SS2017

| Team Members | | |
|---|---|---|
| Last name | First name | Matriculation Number |
| Chen | Yung-Yu | 1646883 |
| Káčerová | Erika | 1647045 |
| | | |

## 1 Linear Regression

### 1.1 Derivation of Regularized Linear Regression

$$J(\boldsymbol{\theta}) = \frac{1}{m} \|\boldsymbol{X\theta} - \boldsymbol{y}\|^2 + \frac{\lambda}{m} \|\boldsymbol{\theta}\|^2$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{2}{m} (\boldsymbol{X\theta} - \boldsymbol{y})^T \cdot \frac{\partial (\boldsymbol{X\theta} - \boldsymbol{y})}{\partial \boldsymbol{\theta}} + \frac{2\lambda}{m} \boldsymbol{\theta}^T$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{2}{m} (\boldsymbol{X\theta} - \boldsymbol{y})^T \cdot \boldsymbol{X} + \frac{2\lambda}{m} \boldsymbol{\theta}^T$$

$$\mathbf{0} = \frac{2}{m} (\boldsymbol{X\theta} - \boldsymbol{y})^T \cdot \boldsymbol{X} + \frac{2\lambda}{m} \boldsymbol{\theta}^T$$

$$\mathbf{0} = (\boldsymbol{X\theta} - \boldsymbol{y})^T \cdot \boldsymbol{X} + \lambda \boldsymbol{\theta}^T$$

$$\mathbf{0} = (\boldsymbol{X\theta} - \boldsymbol{y}) \cdot \boldsymbol{X}^T + \lambda \boldsymbol{\theta}$$

$$-\lambda \boldsymbol{\theta} = \boldsymbol{X}^T \boldsymbol{X\theta} - \boldsymbol{X}^T \boldsymbol{y}$$

$$-\boldsymbol{X}^T \boldsymbol{X\theta} - \lambda \boldsymbol{\theta} = -\boldsymbol{X}^T \boldsymbol{y}$$

$$(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}) \boldsymbol{\theta} = \boldsymbol{X}^T \boldsymbol{y}$$

$$\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

## 1.2 Linear Regression with polynomial features

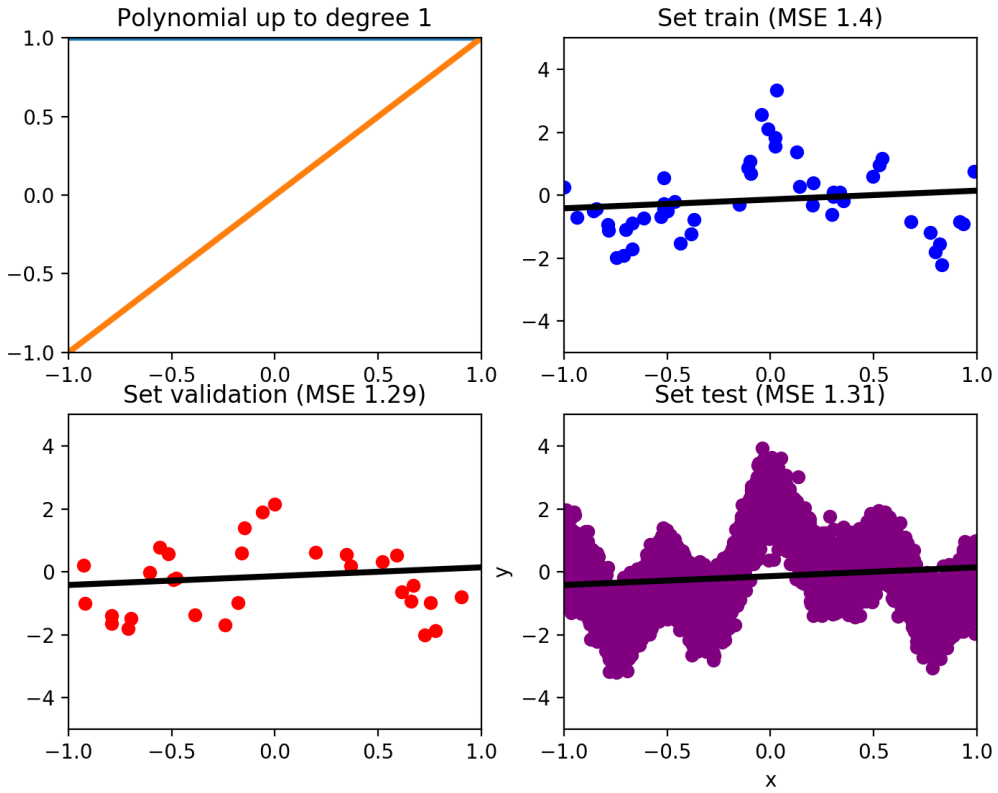


Figure 1.1: *Results of the linear regression with the polynomial degree of 1.*

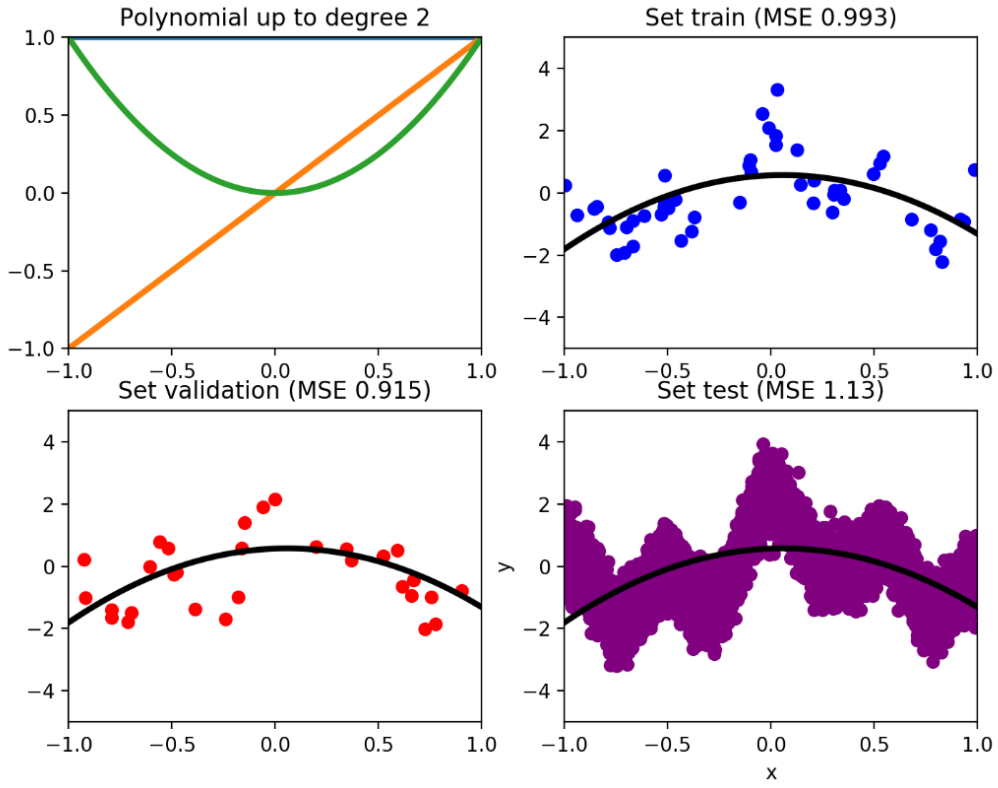


Figure 1.2: *Results of the linear regression with the polynomial degree of 2.*

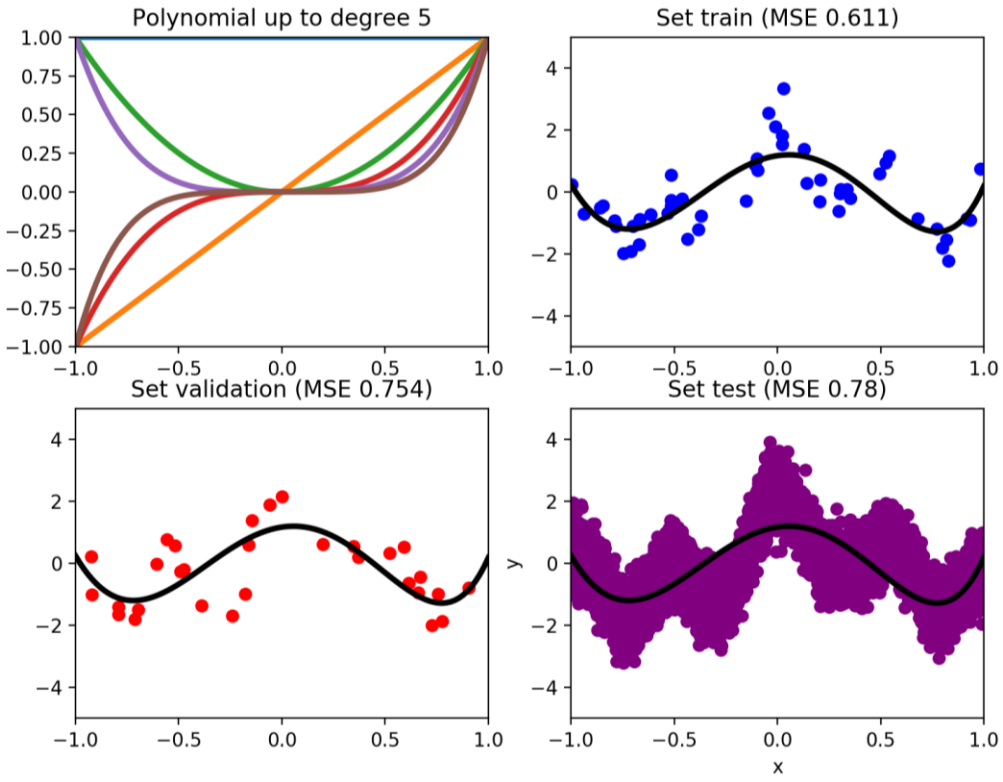


Figure 1.3: *Results of the linear regression with the polynomial degree of 5.*

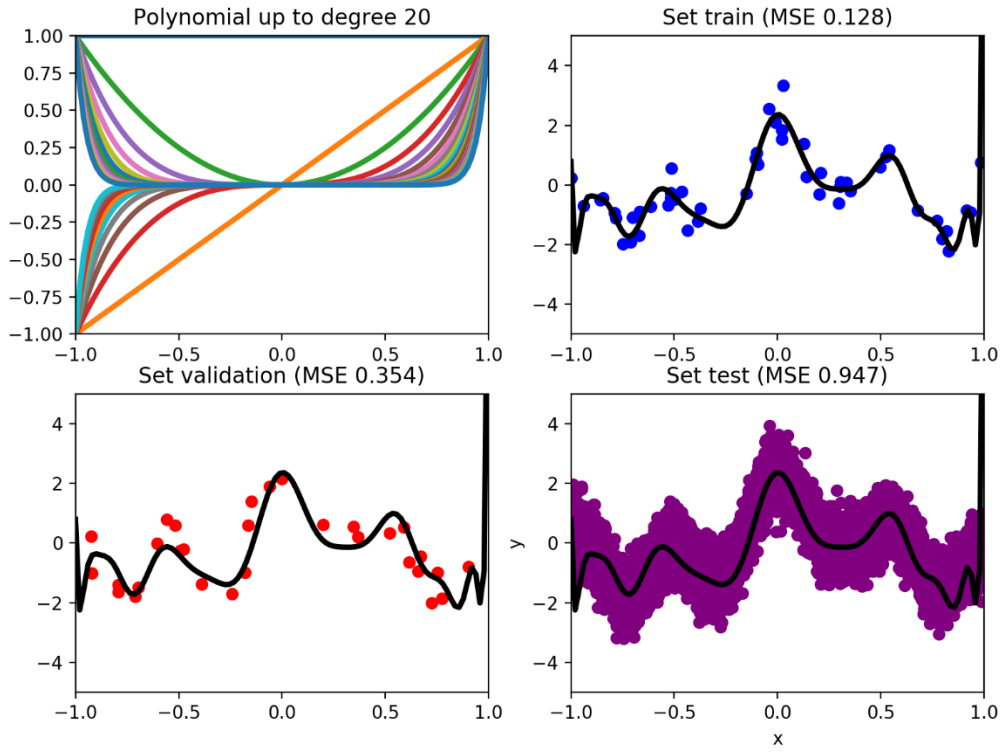


Figure 1.4: *Results of the linear regression with the polynomial degree of 20.*

The lowest training error is given by the polynomial regression with degree of 30 as shown on the following figure. The test error is this case has the value of 366148.05210642.

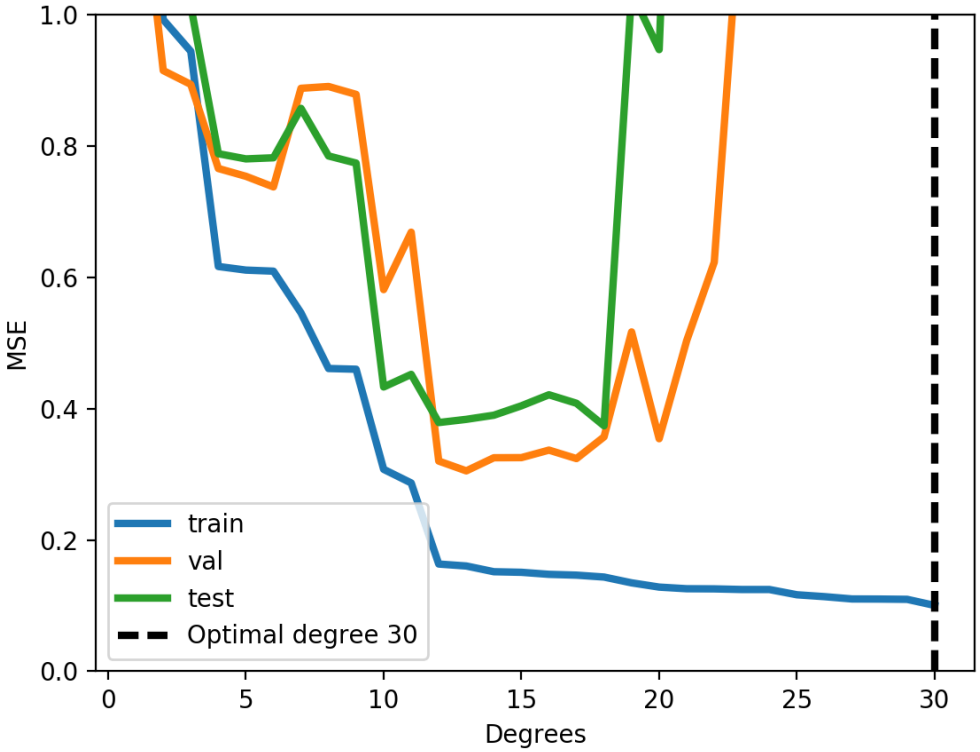


Figure 1.5: *Training, validation and testing errors as a function of the polynomial degree.*

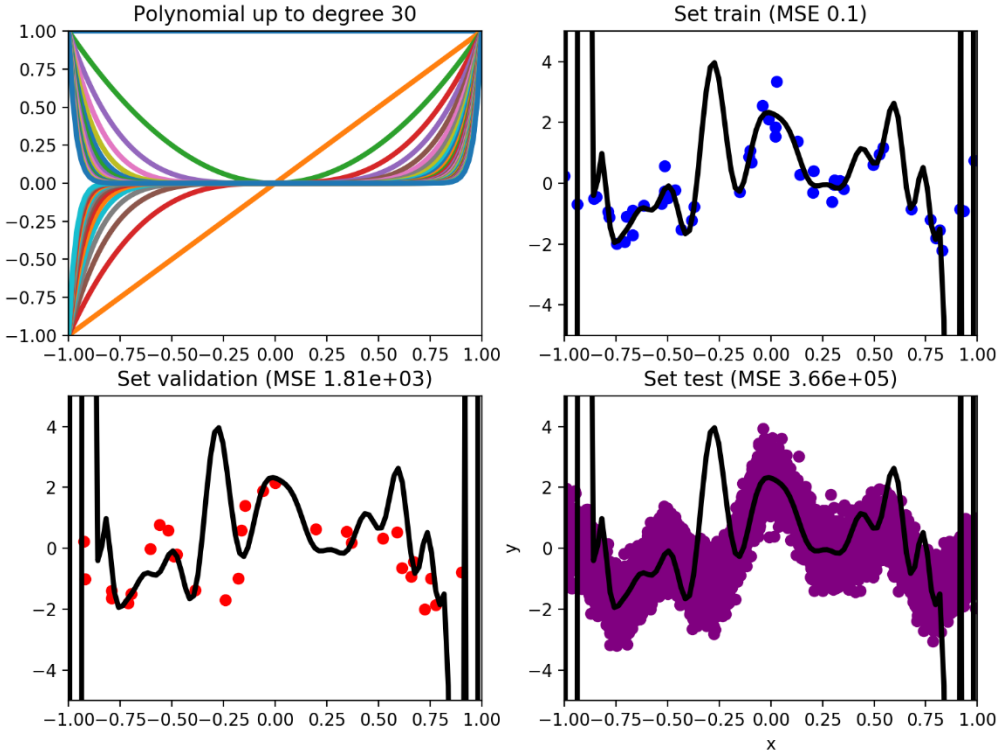


Figure 1.6: *Results of the linear regression with the lowest training error (the polynomial degree 30).*

The lowest validation error is given by the polynomial regression with degree of 13 as shown on the following figure. The test error is this case has the value of 0.3837016.

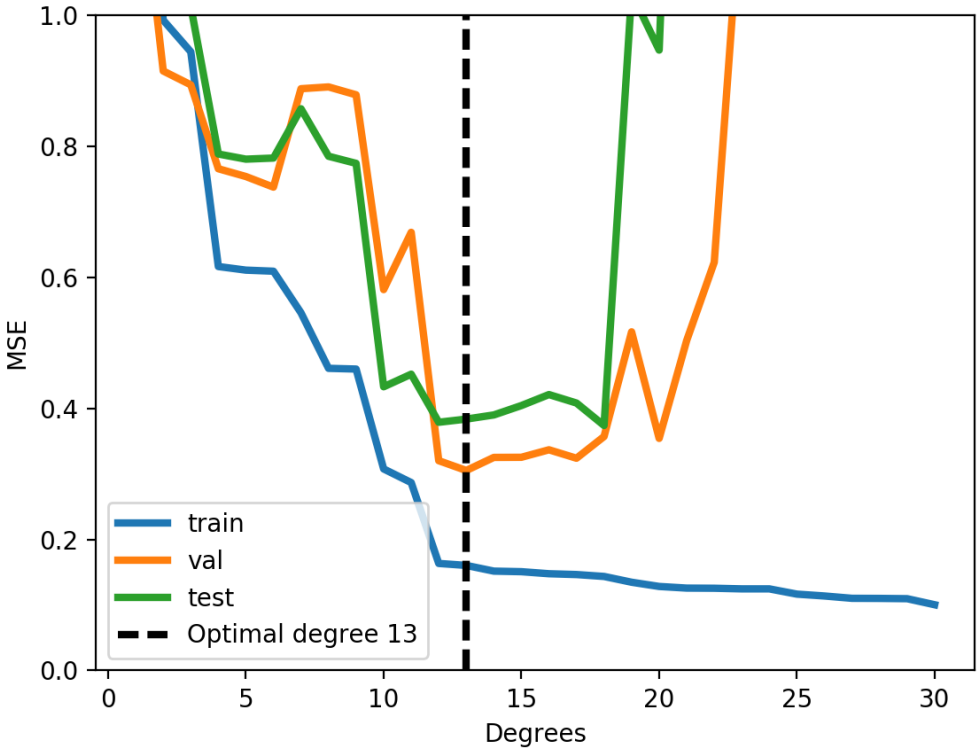


Figure 1.7: *Training, validation and testing errors as a function of the polynomial degree.*

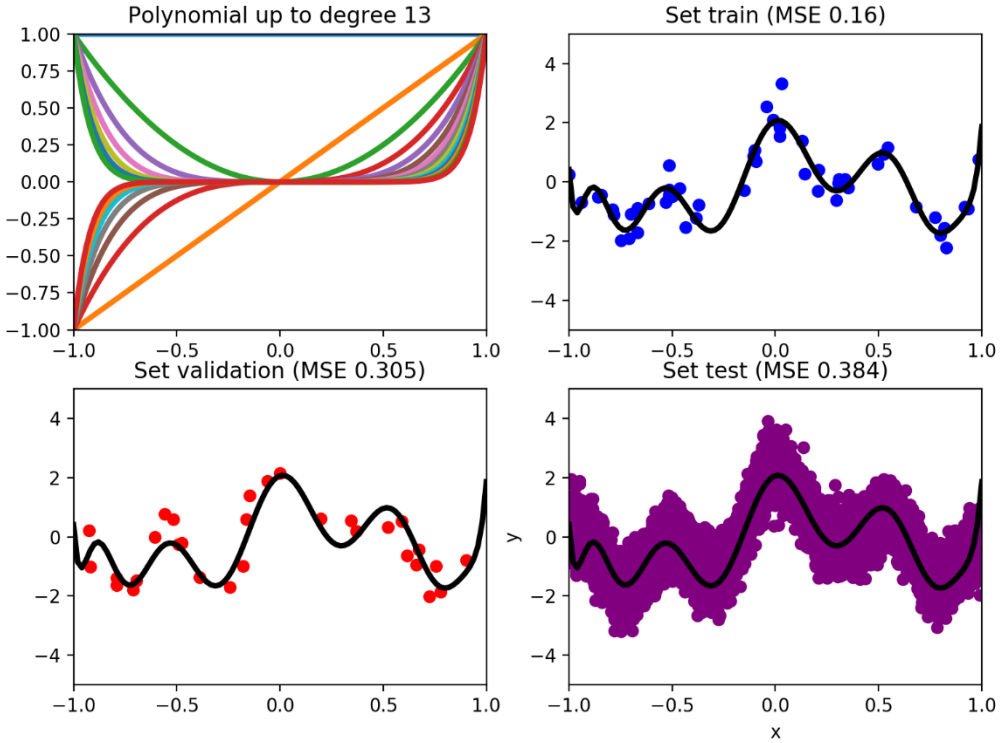


Figure 1.8: *Results of the linear regression with the lowest validation error (the polynomial degree 13).*

On the figure 1.5 one can easily see that the testing error decreases with every ascend of the degree of the polynomial regression. The lowest testing error is reached using the degree of 30, the highest we used. At the first sight, this may seem as a good achievement. But by looking at the extremely high values of validation and testing error at the same point, it is clear that this model is useless. Thus that model achieved a good results on the training set, it has a poor predictive performance. At some point in the process of increasing the degree, the models started to over-fit the training data. It means that instead of learning from the training data, the models started to memorize them, which led to performance failure on the unknow data from both of the other, the validation and the testing set.
Because of that it is very important to use the validation data set. By taking care of the error on validation set with data which are unknown during the learning process, we can prevent the model from over-fitting. Based on the results shown on the figure 1.7, we can determine the best fitting model eliminated the over-fitting on training data.

## 1.3 Linear Regression with radial basis functions

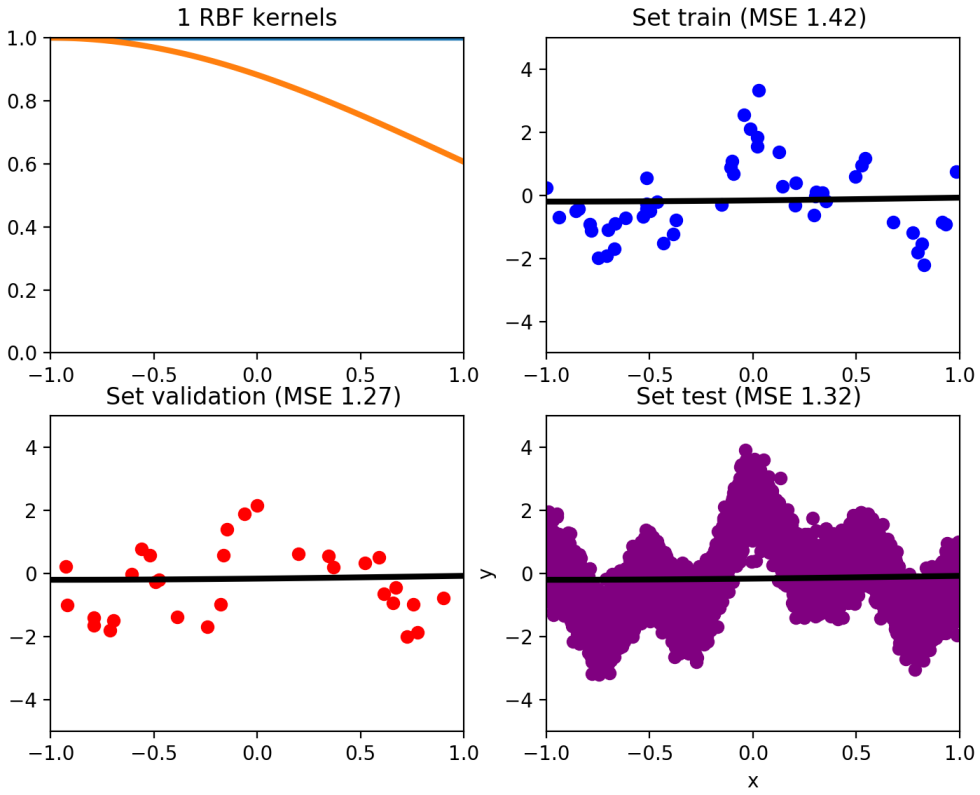


Figure 1.9: *Results of the linear regression with RBF degree of 1.*

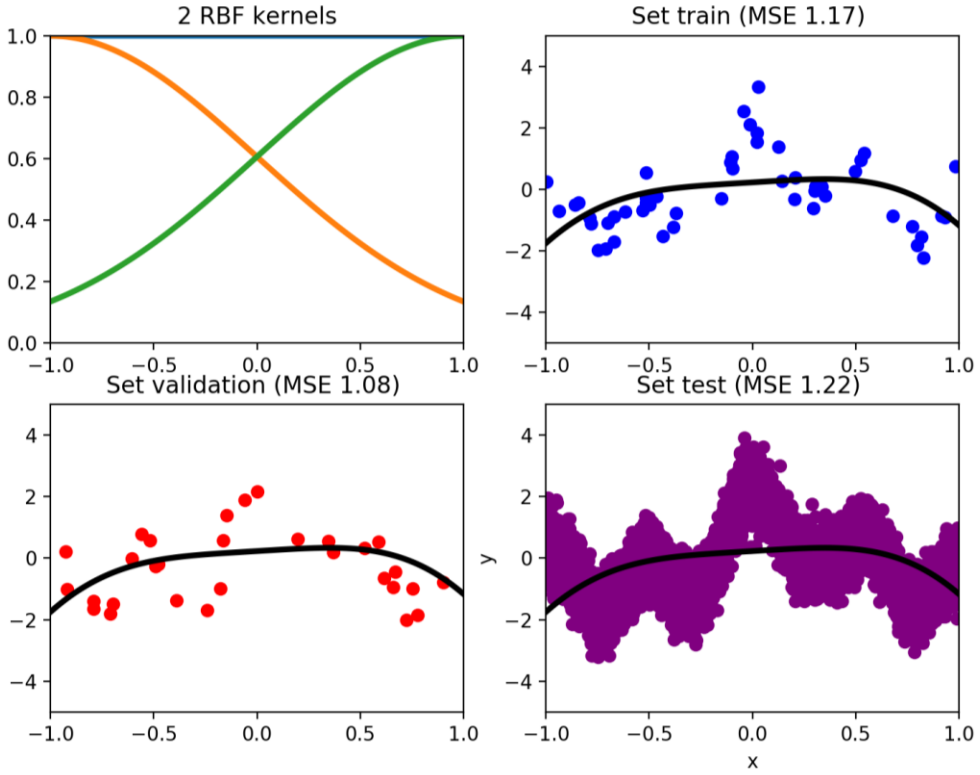


Figure 1.10: *Results of the linear regression with RBF degree of 2.*

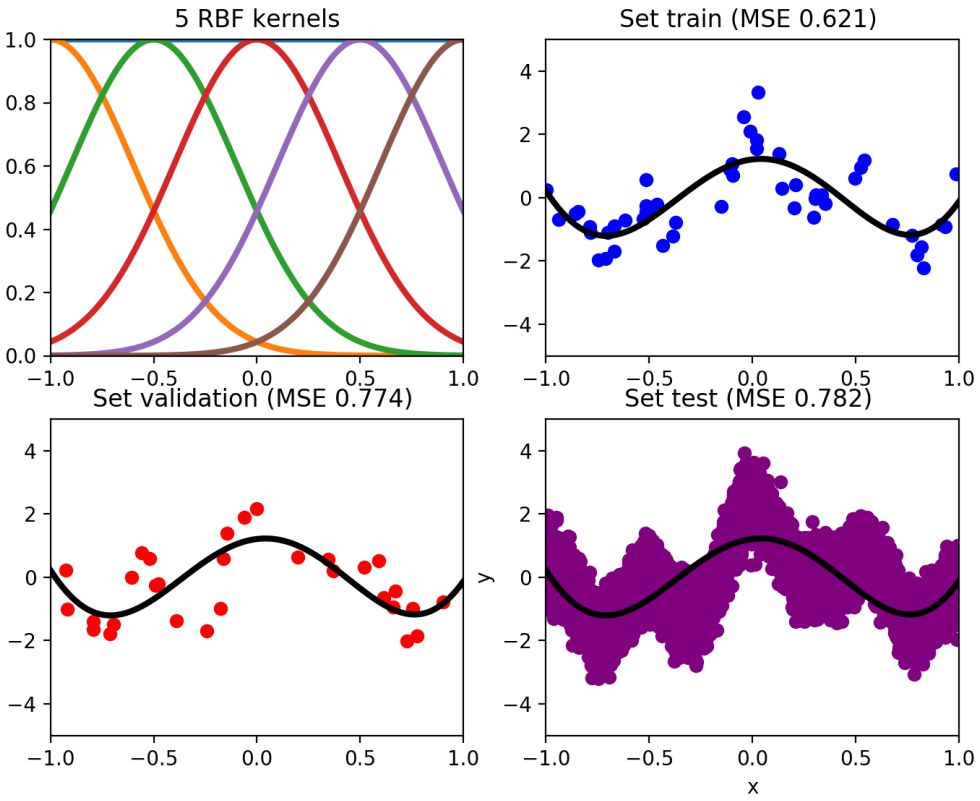


Figure 1.11: *Results of the linear regression with RBF degree of 5.*

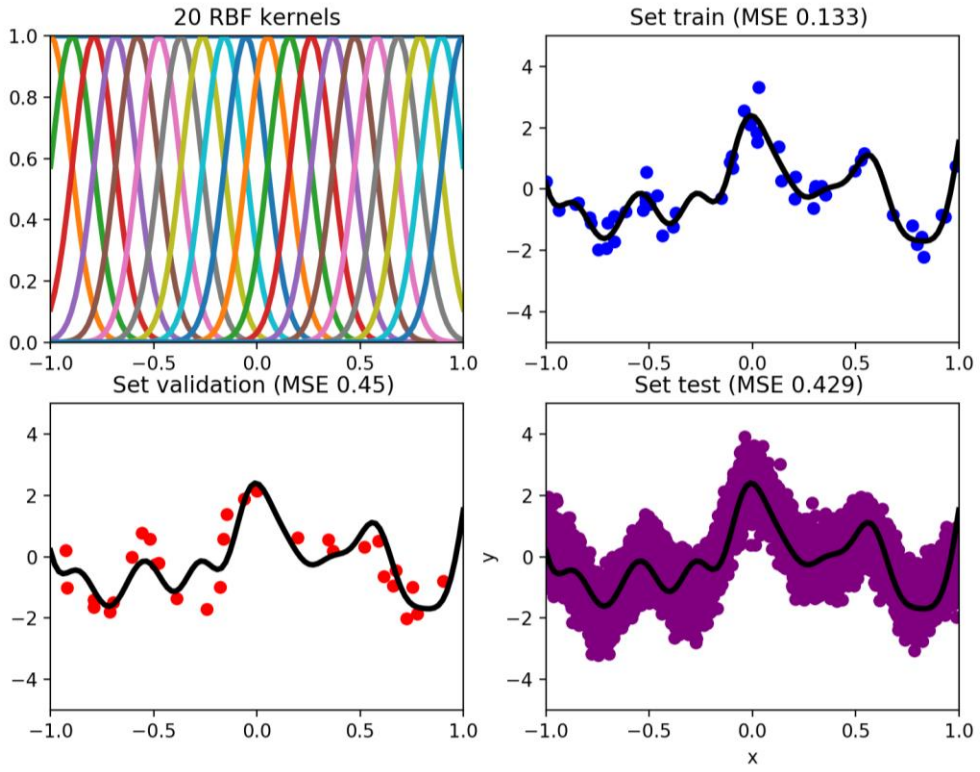


Figure 1.12: *Results of the linear regression with RBF degree of 20.*

The lowest training error is given by the RBF regression with degree of 40 as shown on the following figure. The test error is this case has the value of 181.69182615.

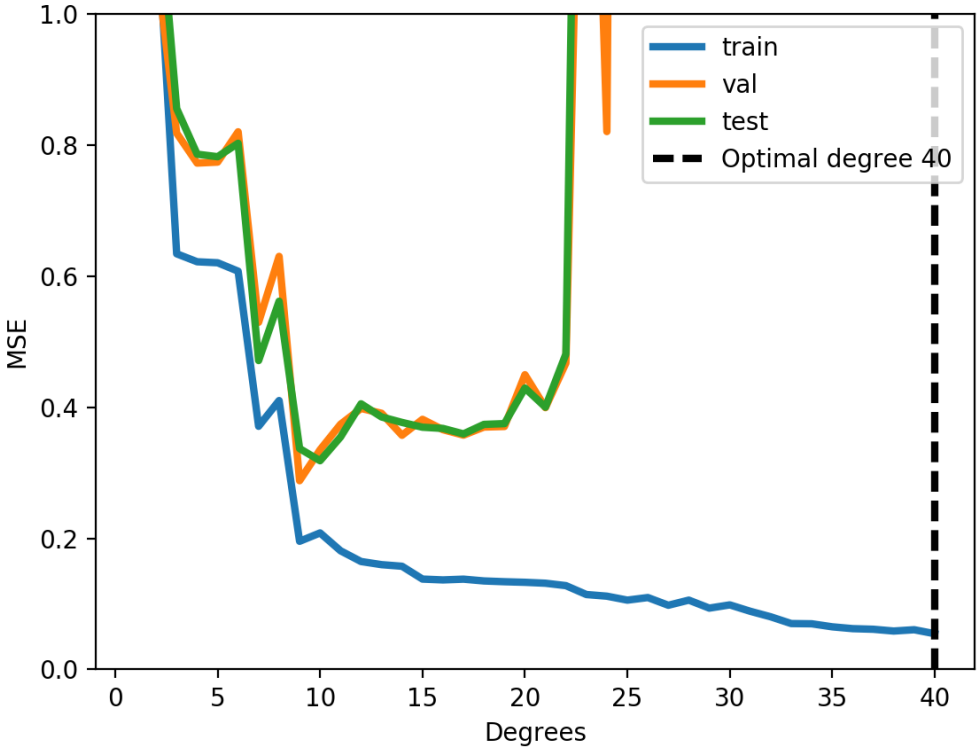


Figure 1.13: *Training, validation and testing errors as a function of the RBF degree.*

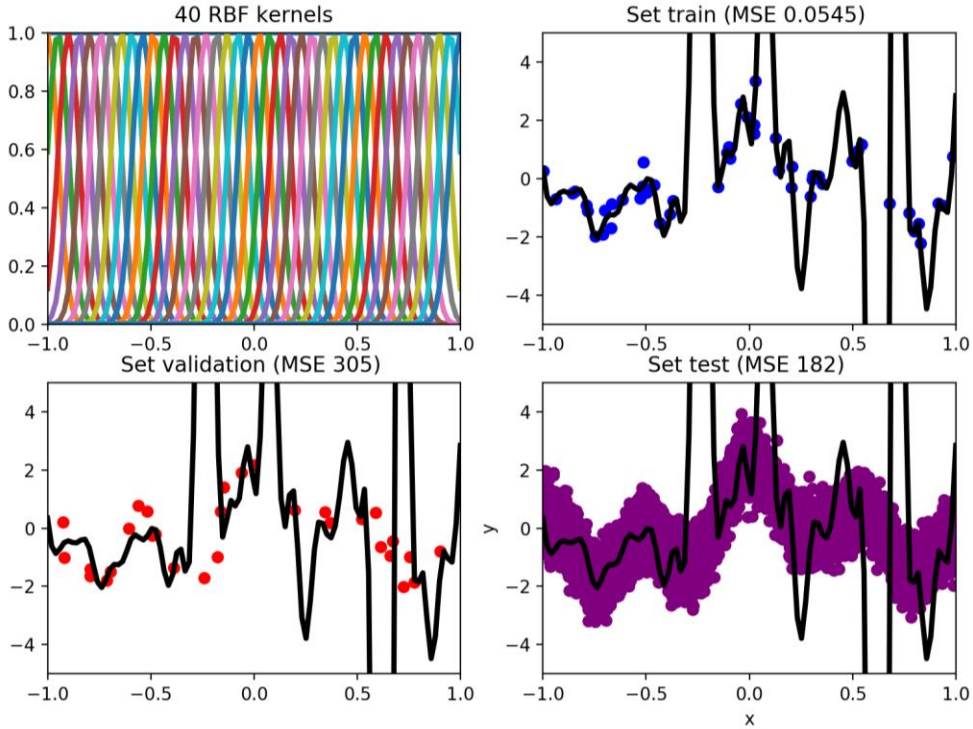


Figure 1.14: *Results of the linear regression with the lowest training error (the RBF degree 40).*

The lowest validation error is given by the *RBF* regression with degree of 9 as shown on the following figure. The test error is this case has the value of 0.33701301.

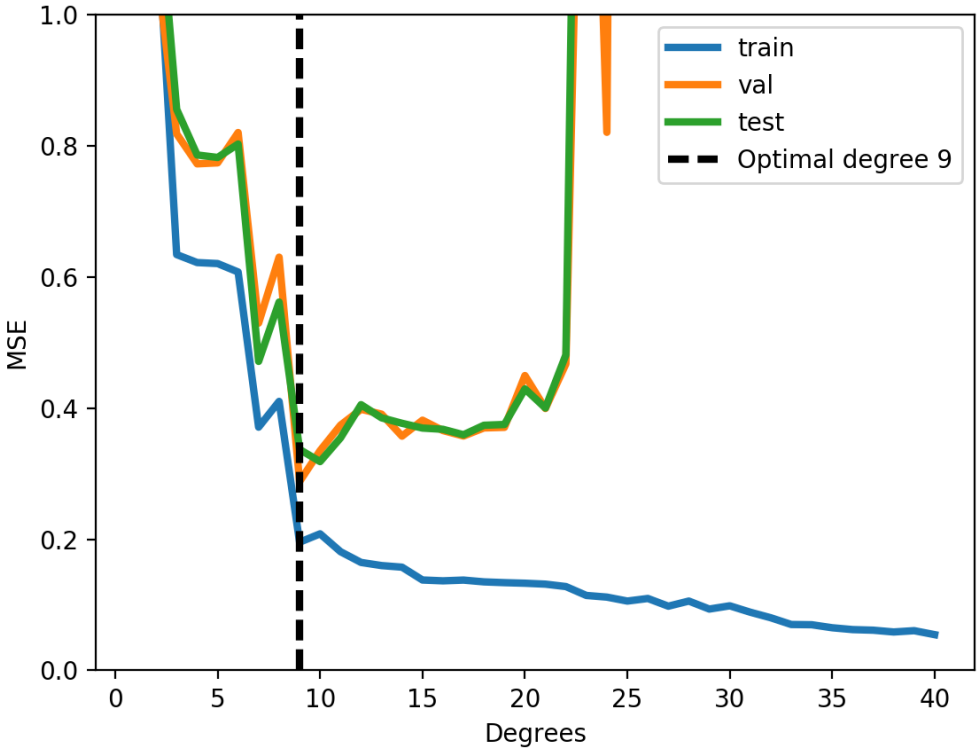


Figure 1.15: *Training, validation and testing errors as a function of the RBF degree.*

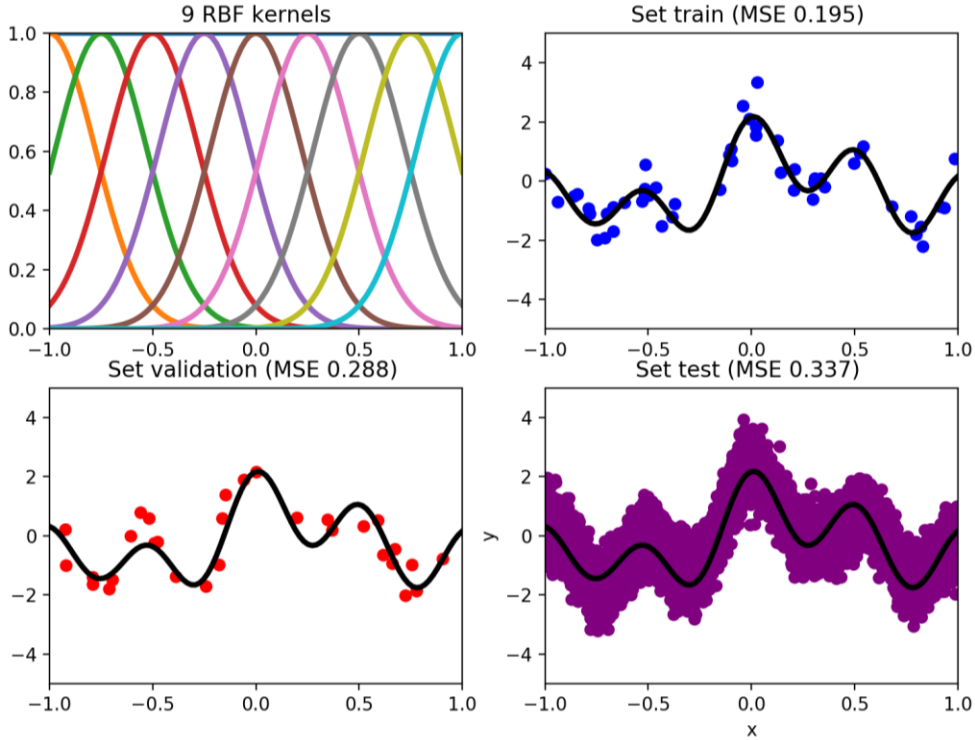


Figure 1.16: *Results of the linear regression with the lowest validation error (the RBF degree 9).*

By looking at the figures 1.13 and 1.15 it is easy to observe the same isue with the importance of the using validation data set like with polynomial regression. By using degree 9 of the RBFs we got the lowest testing error. With RBFs even without using a validation set, in the worst overfitting case, we did not get such enourmous high testing error as with the highest degree of polynomial regression. Therefore the RBF model fits our data better.

## 2 Logistic Regression

### 2.1 Derivation of Gradient

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right)$$

$$y^{(i)} log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) =$$

$$= y^{(i)} log\left(\frac{1}{1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\right) + \left(1-y^{(i)}\right)\log\left(1-\frac{1}{1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\right) =$$

$$= -y^{(i)}\log(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}) + \left(1-y^{(i)}\right)\log\left(\frac{1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}-1}{1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\right)$$

$$= -y^{(i)}\log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) + \left(1-y^{(i)}\right)\left(\log\left(e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - \log(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}})\right) =$$

$$= -y^{(i)}\log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) + \log\left(e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - \log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - y^{(i)}\log\left(e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) + y^{(i)}\log(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}})) =$$

$$= \log\left(e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - \log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - y^{(i)}\log\left(e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) =$$

$$= -\boldsymbol{x}^T\boldsymbol{\theta} - \log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) - y^{(i)}(-\boldsymbol{x}^T\boldsymbol{\theta}) =$$

$$= -\log\left(e^{\boldsymbol{x}^T\boldsymbol{\theta}}\right) - \log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right) + y^{(i)}\boldsymbol{x}^T\boldsymbol{\theta} =$$

$$= -(\log\left(e^{\boldsymbol{x}^T\boldsymbol{\theta}}\right) + \log\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right)) + y^{(i)}\boldsymbol{x}^T\boldsymbol{\theta} =$$

$$= -(\log\left(e^{\boldsymbol{x}^T\boldsymbol{\theta}}\left(1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}\right)\right)) + y^{(i)}\boldsymbol{x}^T\boldsymbol{\theta} =$$

$$= -\left(\log\left(e^{\boldsymbol{x}^T\boldsymbol{\theta}}+1\right)\right) + y^{(i)}\boldsymbol{x}^T\boldsymbol{\theta}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial \boldsymbol{\theta}}\left(y^{(i)} log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right)$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial \boldsymbol{\theta}}\left(-\left(\log\left(e^{\boldsymbol{x}^T\boldsymbol{\theta}}+1\right)\right) + y^{(i)}\boldsymbol{x}^T\boldsymbol{\theta}\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-\left(\frac{1}{e^{\boldsymbol{x}^T\boldsymbol{\theta}}+1}\right)e^{\boldsymbol{x}^T\boldsymbol{\theta}}\boldsymbol{x}^{(i)} + y^{(i)}\boldsymbol{x}^T\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{(e^{\boldsymbol{x}^T\boldsymbol{\theta}}+1)e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\,\boldsymbol{x}^{(i)} + y^{(i)}\boldsymbol{x}^T\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{e^0+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\,\boldsymbol{x}^{(i)} + y^{(i)}\boldsymbol{x}^T\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{1+e^{-\boldsymbol{x}^T\boldsymbol{\theta}}}\,\boldsymbol{x}^{(i)} + y^{(i)}\boldsymbol{x}^T\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)} + y^{(i)}\boldsymbol{x}^{(i)}\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) + y^{(i)}\right)\boldsymbol{x}^{(i)}$$

## 2.2 Logistic Regression training with gradient descent and scipy.optimize

### 2.2.1Gradient descent

#### 2.2.1.1

The function check_gradient calculates a difference between two gradients and checks whether the value is located in given range.

#### 2.2.1.2

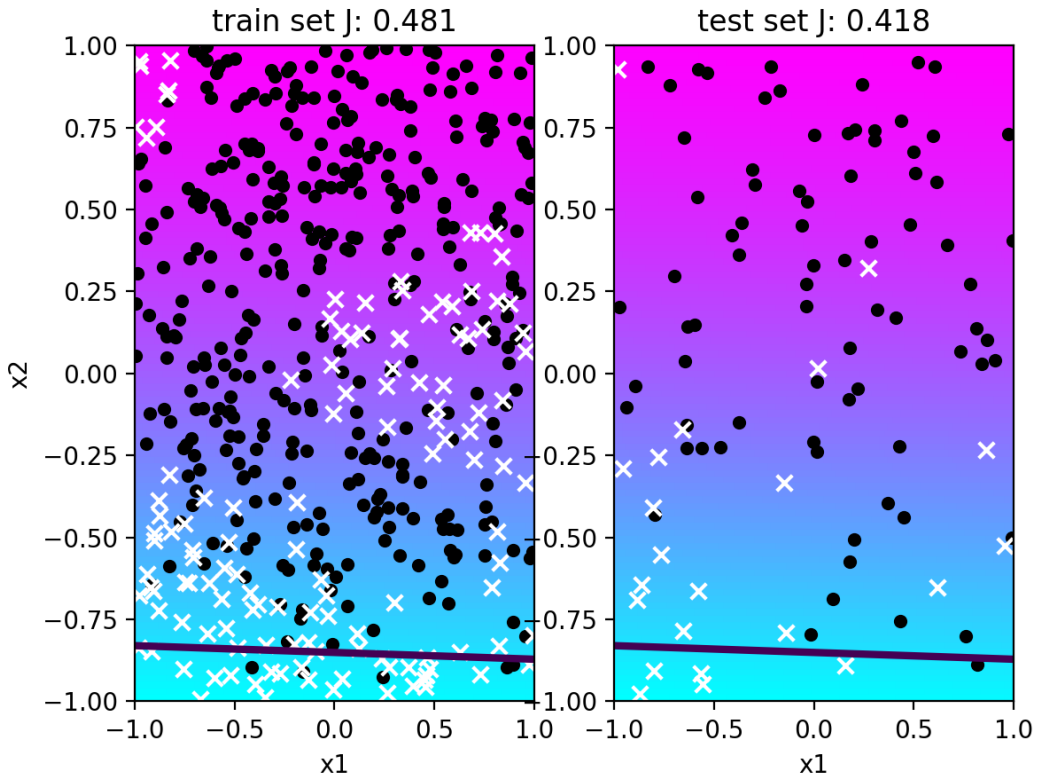


Figure 2.1: *Results of the logistic regression with degree l=1, 20 iterations and learning rate η=1.*

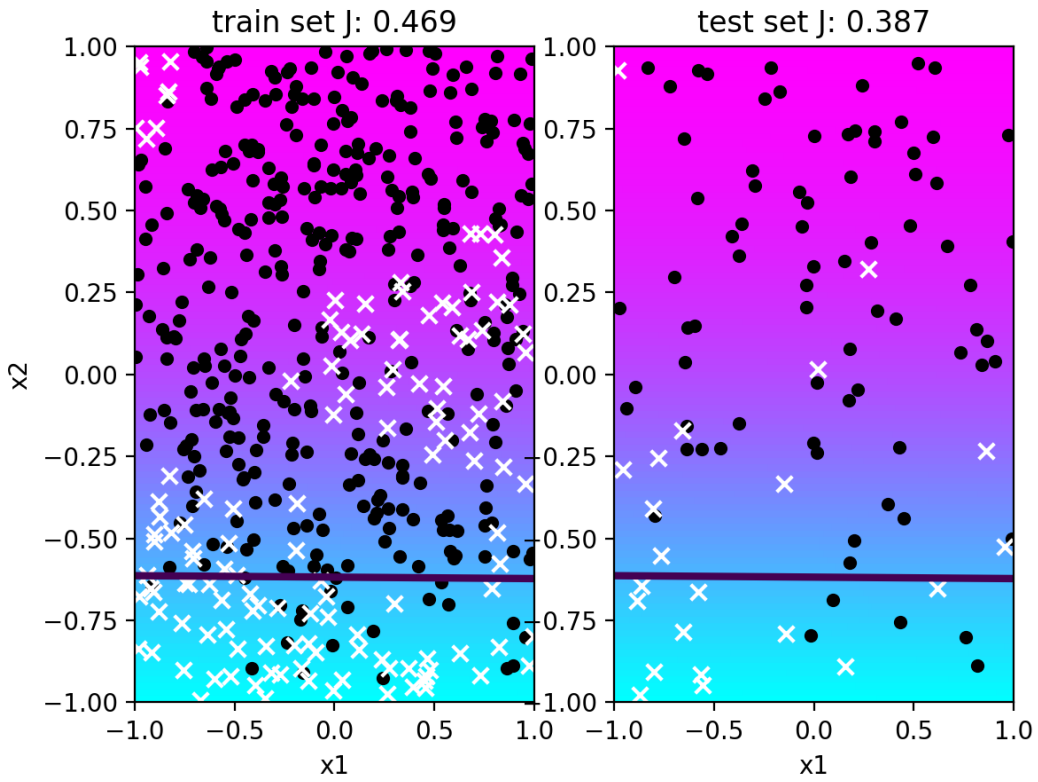


Figure 2.2: *Results of the logistic regression with degree l=1, 2000 iterations and learning rate η=1.*

In the case of low number of iterations, the model has not enough time to learn well. In the case of too many iterations problems with overfitting the data can occure.

2.2.1.3

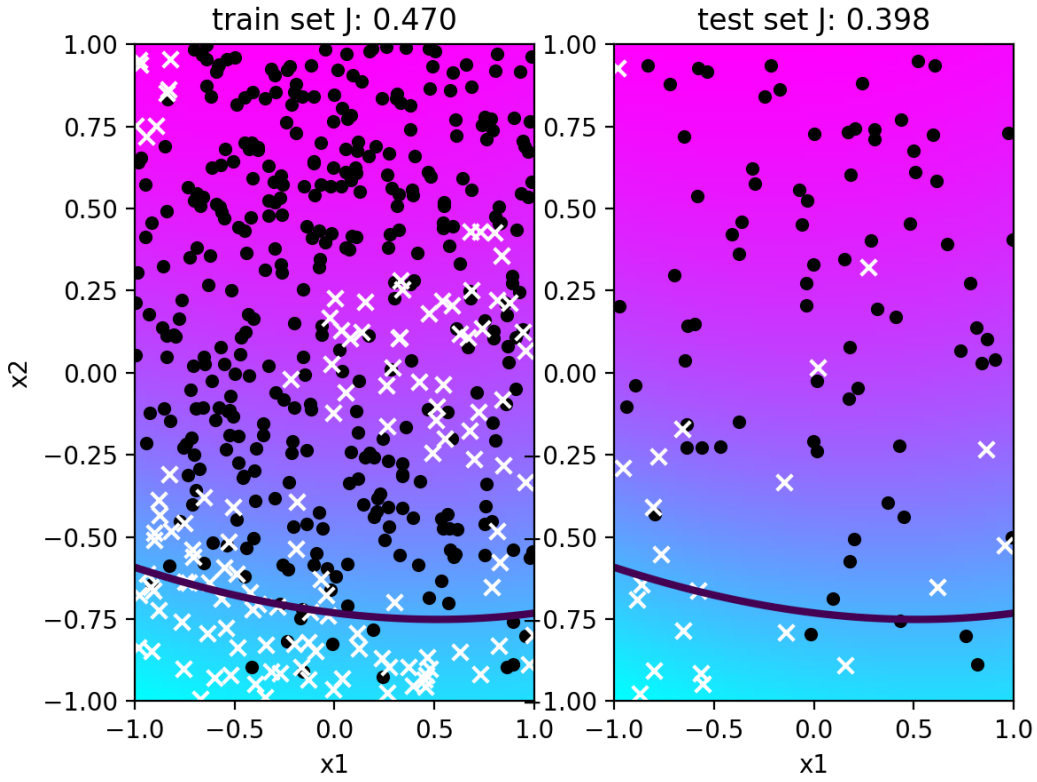


Figure 2.3: *Results of the logistic regression with degree l=2, 200 iterations and learning rate η=0.15.*

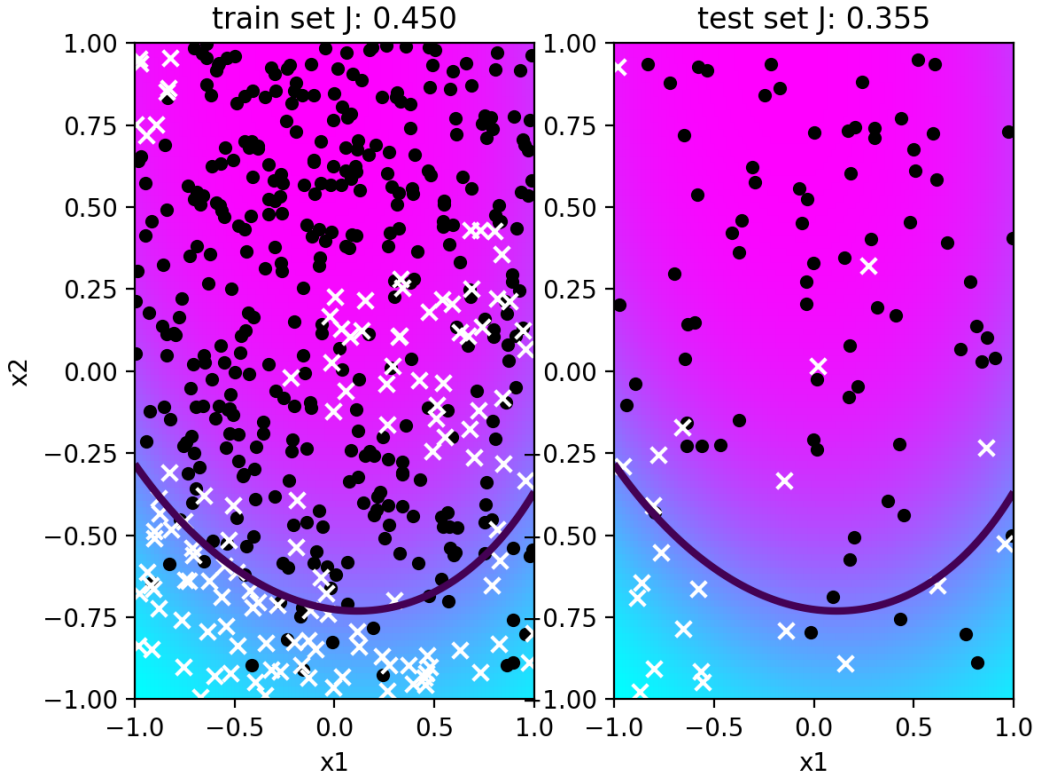


Figure 2.4: *Results of the logistic regression with degree l=2, 200 iterations and learning rate η=1.5.*

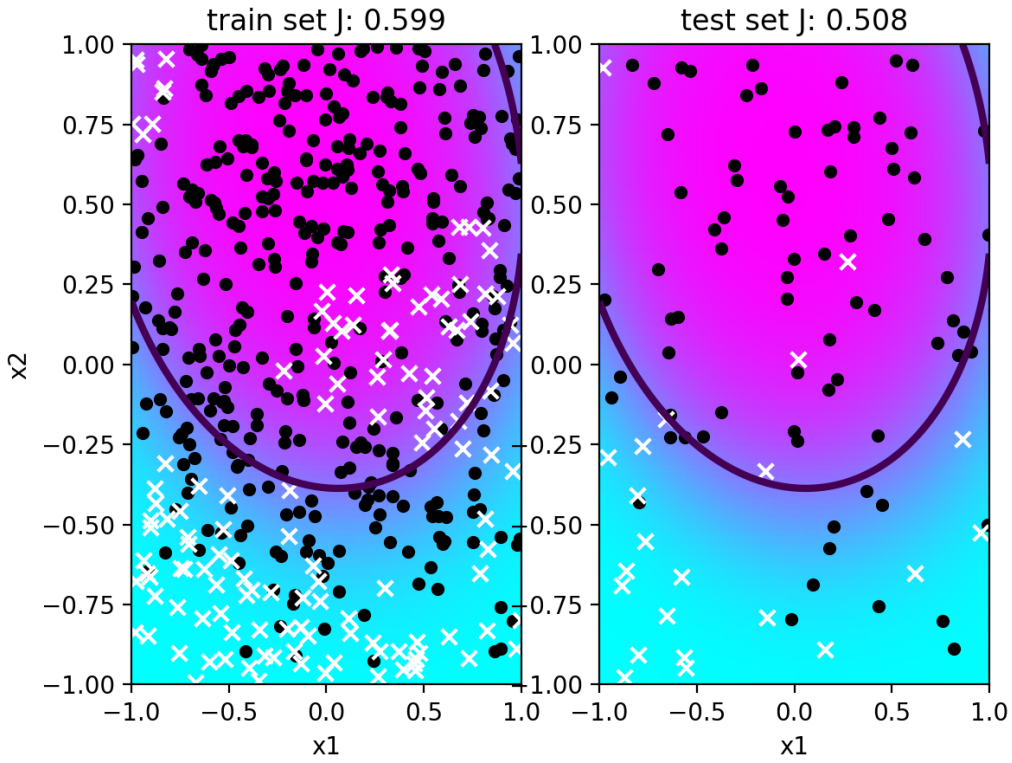


Figure 2.5: *Results of the logistic regression with degree l=2, 200 iterations and learning rate η=15.*

In case the learning rate is too large, the step between interations is to big and the learning will not be able to reach the minimum of the cost. In the other case, the learning is too small, the learning process is updating too slowly and it will not be able the reach the minimum as well.

2.2.1.4

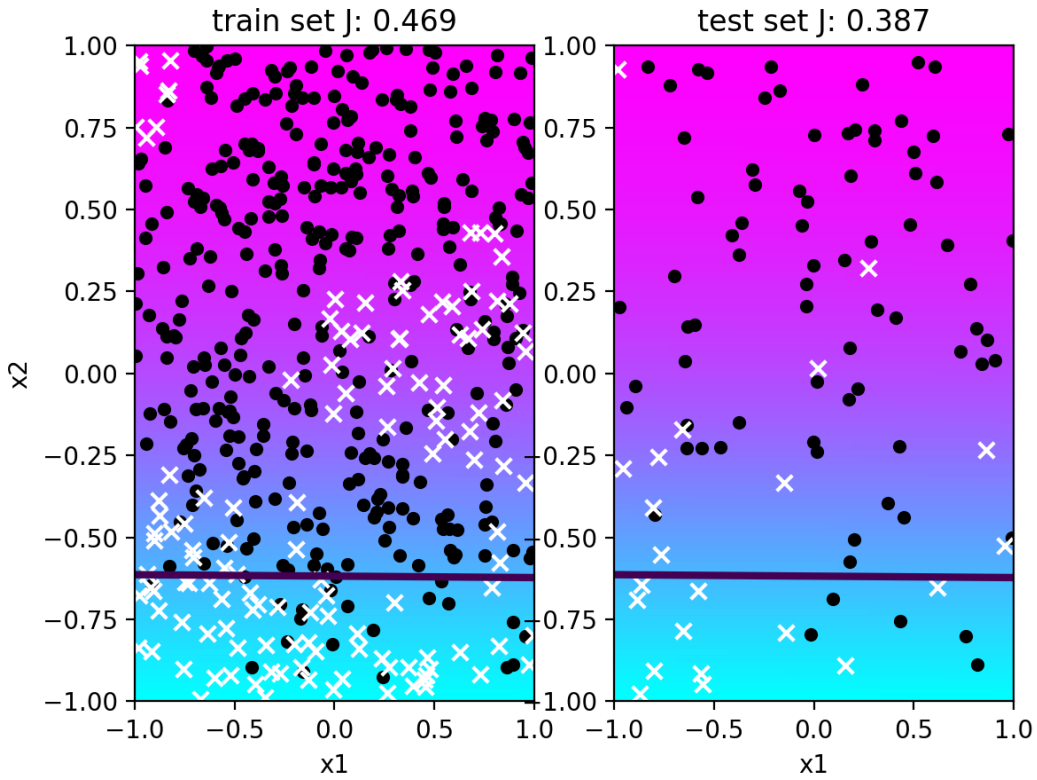


Figure 2.6: *Results of the logistic regression with degree l=1, 2000 iterations and learning rate η=1.5.*

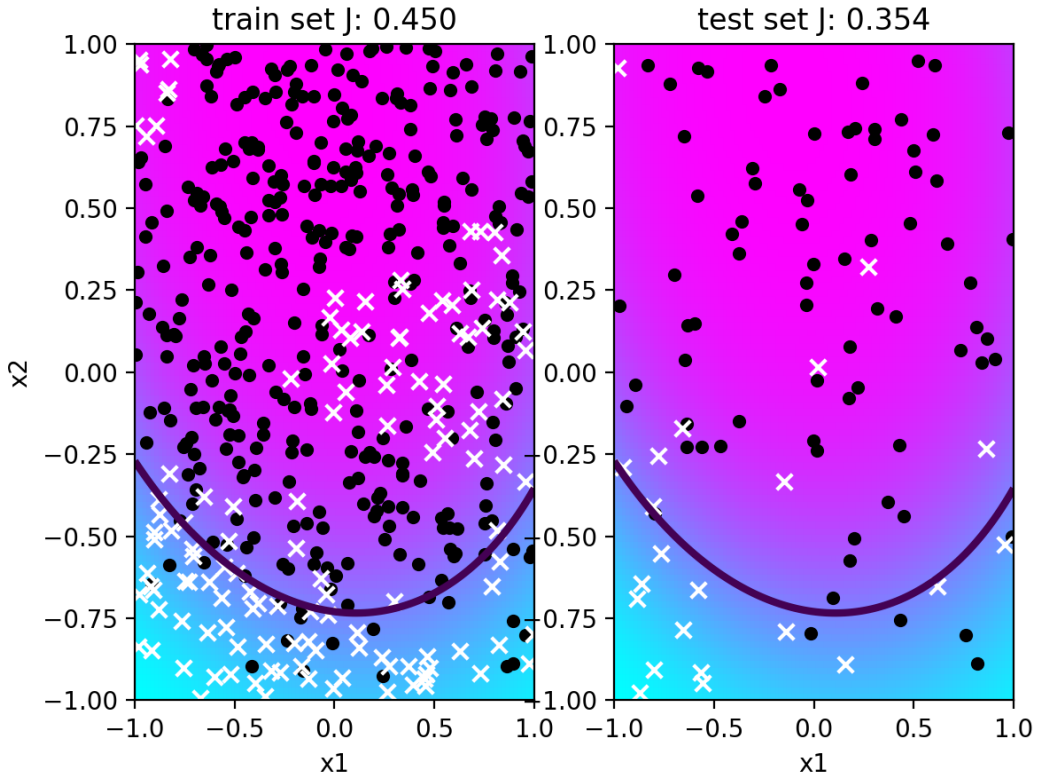


Figure 2.7: *Results of the logistic regression with degree l=2, 2000 iterations and learning rate η=1.5.*

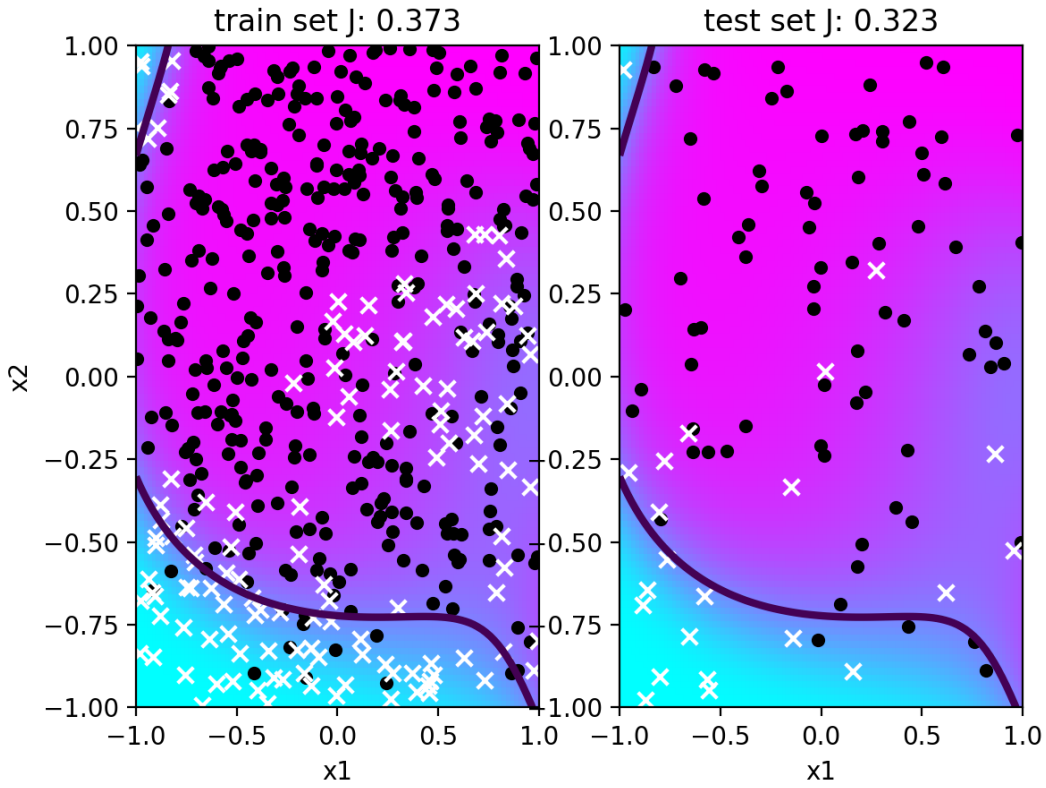


Figure 2.8: *Results of the logistic regression with degree l=5, 2000 iterations and learning rate η=0.15.*

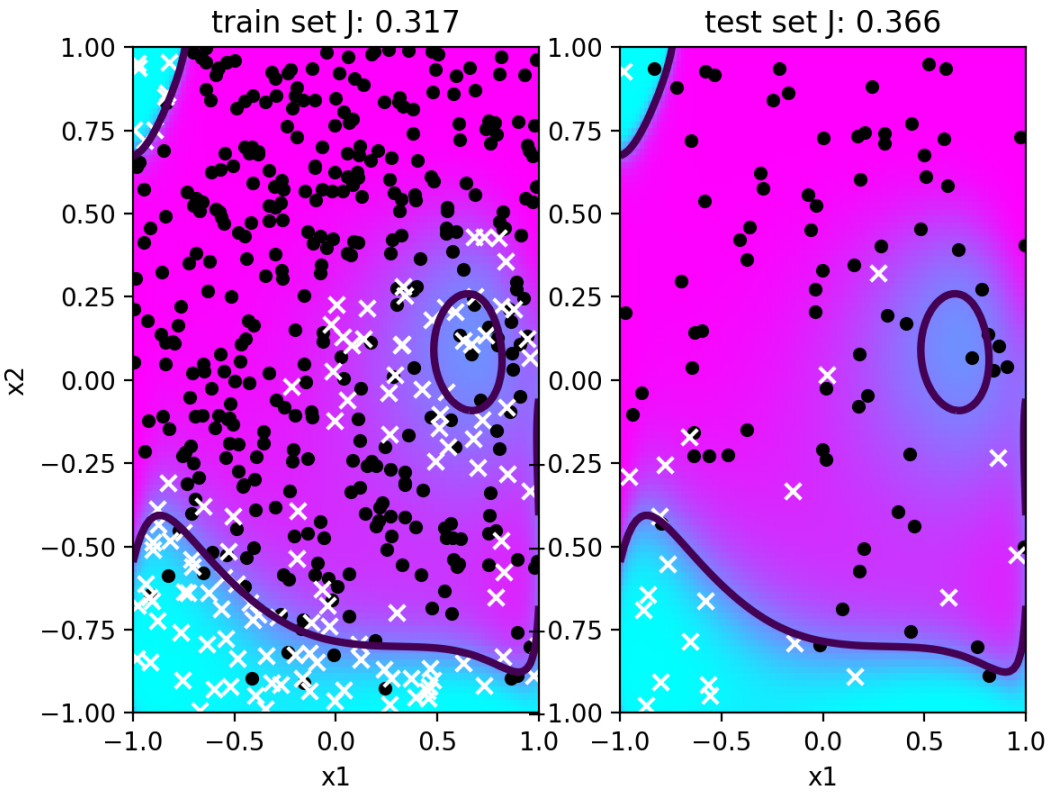


Figure 2.9: *Results of the logistic regression with degree l=15, 2000 iterations and learning rate η=1.5.*

The model on the figure 2.8 seems to be the best fit for the data. This model achieved the lowest error on the testing data set 0.323. The degree of *l*=5 with learning rate of 0.15 and 2000 iterations is the most appropriate one.

2.2.1.5

After some iterations, it is possible that the update between following iterations may be too small. In that case it is smarter to define a stopping criterion for the model, in order to avoid doing too many iterations pointlessly. It makes sense to set that criterion for the gradient smaller than $10^{-10}$.

### 2.2.2 Adaptive gradient descend

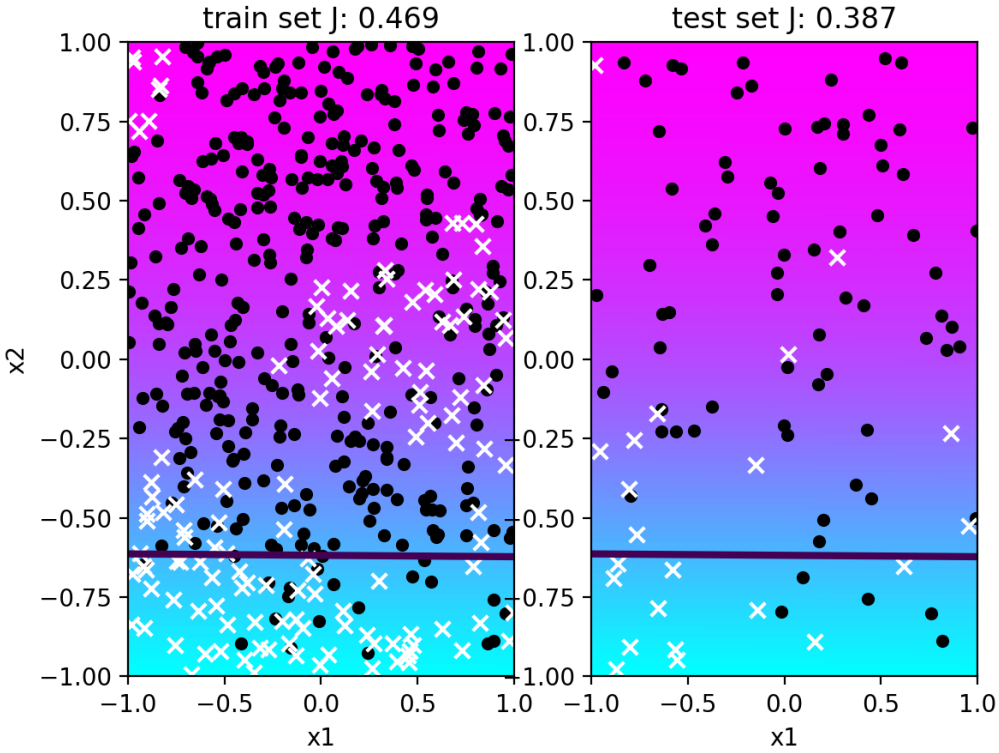


Figure 2.10: *Results of the logistic regression with degree l=1 gained with adaptive gradient descend ($\eta=7.41x1^{-9}$)*

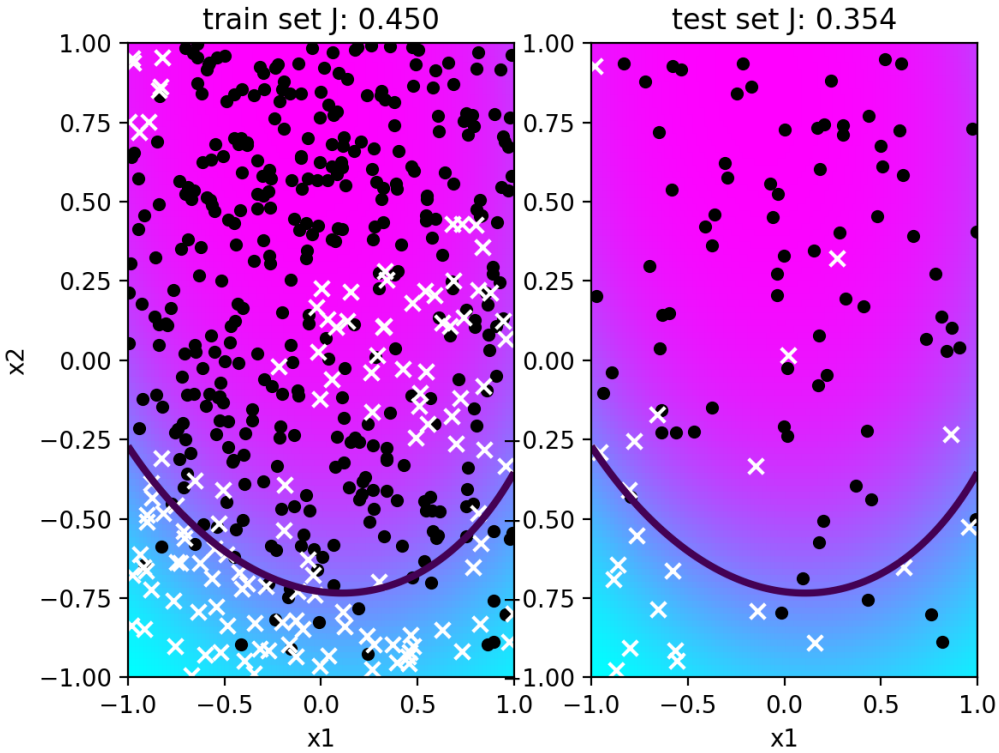


Figure 2.11: *Results of the logistic regression with degree l=2 gained with adaptive gradient descend. ($\eta=1.63x10^{-7}$)*

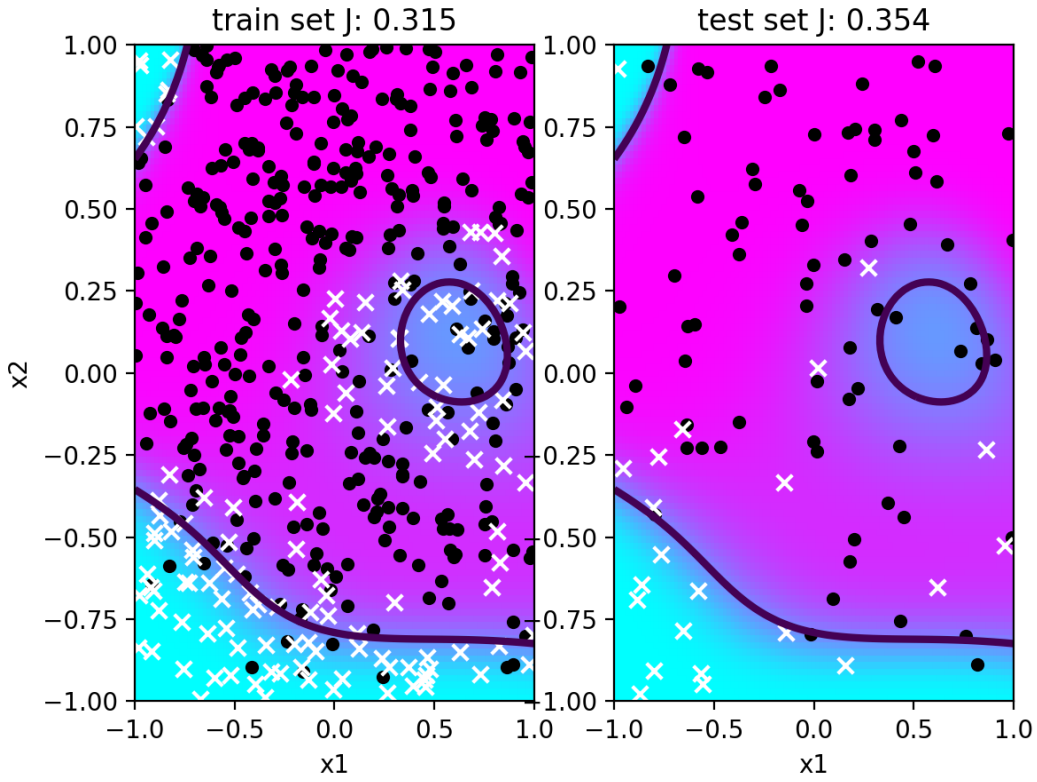


Figure 2.12: *Results of the logistic regression with degree l=5 gained with adaptive gradient descend (η=12.5).*

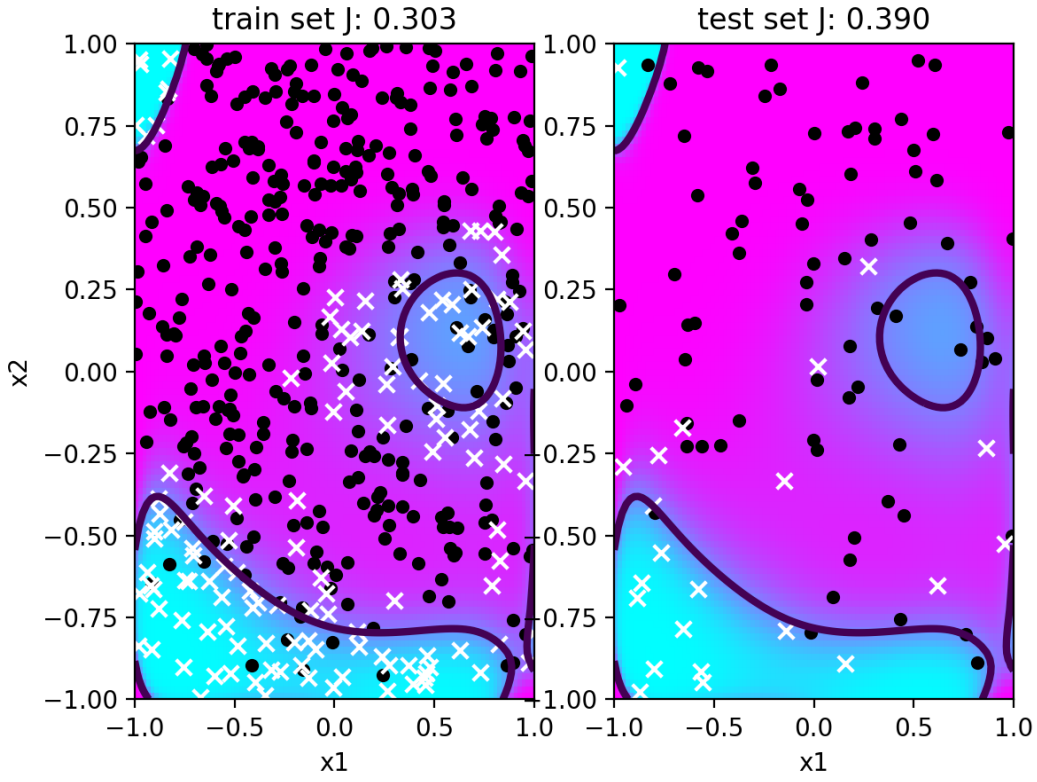


Figure 2.13: *Results of the logistic regression with degree l=15 gained with adaptive gradient descend (η=12.5).*

With both of the first degrees, degree 1 and 2, we got exactly the same error values as with the non-adaptive variant. The adequate learning rate was in both of these cases significantly lower with using adaptive method.

With degree 5 and degree 15, the adaptive method achieved better tranning set results, but the testing error was lower by using our own pair of parameters before. Therefore the model learned with the adaptive method seems to be a bit over-fitting in this particular case. Learning rates set by the adaptive method were higher with these degrees than ours.

### 2.1.3 Scipy optimizer

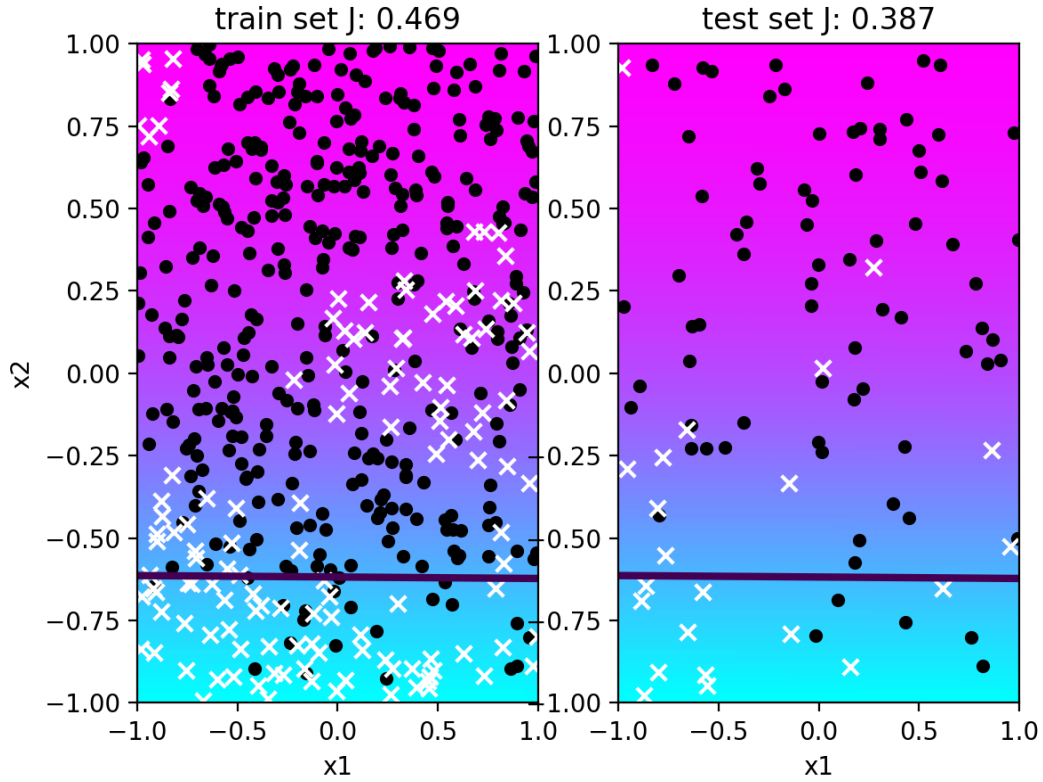


Figure 2.14: *Results of the logistic regression with degree l=1 gained with the Scipy optimizer.*

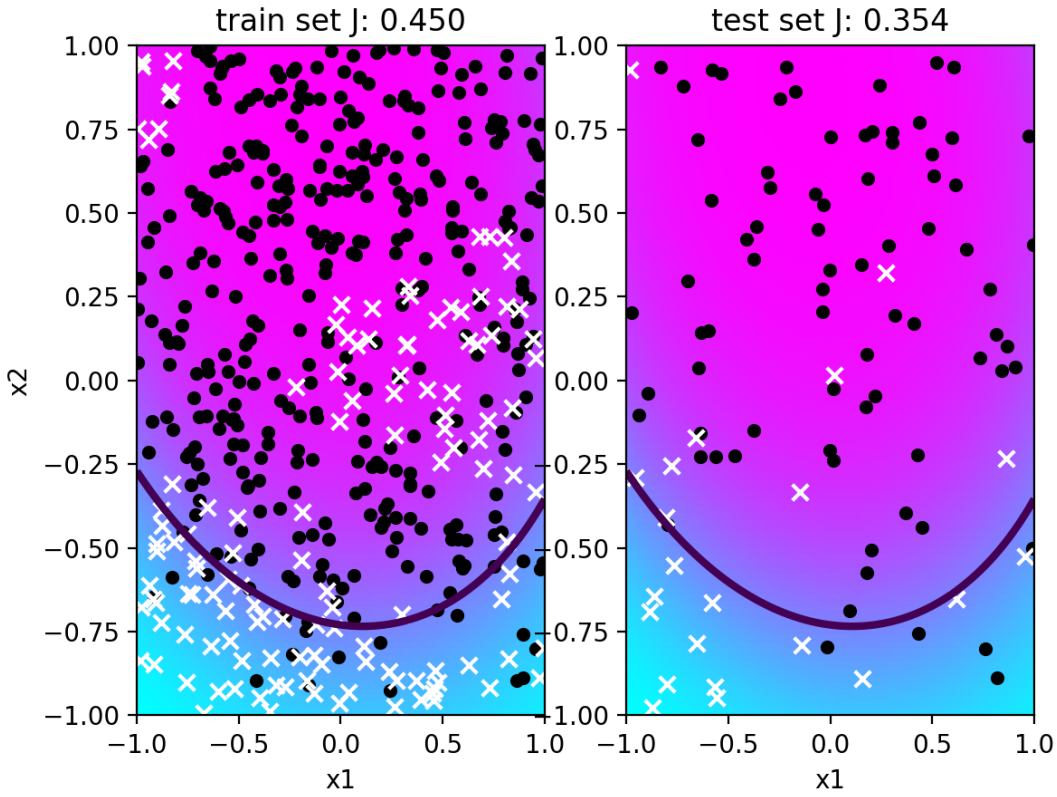


Figure 2.15: *Results of the logistic regression with degree l=2 gained with the Scipy optimizer.*

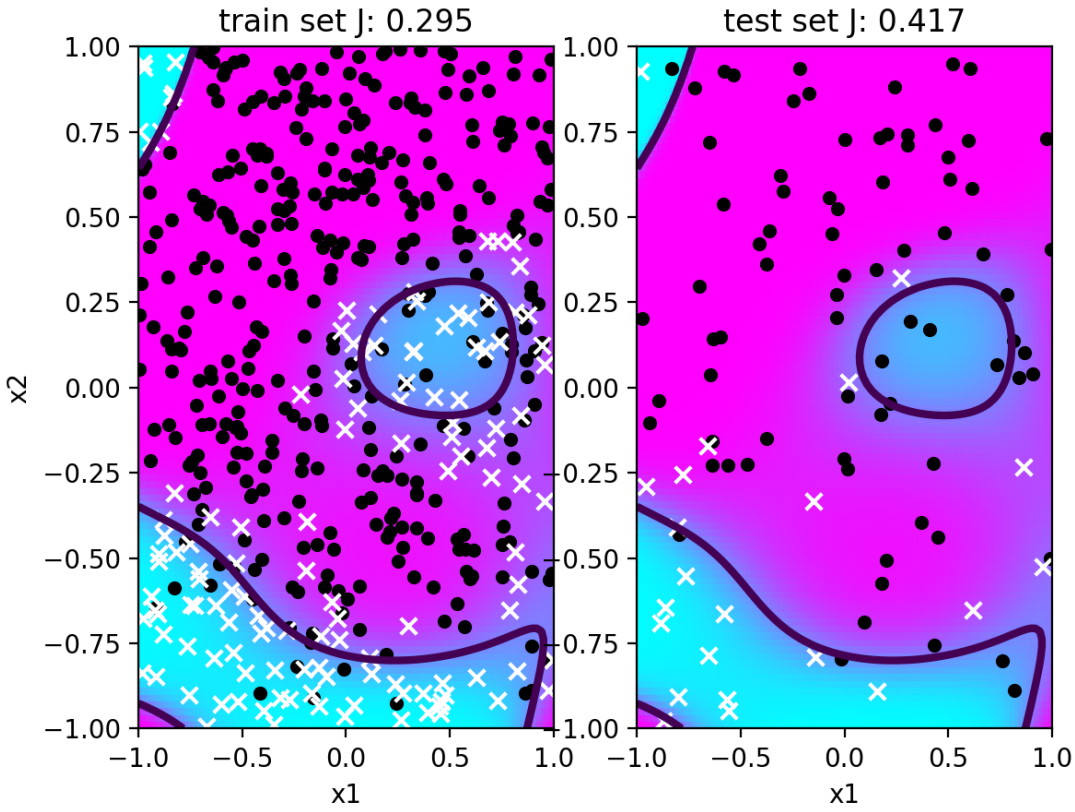


Figure 2.16: *Results of the logistic regression with degree l=5 gained with the Scipy optimizer.*

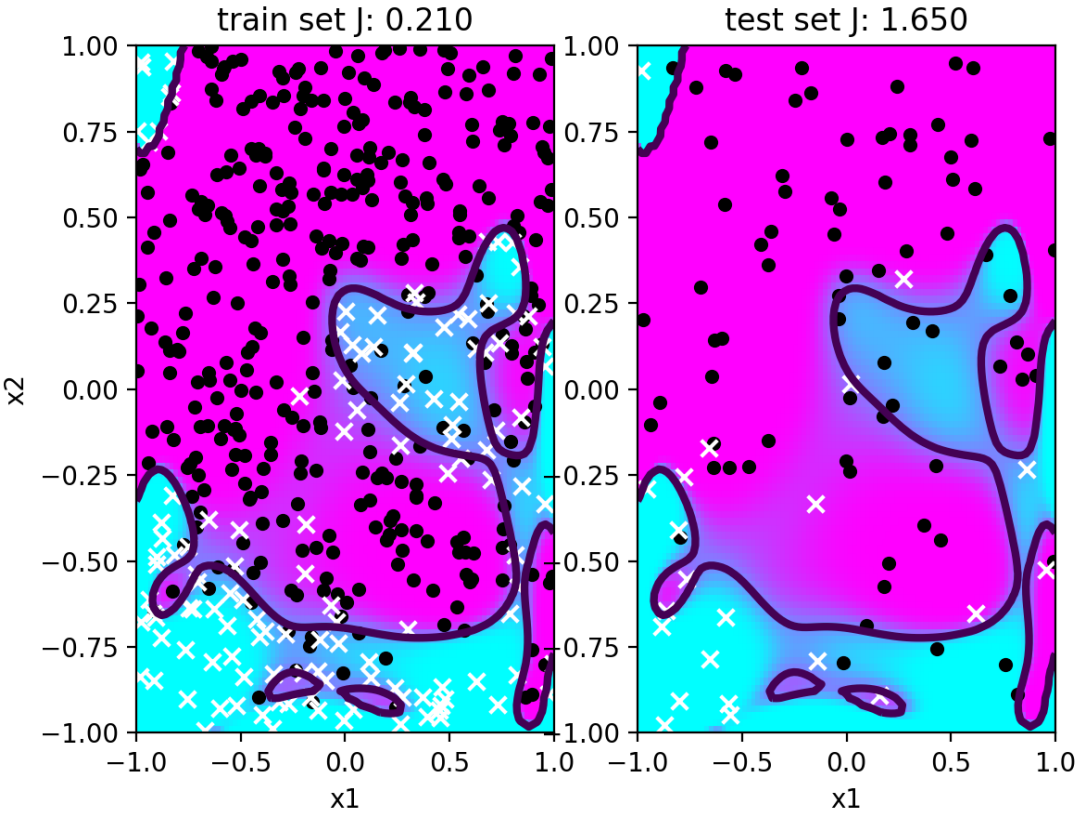


Figure 2.17: *Results of the logistic regression with degree l=15 gained with the Scipy optimizer.*

By looking at the resulting errors obtained with using scipy optimizer we can say that we reached the lowest testing errors of all the models before. The hypothesis line in the plot looks pretty nice on both of the figures with higher degree (figure 2.16 and 2.17). But it looks like the scipy optimizer does not care about the testing error at all. Both of these last models seem to be quite overfitting. With the highest degree of 15, the last model even reached the highest testing error of them all.

Before the last optimalization it was quite clear that the degree which fits the best is the degree of 5. But considering only the scipy optimalization, the degree of 2 is enough and fits even better.