Classification of physical activities

Machine learning

Soufiane Salama, michel banken

2019

Index

[Introduction 2](#_Toc6921366)

[Exercise 1: Feature selection 2](#_Toc6921367)

[Exercise 2: Classification: Logistic regression 4](#_Toc6921368)

[2.1 Cost function and gradient 4](#_Toc6921369)

[2.2 Linear model with 2 features 4](#_Toc6921370)

[2.3 Polynomial features from 2 features 6](#_Toc6921371)

[Optimizing lambda 7](#_Toc6921372)

[Variance and bias 8](#_Toc6921373)

[2.4.1 Linear classifier with 8 features 9](#_Toc6921374)

[2.4.2 Non-linear classifier with 8 features 10](#_Toc6921375)

[Variance and bias 10](#_Toc6921376)

[Adding more training examples 11](#_Toc6921377)

[Conclusion 12](#_Toc6921378)

[Appendix A 13](#_Toc6921379)

# Introduction

This assignment is part of the machine learning course.

The dataset is a public collection of labelled accelerometer data recordings to be used for the creation and validation of acceleration models of simple physical activities.

The datasets consists of 6 different activities performed by 30 volunteers:

* Walking
* Walking upstairs
* Walking downstairs
* Sitting
* Standing
* Laying

# Exercise 1: Feature selection

Human activities are classified by a binary classifier, a classifier that only detects two classes, also called one vs rest. One of the activities as described above must be selected as the “interested” activity for classification. By selecting a class as “interested’, the class will be selected as class 1 while the other classes will be assigned to class 0.

By plotting the features with a class selected as “interested” activity, it’s possible to check whether there are distinguished differences between the activities. If there is a clear separation between the activities, the features will be suitable for creating a model. After plotting the possible combinations two features are selected with distinguished separation between them. The plots for all the different activities can be found in appendix 1.

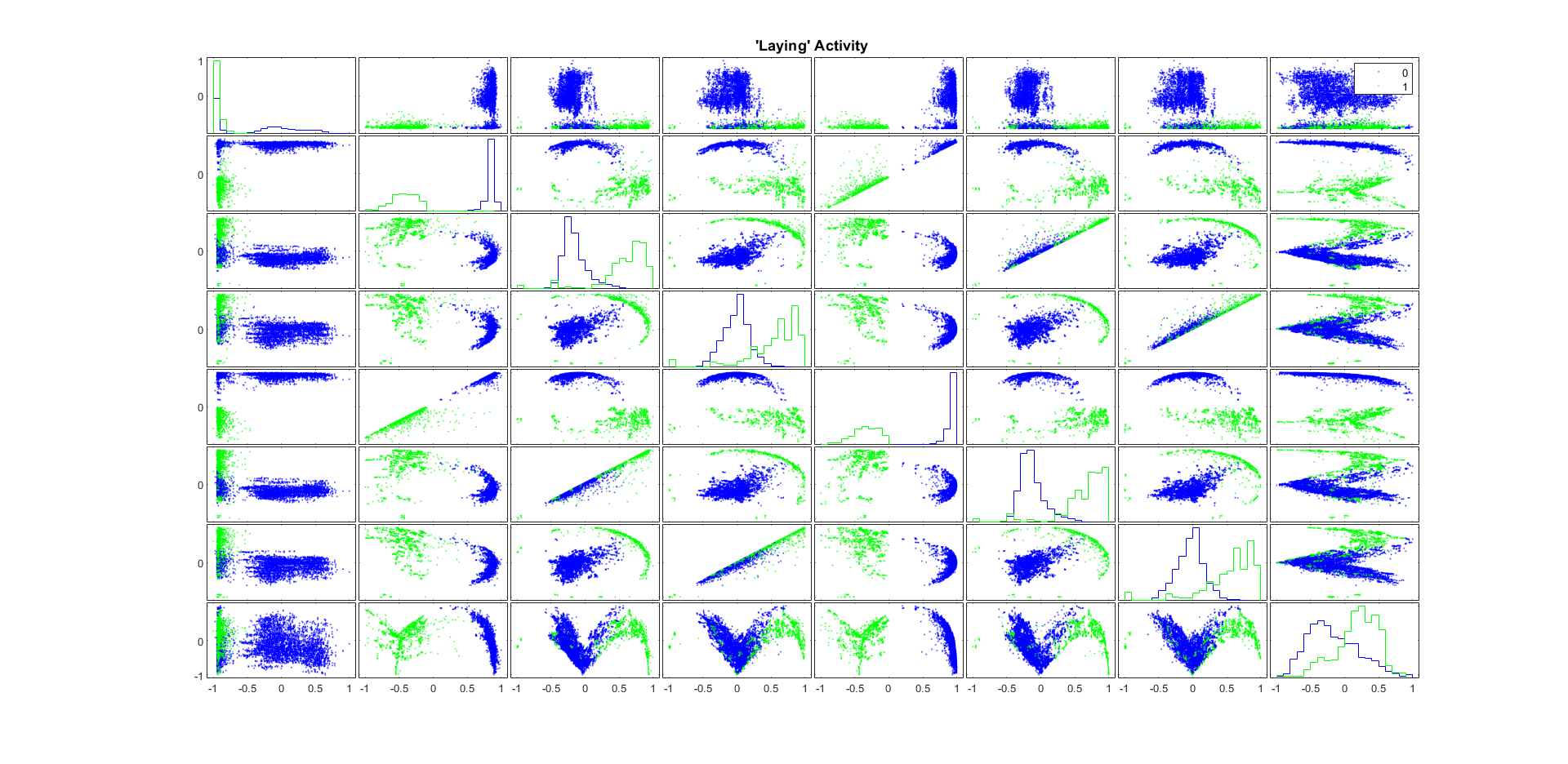


figure 1: visualization feature selection 'laying'

One of those sets of features consists of feature 2 and feature 6. The report will use these two features as selected features. The figure below plots the two features individually for a clear view of the features and separation.

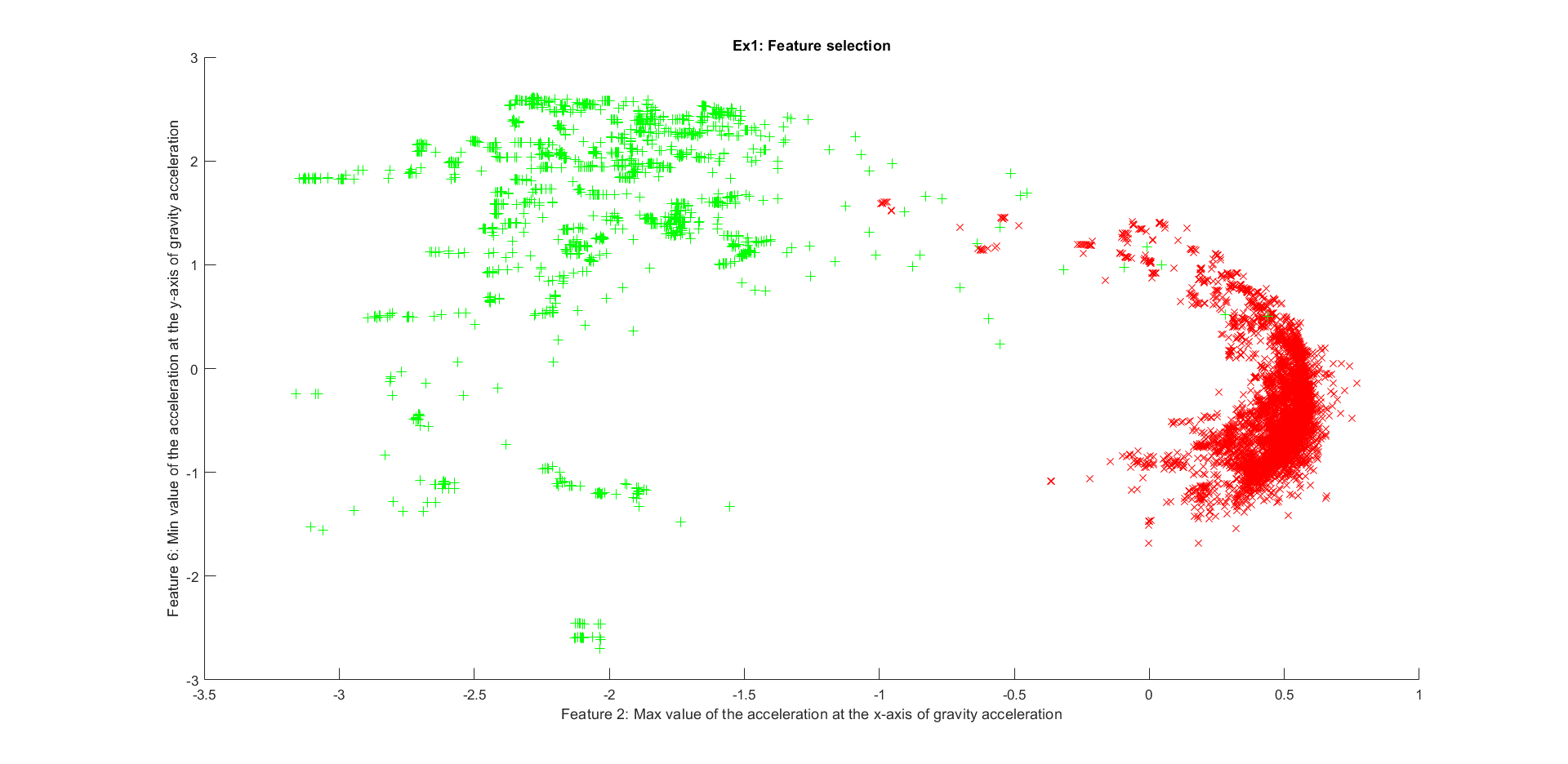


figure 2: feature selection feature 2 and 6

# Exercise 2: Classification: Logistic regression

## 2.1 Cost function and gradient

Adding the MATLAB functions from programming exercise 2.

## 2.2 Linear model with 2 features

After selecting two features, normalization (Mean Normalization) was implemented. The features are now better scaled and show a better range and in general more circular shaped. The axis show that there is a wider range in values.

Mean Normalization:

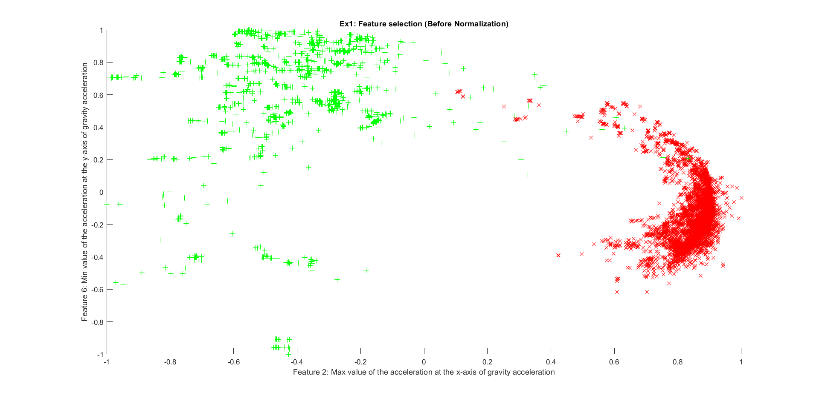
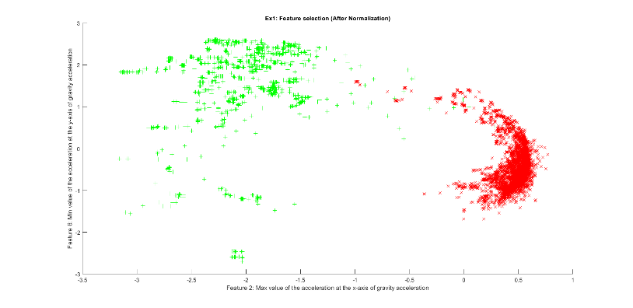
 

figure 3: features after normalization figure 4: features after normalization

After the features are normalized, they are divided into three different datasets: Training, Cross Validation and Test dataset with the ratio of 0.4:0.3:0.3.

Using the data from the training dataset, the binary classifier can be trained and a linear decision boundary constructed using lambda with a value of 0. Because there already was a good separation between the features, the boundary is well constructed and seems to work well for our data.

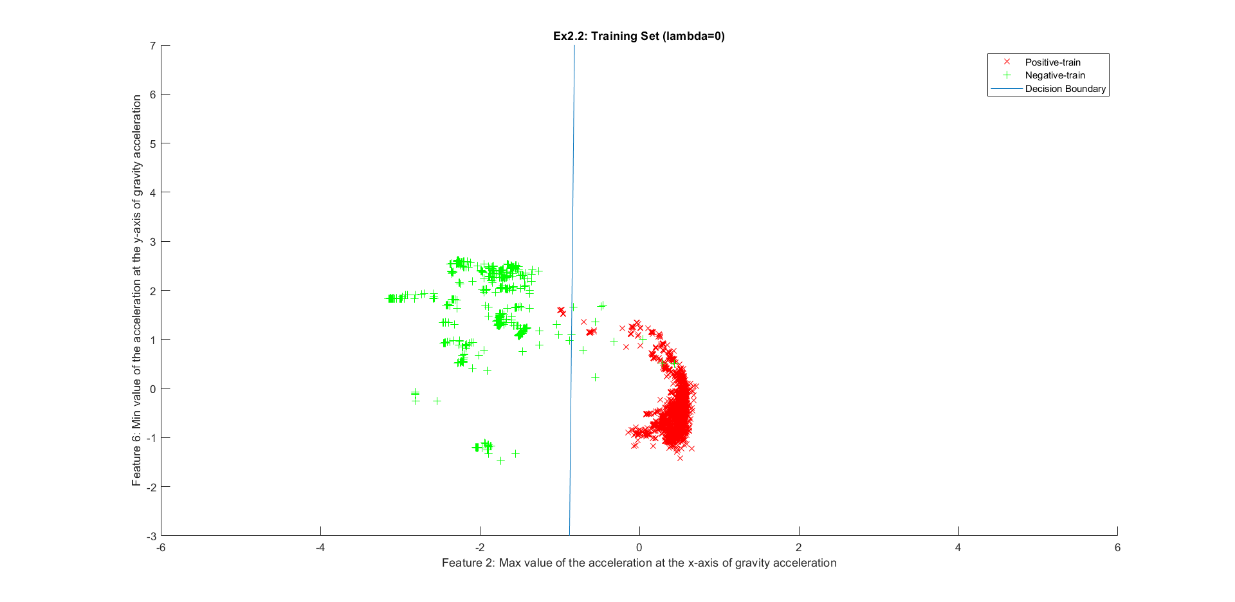


figure 5: Linear decision boundary feature 2 and 6, activity 6

The obtained F1 scores are as followed:

* Training set -> 0.983902
* Cross Validation set -> 1.00

MATLAB has a built-in function “confusionmat” (and “confusiongraph”) that will calculate the ‘true positives’ (2;2), ‘false positives’(1;2) and ‘false negatives’(2;1) that are needed to calculate the precision and recall values .

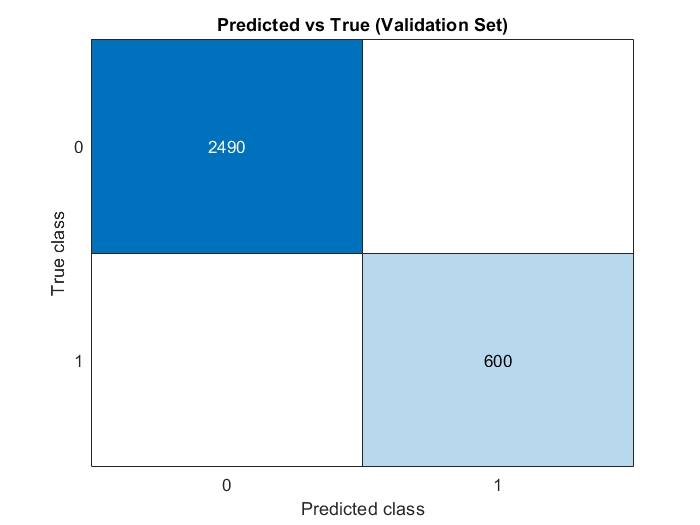
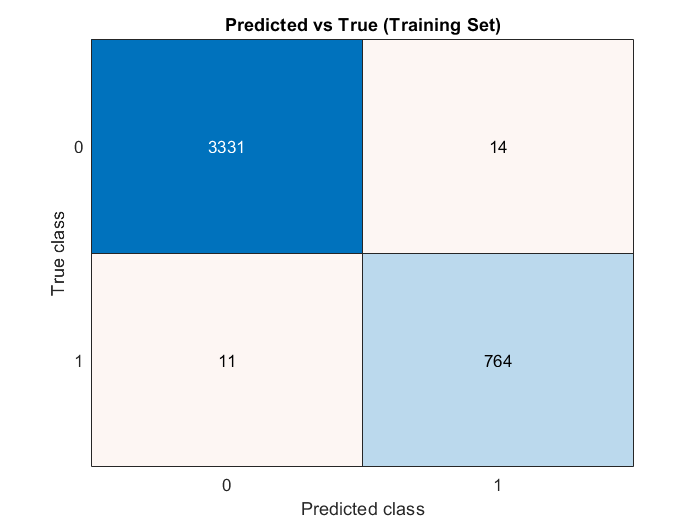


figure 6: confusiongraph training dataset  figure7: confusiongraph validation dataset

The obtained F1 Scores are very high, but this was already known by looking at the plotted decision boundery. If we now for example use two features and an activity which doesn’t show good seperation like features 1 and 8 for activity ‘Walking Upstairs’, we get the following (linear) decision boundary :

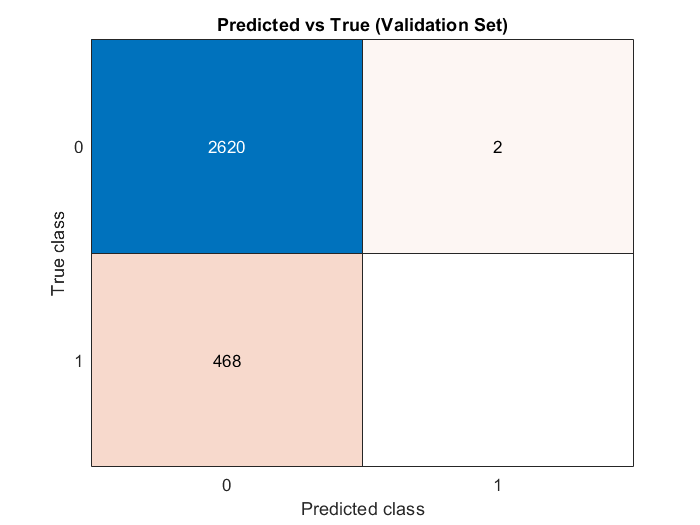
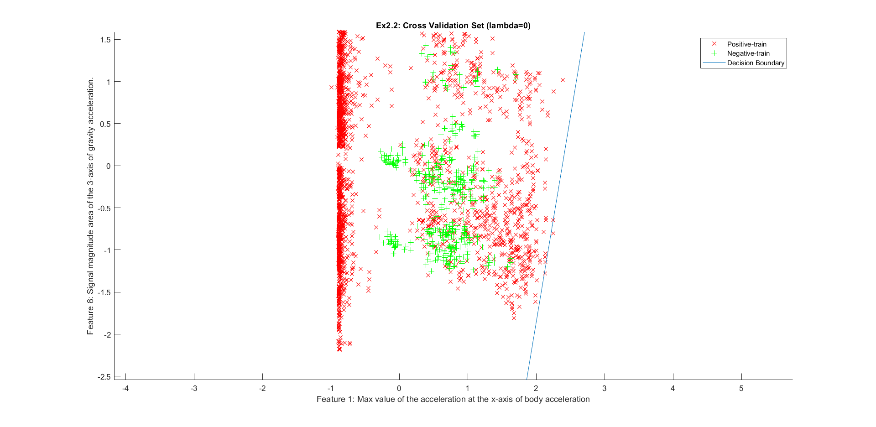


Figure8: linear decision boundery, feature 1 and 8, activity 1 figure 9: confusiongraph feature 1 and 8, activity 1

It is not possible for the algorithm to draw a linear decision boundary because the positive and the negative are not clearly separated. Also the F1 Score cannot be calculated (NaN) because our hypothesis hasn’t predicted any class that is positive as shown in fig. 9. For examples like these, it is not enough to work with a linear hypothesis, instead it is necessary to work with a polynomial hypothesis.

2.3 Polynomial features from 2 features

To improve our hypothesis, we can map the features into polynomial terms. Depending on the situation you can vary the degree of polynomial, for example a degree of 6 will give us 28 features. Using this feature vector, the decision boundary can greatly be improved. In our case where we already had a very successful hypothesis, it wasn’t necessary to do this. In matter fact our F1 Score has even dropped to **0.811359** (Cross Validation Set) by using a 6 degree polynomial as hypothesis.

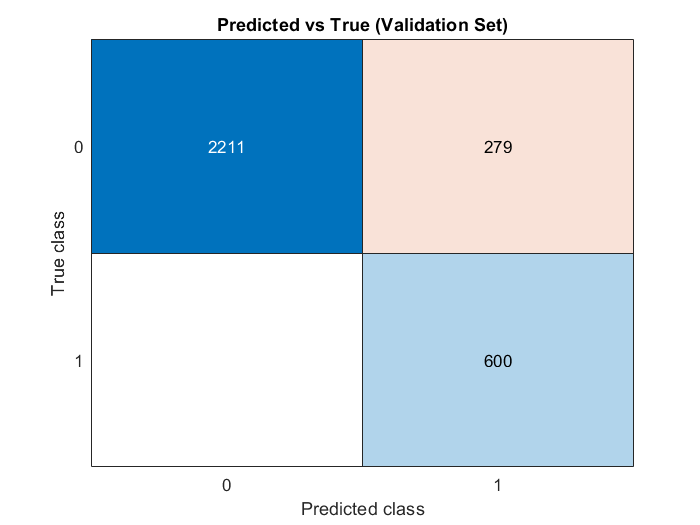
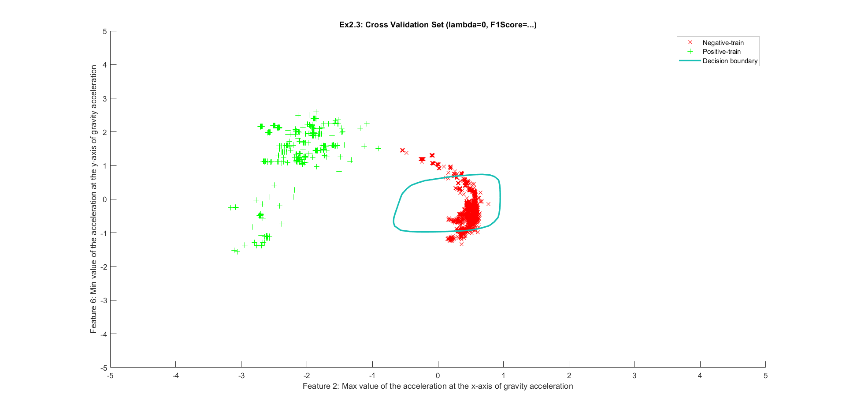


figure 10: polynomial (6) decision boundary, features 2 and 6, activity 6

figure 11: confusiongraph features 2 and 6, activity 6

Even if we mapped the features to a polynomial of degree 2, the F1 Score (0.999166 Cross Validation Set) is still very good but the implementation of the feature mapping has no additional value.

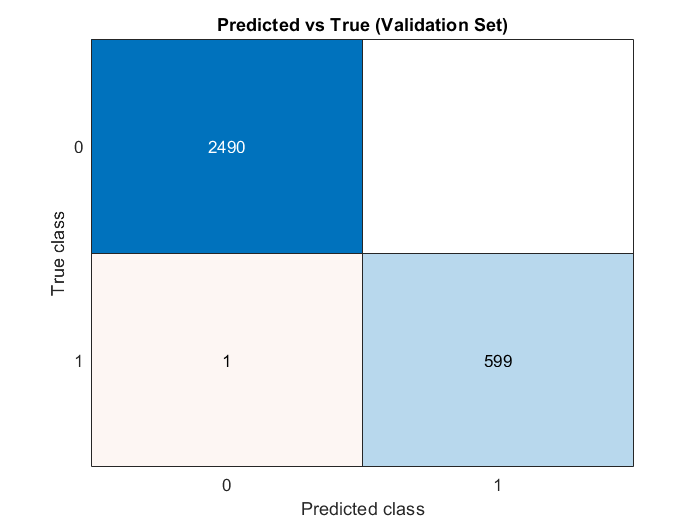
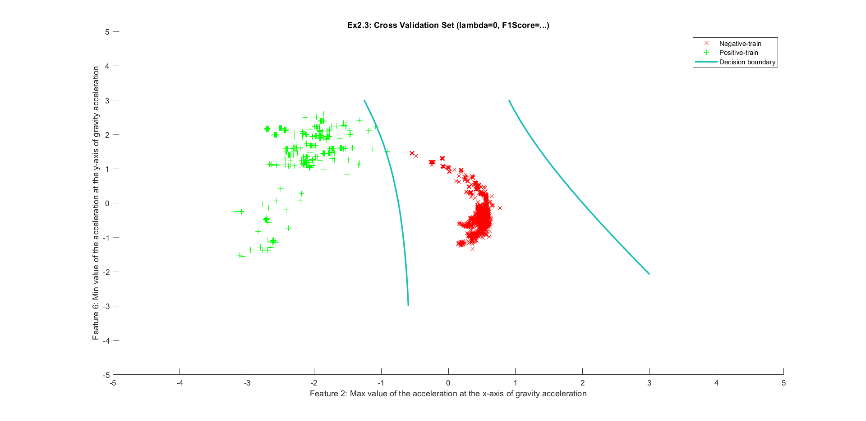


figure 12: polynomial (2) decision boundary, features 2 and 6, activity 6

figure 13: confusiongraph polynomial features (2nd degree) from 2 and 6, activity 6

That is why we continued with features 1 and 8 for the activity of ‘Walking’ so we can still optimize the F1 Score by changing lambda. After mapping the features to a polynomial of degree 6, we get a F1 Score of **0.682199** (lambda=0):

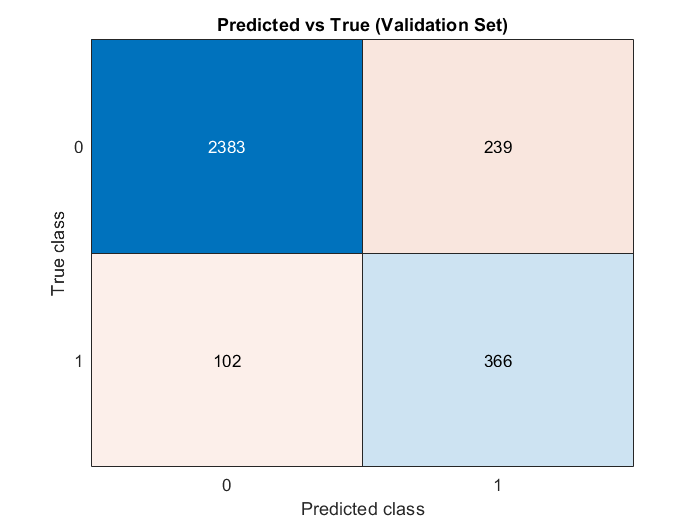
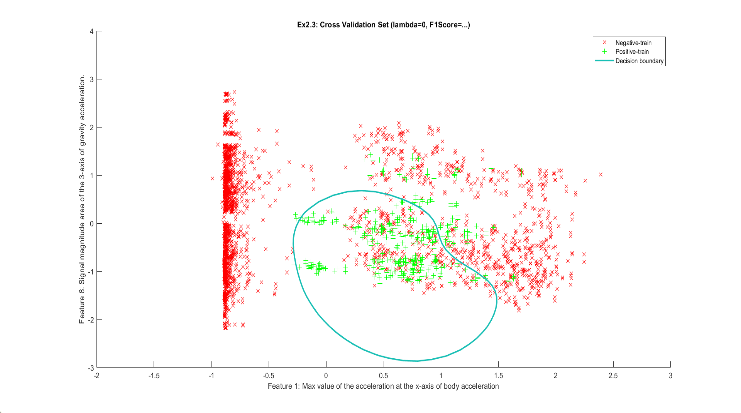


figure 14: polynomial (6) decision boundary, features1 and 8, activity 1

figure 15: confusiongraph features 1 and 8, activity 1

### Optimizing lambda

To optimize the F1 Score we can vary lambda between the interval -3^(10):3^(10), this can be done by using the MATLAB function logspace between -4.77 and +4.77 to generate the values. We used logspace because linspace didn’t give accurate values.

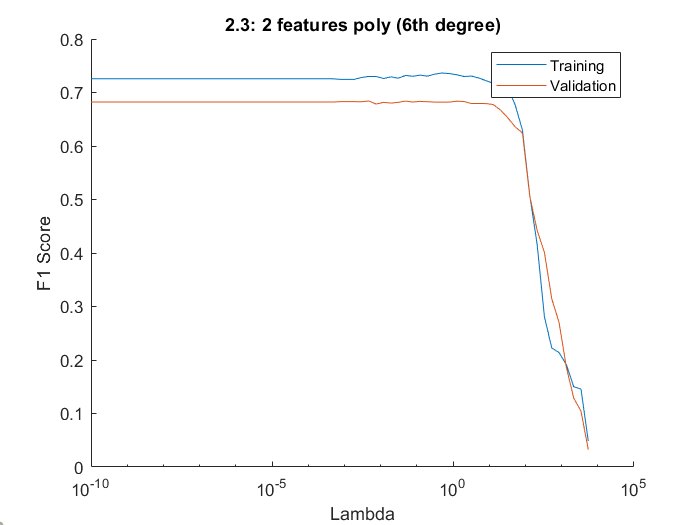


figure 16: F1 score in function of lambda for 2 features

By plotting F1 score in function of lambda an ideal value of lambda can be found where the F1 score is the highest and will give the best results. We can conclude that the F1 score keeps stable at a somewhat maximum for values of lambda under 10.

### Variance and bias

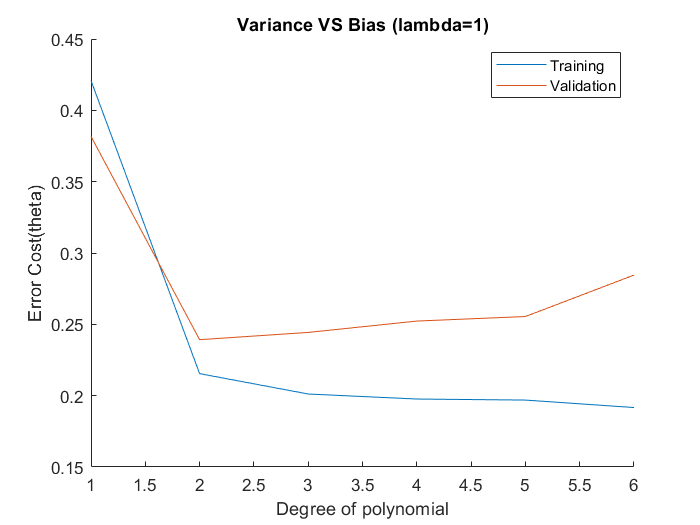
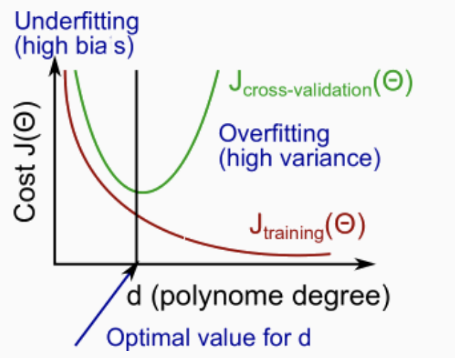


figure 17: cost in function of the degree of polynomial

The variance and bias or over- and underfitting problems is an issue depending on complexity of the model. Plotting the cost of training and validation in function of the degree of polynomial gives us an insight.  
Because we implemented regularization in the cost function, we already prevented variance considerably.

Rule of thumb

Bias problem (underfitting):

* J\_train\_error will be high
* J\_cv\_error will be ≈ as J\_train\_error

Variance problem (overfitting):

* T\_train\_error will be low
* J\_cv\_error will be much higher than J\_train\_error

We can conclude that for degrees of polynomials under 2 we have a bias problem (underfitting) and from there on we keep getting a lower cost. Thanks to regularization we have no variance problem. This means that in our case where we mapped the features to a sixth degree, we have no variance or bias problem.

## 2.4.1 Linear classifier with 8 features

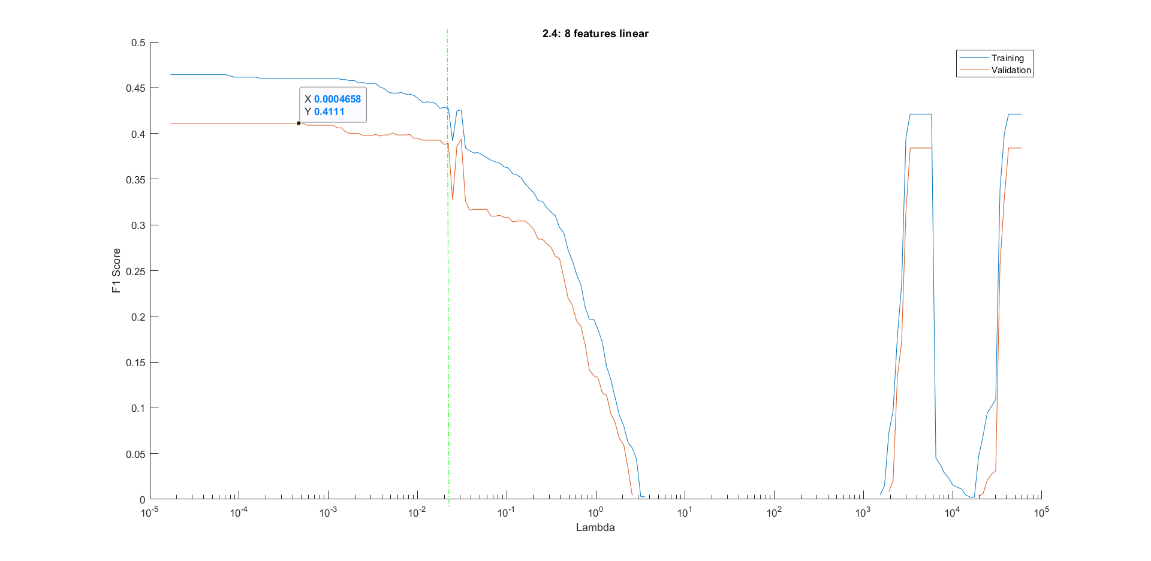
Instead of only using the two ‘ideal’ features, we now calculated the F1 score in function of the eight features. For the activity ‘Walking’, our F1 score has significantly decreased to **0.4111** and from lambda = ± 0.02,the F1 score keeps decreasing. ****

figure 18: F1 score in function of lambda for 8 features, activity walking

If we now for example use the activity ‘Sitting’ to compare, we get the following plot containing ‘good scores’:

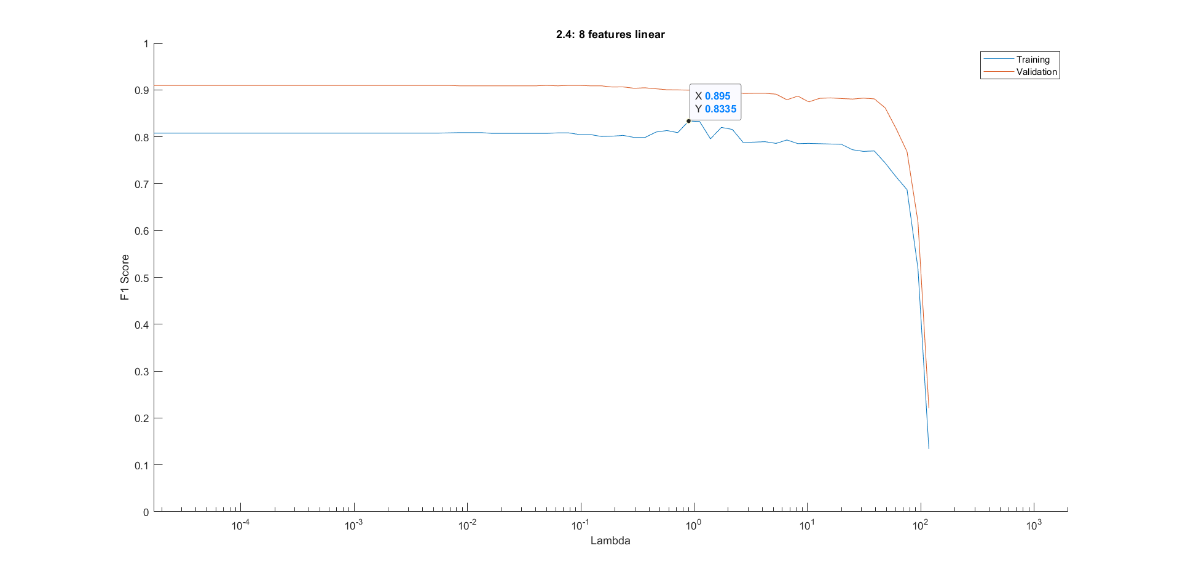


figure 19: F1 score in function of lambda for 8 features, activity sitting

## 2.4.2 Non-linear classifier with 8 features

To improve our previous F1 score (0.4111), where we made use of a linear hypothesis of the eight features, we can now map these features to, for example, a quadratic (degree=2) polynomial.

As for result our F1 score has indeed increased to maximum of **0.7563.**

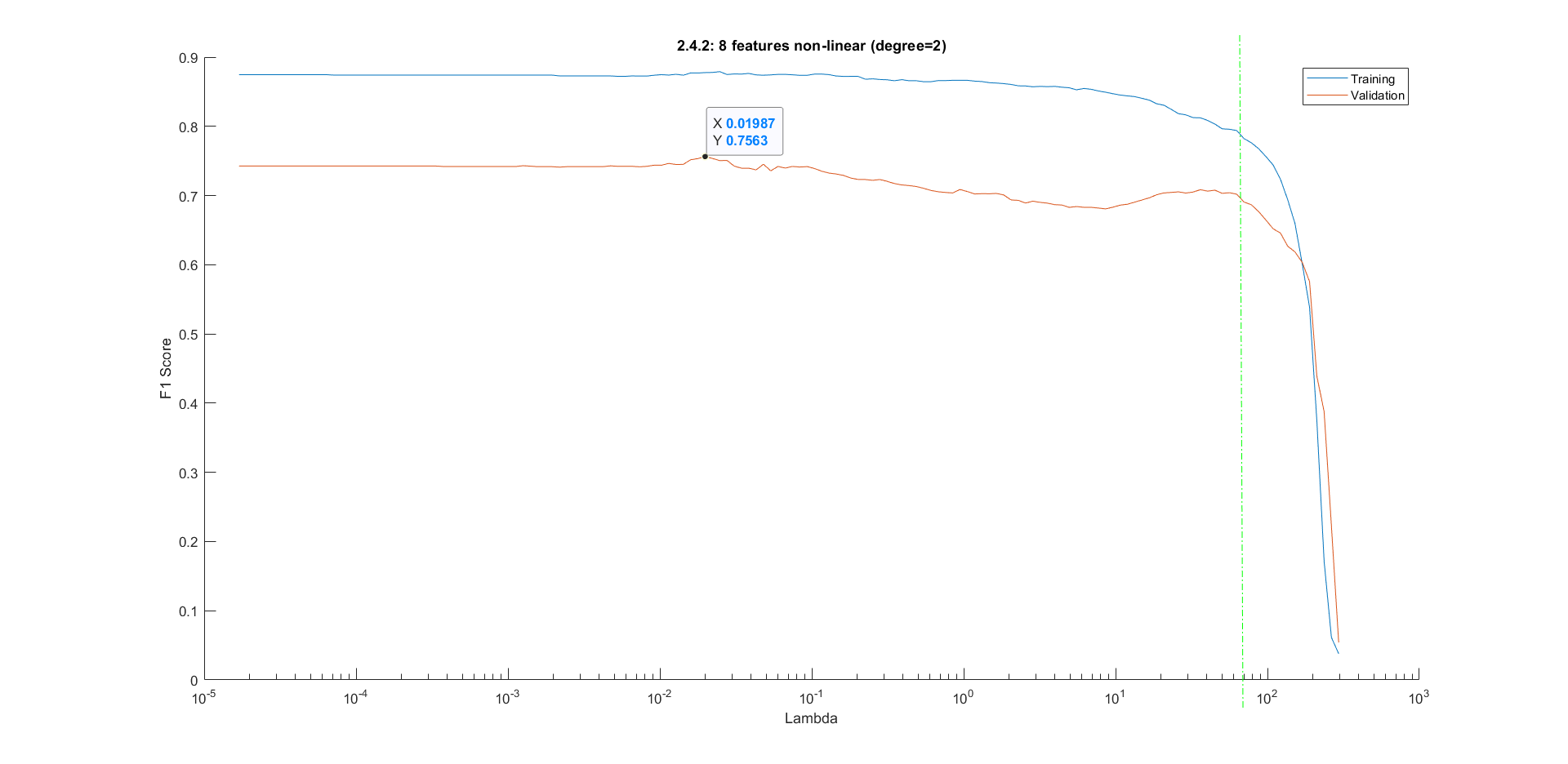


figure 20: F1 score in function of lambda for 8 features( non-lineair), activity walking

### Variance and bias

To see which of the two problems we have, we can visualize the cost of the training and validation set in function of the degree of polynomial like in chapter 2.3. For lambda we used the value that gave the best F1 score (0.01987).

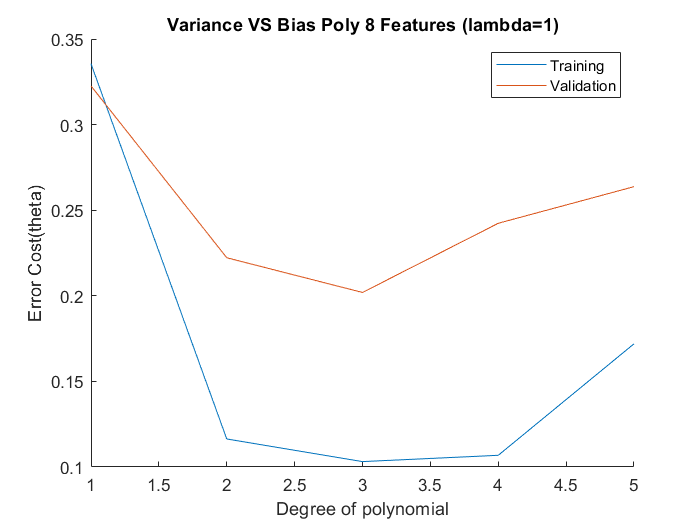


figure 21: Cost in function of the degree of polynomial

You can see that for degrees of polynomials under 2 we have a bias problem (underfitting) and from there on we keep getting a lower cost. Because we make use of regularization, it will reduce variance considerably.

### Adding more training examples

By plotting the number of training examples, an ideal number of training examples can be selected for a value where a high F1 score is found. The plot shows using few training examples can be problematic resulting in a low F1 score. At around 3000 training examples, which was very close to our case, the F1 score seems to have the best values.

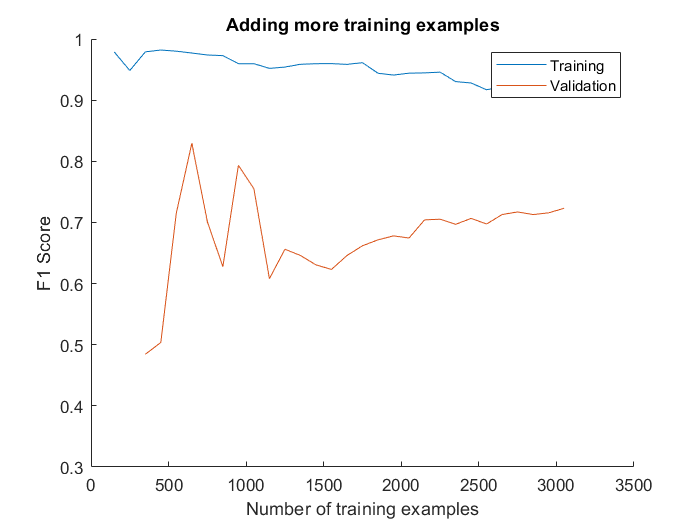


figure 22: Number of training examples in function of F1 score

# Conclusion

The report talked about different parameters to be optimized to improve the success rate of the model and demonstrates this using different examples and optimizations. Depending on the data you are working with, in this case the selected features, parameters or optimization may greatly vary. Some features only require few optimizations to none with simple implementation while others will have to be optimized several times with complicated implementations.

Choosing the right parameters with a good separation can play a major factor in the optimization process. The choice is also mostly responsible for which optimization techniques are optimal to be used.

## Appendix A

