

# Machine Learning:

Karsten Standnes - STNKAR012

October 2017

## 1 Introduction

In the world today there are many methods that fall under the category of "Machine Learning", some being very similar and some very different. All of them share in common that they use data to produce a model that can be used on unseen data. When successful this is very appealing in today's society where we got lots of data on situations where there is likely to be a underlying pattern, another requirement for learning. There is broad agreement that Machine Learning is a good way to make prediction and classification models, and often the only computational feasible way to do so. This makes it a task to decide which method in machine learning to chose for a given problem. The answer is not always the same and several methods can be good, but for different reasons. In this task I will show several different machine learning algorithms and how they perform on classifying pictures of handwritten numbers into even and odd numbers.

COMMENT on data snooping

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Methods</b>	<b>2</b>
2.1	Tree based methods . . . . .	2
2.1.1	Classification tree . . . . .	2
2.1.2	Random Forest . . . . .	2
2.2	Neural Networks . . . . .	2
2.2.1	Artificial? . . . . .	2
2.2.2	Convolutional Neural Network . . . . .	2
2.3	Support vector machines . . . . .	3
2.4	K-Nearest Neighbours . . . . .	3
<b>3</b>	<b>Results</b>	<b>3</b>
3.1	Tree based methods . . . . .	3
3.1.1	Classification tree . . . . .	3
3.1.2	Random Forest . . . . .	4
3.2	Neural Networks . . . . .	6
3.2.1	Artificial Neural Network . . . . .	6
3.3	Support vector machines . . . . .	7
3.4	K-Nearest Neighbors . . . . .	8
<b>4</b>	<b>Discussion</b>	<b>8</b>
<b>5</b>	<b>Acknowledgments</b>	<b>8</b>
<b>6</b>	<b>Appendices</b>	<b>8</b>

## 2 Theory

As mentioned in the different methods in Machine Learning will be used to try to build a model that manage to correctly classify if pictures of handwritten numbers are odd or even. This would be simple if the method should only classify previously seen data, the challenge arise when asked to make a model that is highly accurately in classifying new data. The goal is to make a model that trains without overfitting on the in-sample data in order to well estimate the out-of-sample data. In the methods below this is done by using different techniques like cross-validation, validation-set and regularization. In all of the methods the data is classified into the numbers 0 – 9. Another approach would be to classify the number's directly into either a odd or even category. The reason for using all ten numbers is mainly because this makes the extension of the use purpose a lot smoother, f.ex. to also classify if the number is a prime number or not.

### 2.1 Tree based methods

#### 2.1.1 Classification tree

Classification tree is maybe the method in Machine Learning which is easiest to interpret due to it's intuitive construction and clear visualization. The method uses a greedy approach using recursive binary splitting to structure a tree that can classify input based on it's variables. The goal of the classification tree is to classify a set of data as best as possible while have a low complexity to avoid overfitting. Below we see that one classification tree is not enough to make a great model for the problem, but combining many of them gives us a "Random Forest" which is discussed in subsection 2.1.2. Classification trees also gives a nice visual image of the classification.

#### 2.1.2 Random Forest

Random Forest is as mentioned (subsection 2.1.1) constructed of many classification tree. After such a construction by using a set of training data, new data can be ran through the "forest". The new data is classified with the label the majority of trees labelled it. The trees in other words "vote" for the best classification for the data. As in real life, voting makes little difference if all the votes are the same. To avoid making a forest out of  $n$  ( $n \in \mathbb{Z}_{>0}$ ) identical trees two aspects are changed when the constructing the classification trees. The first is that the data of size  $N$  used in training a tree is sampled from the whole set of  $N$  data **with** replacement. This will on average lead to about  $\frac{2}{3}$  data to be used in creating each tree leaving on average  $\frac{1}{3}$  Out-Of-Bag(OOB) which is used for validation. The OOB-data is critical in making insuring the Random Forest doesn't overfit. The other is that  $m$  randomly selected variables are used for each tree, where  $m \ll M$  and  $M$  is the whole set of variables in the problem. This gives a rich variety of trees where with enough trees a predictive model can be created.

#### 2.1.3 Bagging

Bagging is basically a Random Forest (see 2.1.2) where the number of variables considered for each split is the whole variable set for the problem. It shares the same behaviour as a Random Forest when sampling out the data and in classifying new data. The big difference is that the variables are not sampled in Bagging, this does that Bagging creates trees that are more correlated than a Random Forest making it more susceptible to dominating features. This will not directly lead to a bad classification model, but it does have the unwanted consequence of making semi-important features of the data to be underestimated in the contribution towards new predictions.

### 2.2 Neural Networks

#### 2.2.1 Artificial Neural Network

Artificial Neural Networks got the name because it to some degree mimic the behaviour of neurons in the brain. It does so by connecting nodes (neurons) together with weights (synapses) connecting

them. In the nodes that are not input nodes there is an activation function altering the values that are brought to the nodes through weights from other nodes. The activation is one of several factors that can be tuned when creating a neural network. Other factors are the size of the hidden layers and the amount of nodes in each layer of the network. Neural networks are also exposed to overfitting, several techniques can be used to decrease the chance of overfitting. Some of these are validation, using dropout and regularization.

### **2.2.2 Convolutional Neural Network**

For Neural Networks I have decided to use a convolutional neural network because of its great performance in problems that can be represented as an "image". This is because convolutional neural networks (CNN's) work by recognizing patterns in images or represented as a matrix or tensor (multidimensional matrix) in the computer. It does so by using convolutional, pooling and voting layers. Multiple of these can be stacked to create a very precise model for recognising images.

## **2.3 Support vector machines**

### **2.4 K-Nearest Neighbours**

K-Nearest Neighbours (KNN) is a easy-to interpret and maybe one of the simplest methods in Machine Learning. The method does exactly as the name suggest; find the  $k$  nearest neighbours, where  $k$  is a number and labels each data point based on those neighbours. The method is attractive because of its easy interpretability, fast execution time and often precise classification and regression power. There are many variants of the method where the distance measure and way to handle draws differ. KNN's with  $k = 1$  have a special property that the hyperplane of the data points will be partitioned into an equal amount of partitions as there are data points. Each partition consist of all point in the plane closer a specific data point than any other. The amount of neighbours  $k$  can have a great impact on the accuracy of the classification. In some cases a low  $k$  will be preferable with very separated data, in other case it can lead to overfitting due to higher influence from noise and outliers. Because a even number of categories is classified in this problem the concerns of draws is present when using a even  $k$ . This makes a odd  $k$  more attractive in this problem.

## 3 Methods

### 3.1 Tree based methods

#### 3.1.1 Classification tree

#### 3.1.2 Random Forest

#### 3.1.3 Bagging

### 3.2 Neural Networks

#### 3.2.1 Artificial Neural Network

#### 3.2.2 Convolutional Neural Network

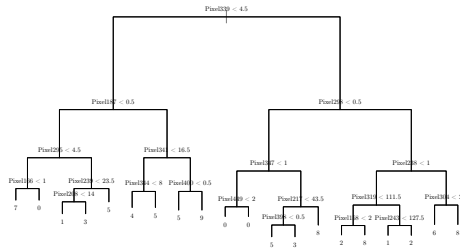
### 3.3 Support vector machines

### 3.4 K-Nearest Neighbours

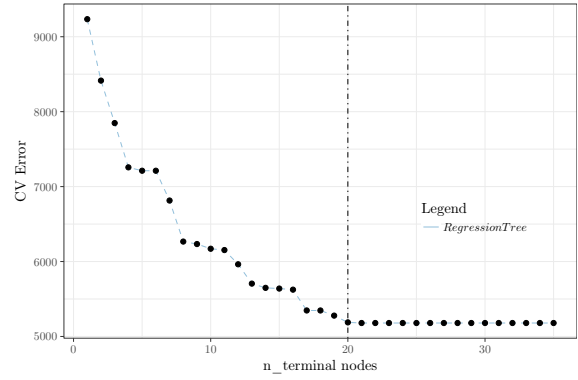
## 4 Results

### 4.1 Tree based methods

#### 4.1.1 Classification tree



(a)  $E_{CV}$  for different amounts of leaf nodes.



(b) Regression tree after pruning

Figure 1: A figure with two subfigures

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	31	0	5	1	1	3	4	1	0	1	0.340	16/47
1	0	44	1	2	0	2	3	0	1	0	0.170	9/53
2	0	3	32	2	1	0	0	0	2	0	0.200	8/40
3	0	1	1	26	1	0	1	1	0	3	0.235	8/34
4	0	0	0	0	26	0	10	0	1	1	0.316	12/38
5	4	2	2	12	9	31	6	10	3	7	0.640	55/86
6	0	1	3	1	2	1	23	0	0	2	0.303	10/33
7	1	0	1	2	0	0	2	34	0	5	0.244	11/45
8	0	2	5	3	6	5	6	0	39	3	0.435	30/69
9	3	0	2	2	7	3	2	1	6	27	0.491	26/53
Total	8	9	20	25	27	14	34	13	13	22	0.371	185/498

#### 4.1.2 Random Forest

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	46	0	0	1	0	0	0	0	0	0	0.021	1/47
1	0	42	0	0	0	1	0	1	0	0	0.045	2/44
2	0	2	48	0	0	0	0	1	0	0	0.059	3/51
3	0	0	0	41	0	1	0	0	3	0	0.089	4/45
4	0	1	0	0	52	1	1	4	0	1	0.133	8/60
5	0	0	0	4	0	42	1	0	1	0	0.125	6/48
6	0	0	0	0	1	0	47	0	2	0	0.060	3/50
7	0	1	0	0	0	0	0	48	0	2	0.059	3/51
8	1	2	0	1	0	0	0	0	42	1	0.106	5/47
9	0	0	0	1	2	1	0	0	1	48	0.094	5/53
Total	1	6	0	7	3	4	2	6	7	4	0.081	40/496

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	47	0	0	2	0	3	0	0	0	0	0.096	5/52
1	0	42	0	0	0	0	0	0	0	0	0.000	0/42
2	0	2	46	0	0	0	0	1	0	0	0.061	3/49
3	0	1	0	41	0	2	0	1	2	0	0.128	6/47
4	0	1	0	0	49	1	1	2	0	2	0.125	7/56
5	0	0	0	3	0	38	1	0	1	0	0.116	5/43
6	0	0	1	0	1	1	47	0	2	1	0.113	6/53
7	0	0	0	0	0	0	0	47	0	3	0.060	3/50
8	0	2	1	1	1	0	0	2	43	1	0.157	8/51
9	0	0	0	1	4	1	0	1	1	45	0.151	8/53
Total	0	6	2	7	6	8	2	7	6	7	0.103	51/496

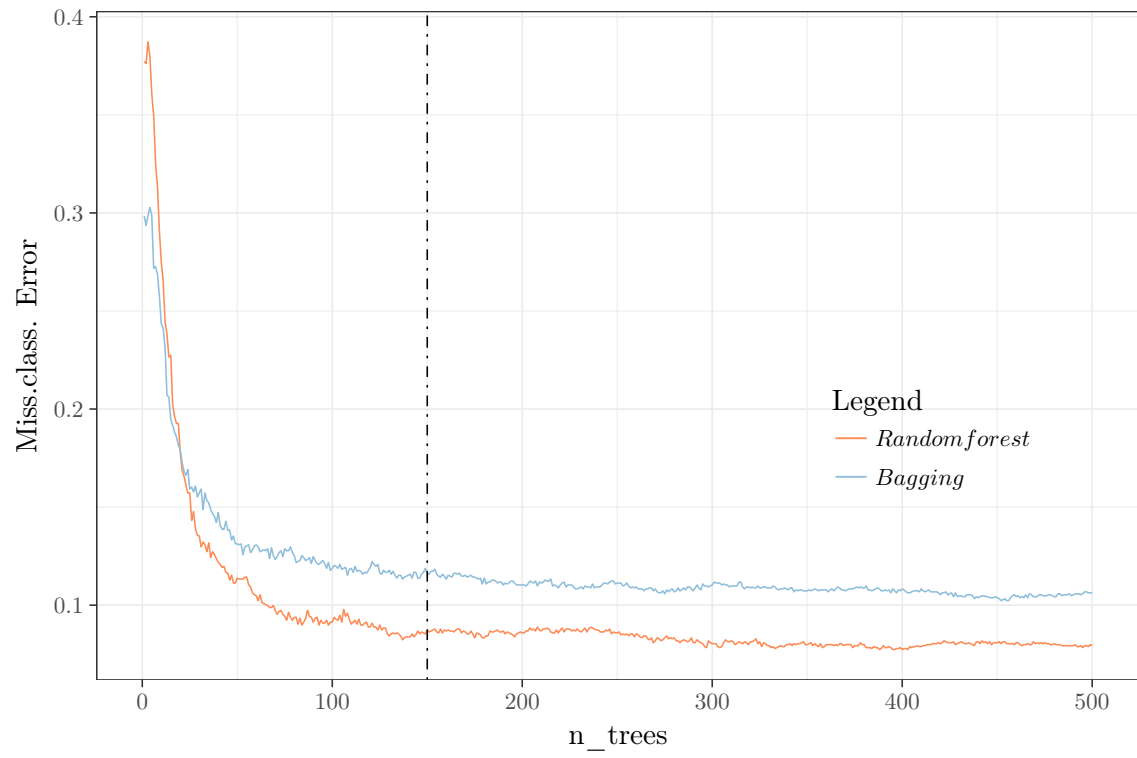


Figure 2

## 4.2 Neural Networks

### 4.2.1 Artificial Neural Network

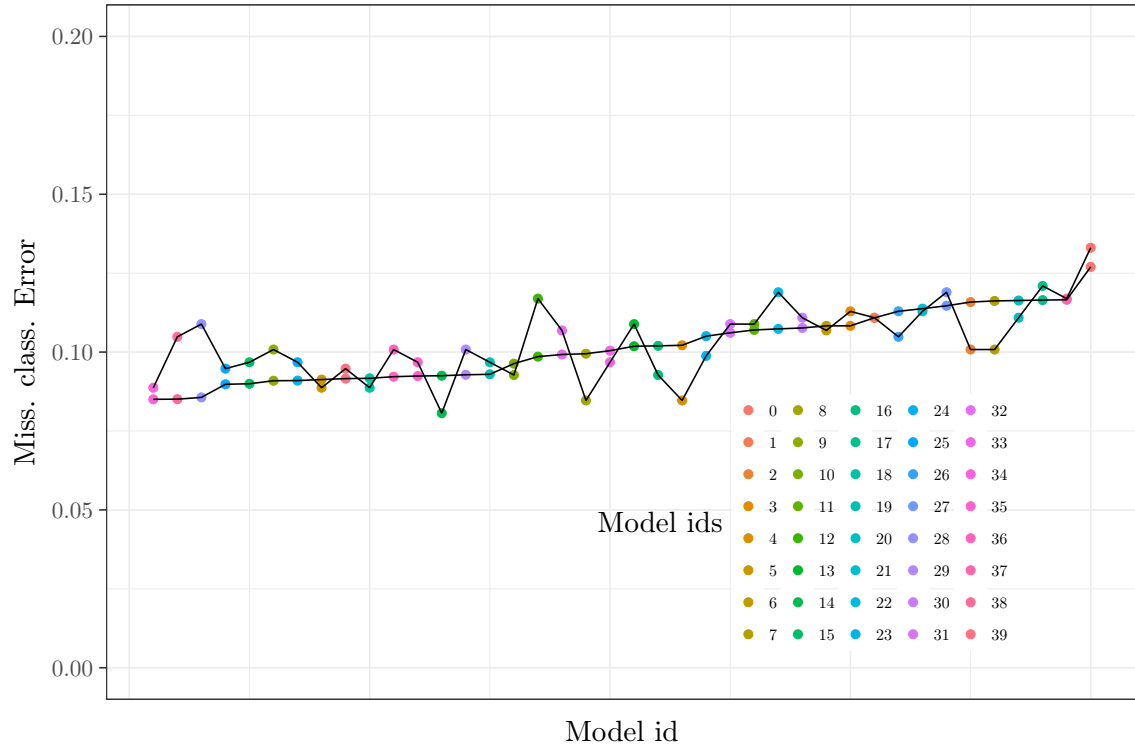
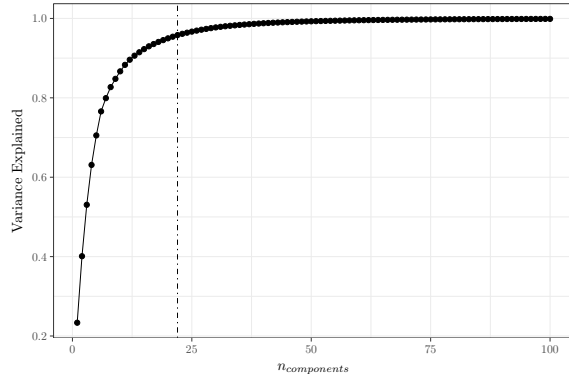


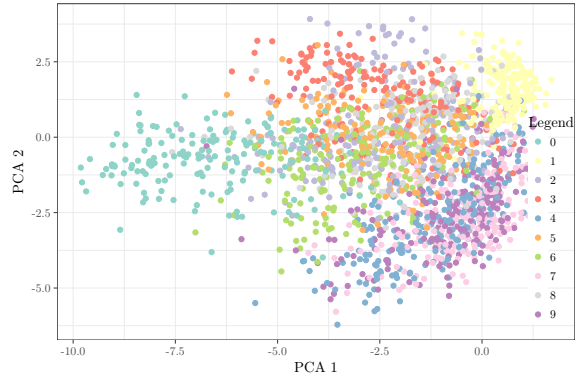
Figure 3

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	42	0	0	0	0	2	0	0	0	0	0.045	2/44
1	0	47	0	0	1	0	0	0	4	0	0.096	5/52
2	2	1	46	2	0	0	1	0	0	0	0.115	6/52
3	0	0	0	42	1	1	0	1	0	0	0.067	3/45
4	0	0	0	0	50	0	0	2	0	2	0.074	4/54
5	1	0	0	2	1	43	0	1	6	1	0.218	12/55
6	1	0	0	1	1	0	48	0	0	0	0.059	3/51
7	1	0	2	0	1	0	0	49	1	3	0.140	8/57
8	0	0	0	1	0	0	0	0	38	0	0.026	1/39
9	0	0	0	0	0	0	0	1	0	46	0.021	1/47
Total	5	1	2	4	5	3	1	5	11	6	0.091	45/496

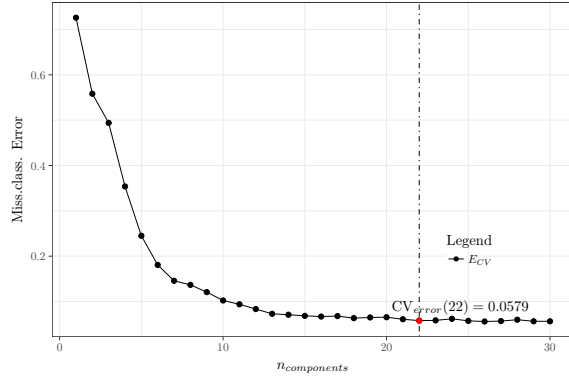
### 4.3 Support vector machines



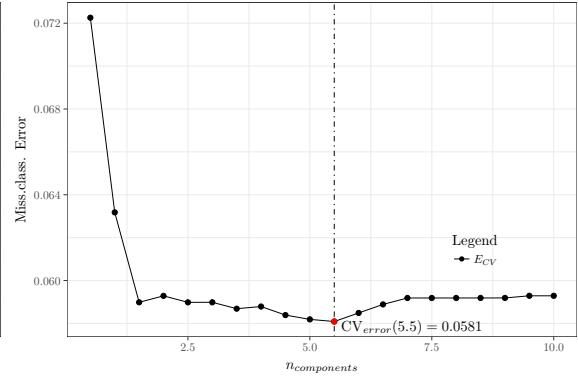
(a)



(b)



(a)



(b)

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	47	0	0	0	0	0	0	0	0	0	0.000	0/47
1	0	48	0	0	0	0	1	0	0	0	0.020	1/49
2	0	0	46	0	0	0	0	0	1	1	0.042	2/48
3	0	0	1	41	0	0	0	1	2	0	0.089	4/45
4	0	0	0	0	53	0	0	2	0	1	0.054	3/56
5	0	0	0	4	0	46	0	0	0	0	0.080	4/50
6	0	0	0	0	1	0	48	0	1	0	0.040	2/50
7	0	0	0	0	0	0	0	49	0	4	0.075	4/53
8	0	0	1	1	0	0	0	1	45	1	0.082	4/49
9	0	0	0	2	1	0	0	1	0	45	0.082	4/49
Total	0	0	2	7	2	0	1	5	4	7	0.056	28/496



## 4.4 K-Nearest Neighbors

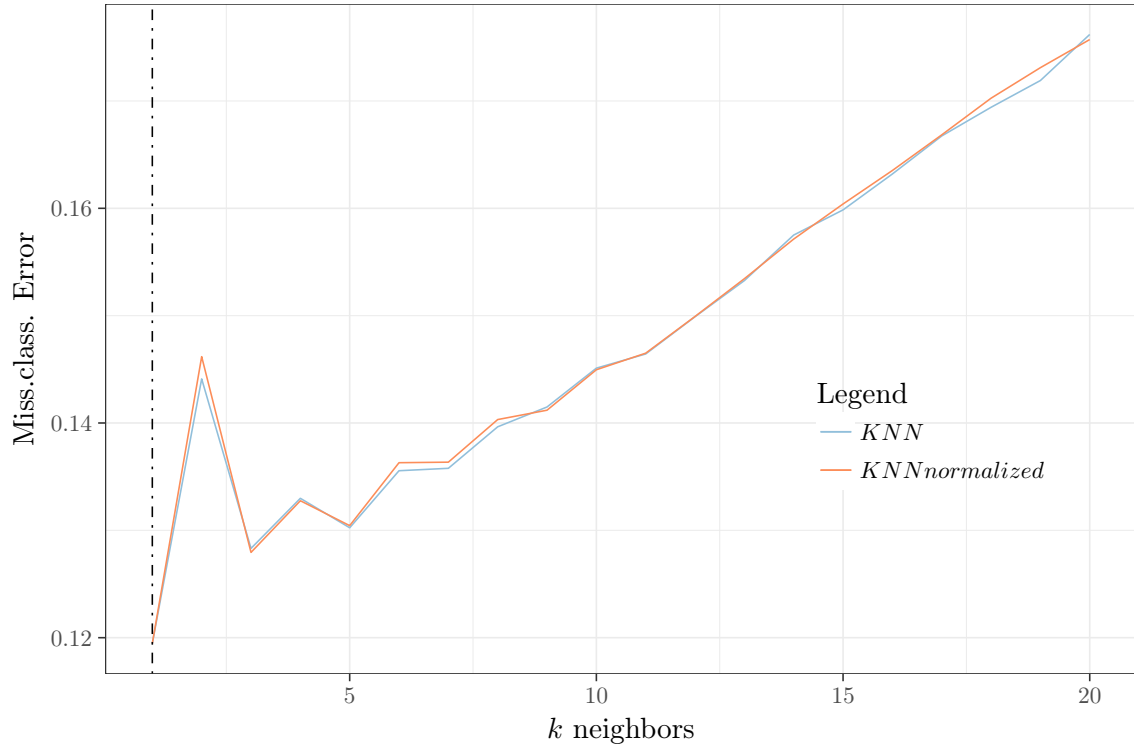


Figure 6

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	45	0	0	1	0	1	0	0	1	0	0.062	3/48
1	0	46	3	0	0	2	0	2	1	0	0.148	8/54
2	0	0	45	0	0	0	0	2	1	0	0.062	3/48
3	0	0	0	40	0	1	0	0	6	0	0.149	7/47
4	0	1	0	0	52	0	0	3	1	4	0.148	9/61
5	1	0	0	4	0	41	1	0	1	0	0.146	7/48
6	1	0	0	0	1	0	48	0	0	0	0.040	2/50
7	0	1	0	0	0	0	0	45	0	1	0.043	2/47
8	0	0	0	3	0	0	0	1	37	0	0.098	4/41
9	0	0	0	0	2	1	0	1	1	47	0.096	5/52
Total	2	2	3	8	3	5	1	9	12	5	0.101	50/496

## 5 Discussion

## 6 Acknowledgments

## 7 Appendices