

# **Detection of human alertness using supervised learning**

Anders Hørsted

Kongens Lyngby 2011  
IMM

Technical University of Denmark  
Informatics and Mathematical Modelling  
Building 321, DK-2800 Kongens Lyngby, Denmark  
Phone +45 45253351, Fax +45 45882673  
[reception@imm.dtu.dk](mailto:reception@imm.dtu.dk)  
[www.imm.dtu.dk](http://www.imm.dtu.dk)

# Summary

---

This is the summary/abstract



# Resumé

---

På dansk



# Preface

---

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in engineering.

The thesis deals with different aspects of mathematical modeling of systems using data and partial knowledge about the structure of the systems. The main focus is on extensions of non-parametric methods, but also stochastic differential equations and neural networks are considered.

The thesis consists of a summary report and a collection of ten research papers written during the period 1996–1999, and elsewhere published.

Lyngby, December 1999

Henrik Aalborg Nielsen





# Acknowledgements

---

I thank my...



# Contents

---

<b>I</b>	<b>Introduction and data exploration</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	The competition . . . . .	4
1.1.1	Problemformulering . . . . .	4
<b>2</b>	<b>The Ford Challenge</b>	<b>5</b>
2.1	Other online machine learning competitions . . . . .	5
2.1.1	Netflix Prize . . . . .	6
2.1.2	KDD Cup . . . . .	6
2.1.3	And many others . . . . .	7
2.2	Kaggle.com and The Ford Challenge . . . . .	7
2.2.1	The data set . . . . .	9
<b>3</b>	<b>Data exploration</b>	<b>11</b>
3.1	A note about Test data, training data . . . . .	11
3.2	Calculating common statistics . . . . .	11
3.3	Determining the datatype of features . . . . .	12
3.3.1	Unique values . . . . .	12
3.3.2	Plotting some features . . . . .	16
3.4	Outlier detection . . . . .	16
3.4.1	Making boxplots of features . . . . .	18
3.4.2	Making layered feature plots . . . . .	20
3.4.3	Features that are constant in a trial . . . . .	21
3.5	Finding possible discriminating features . . . . .	22
3.5.1	Testing binary features . . . . .	22

3.5.2	Scatterplots . . . . .	22
3.6	Principal Component Analysis . . . . .	23
3.6.1	Theory . . . . .	23
3.7	Conclusions . . . . .	24
<b>II</b>	<b>Modelling</b>	<b>25</b>
<b>4</b>	<b>Theory of classification</b>	<b>27</b>
4.1	The binary classification problem . . . . .	27
4.2	Parametric models and maximum likelihood . . . . .	29
4.2.1	Maximum likelihood . . . . .	29
4.3	Logistic Regression . . . . .	30
4.3.1	The decision boundary . . . . .	32
4.3.2	Model complexity . . . . .	32
4.3.3	Regularization . . . . .	33
4.4	Feed Forward Neural Network . . . . .	33
4.4.1	Model complexity . . . . .	33
4.4.2	Regularization . . . . .	33
4.5	Measuring classifier performance . . . . .	33
4.5.1	Splitting the dataset . . . . .	34
4.5.2	Cross validation . . . . .	36
4.5.3	AUC . . . . .	36
4.6	Model selection . . . . .	36
4.6.1	Over- and underfitting . . . . .	36
4.6.2	Forward feature selection . . . . .	36
4.6.3	Regularization . . . . .	36
<b>5</b>	<b>Recreating winning approach</b>	<b>37</b>
<b>6</b>	<b>Improving the winning approach</b>	<b>39</b>
<b>7</b>	<b>Other classification methods</b>	<b>41</b>
<b>III</b>	<b>Workflow and discussion</b>	<b>43</b>
<b>8</b>	<b>Workflow and tools</b>	<b>45</b>
<b>9</b>	<b>Discussion</b>	<b>47</b>

10 Conclusion	49
A Appendices	51
A.1 Calculating common statistics - Code . . . . .	52
A.2 Calculating common statistics - Result . . . . .	53
A.3 Calculate unique values of features - Code . . . . .	54
A.4 Calculate unique values of features - Result . . . . .	56
A.5 Creating boxplots - Code . . . . .	56
A.6 Creating boxplots - Result . . . . .	57
A.7 Creating layered featuer plots - Code . . . . .	57
A.8 Creating layered feature plots - Result . . . . .	57
A.9 Scatterplots - Code . . . . .	57
A.10 Scatterplots - Result . . . . .	58



## **Part I**

# **Introduction and data exploration**





## CHAPTER 1

# Introduction

---

Igennem de sidste 30 år er computerens ydeevne vokset markant. Den forbedrede ydeevne har givet mulighed for at udføre statistisk dataanalyse, i et omfang der ikke tidligere har været muligt. En af de ting der er blevet mulighed for, er at programmere såkaldte "classifiers". Et godt eksempel på en classifier, er de programmer bankerne bruger til at opdage evt. snyd med kreditkort. Baseret på alle tidligere korttransaktioner, og viden om hvilke der var "falske" transaktioner, kan bankerne programmere en classifier, der med høj præcision kan forudsige om en ny transaktion er snyd eller ej.

Denne opgave tager udgangspunkt i en konkurrence på hjemmesiden [kaggle.com](https://www.kaggle.com). Konkurrencen er udbudt af Ford Motors og handler om at udvikle en classifier, der kan forudsige om en bilist er ved at blive ukoncentreret, mens han/hun kører bil. Til at udvikle denne classifier har Ford foretaget en række målinger på bilister, mens de kørte bil, og til hver måling er det blevet noteret om bilisten var opmærksom eller ej. Med udgangspunkt i disse målinger udarbejdes – ved brug af de mest gængse metoder – en række classifiers, og deres evne til at klassificere ny data sammenlignes.

## 1.1 The competition

### 1.1.1 Problemformulering

I denne opgave vil jeg...

- ... bruge de mest gængse klassifikationsmodeller (nearest neighbour, logistic regression, neural networks og SVM) til at lave en classifier der (forhåbentlig) kan forudsige om en bilist er ved at falde i søvn.
- ... undersøge hvor stor indflydelse den indledende databehandling (feature selection og outlier removal) har på det endelige resultat.
- ... undersøge om classifieren kan forbedres ved at implementere en Hidden Markov Model, der tager hensyn til det temporale aspekt af data.
- ... implementere en ensemble classifier, der kombinerer resultatet af flere classifiers i én classifier.

## CHAPTER 2

# The Ford Challenge

---

In this chapter, the ford competition that is the basis for this report, is introduced. Before introducing the ford challenge, a short overview of other online machine learning competitions is given. As part of introducing the ford competition, the kaggle.com website that hosted the ford competition is presented, and the data set used in the ford competition is described in detail. But as mentioned the chapter starts with a short overview of other online machine learning competitions.

### **2.1 Other online machine learning competitions**

To get a little perspective on the ford challenge, this section gives a short review of some past and present online machine learning competitions.

### 2.1.1 Netflix Prize

One of the most talked about competitions, may very well be the Netflix Prize. The Netflix competition was launched on October 2, 2006 and the aim of the competition was to predict how users would grade new movies, based on a large dataset of previous grades. Why did the Netflix Prize gather a lot of attention? First of all the grand prize was \$1M, which is a lot of money. And secondly the dataset was huge, consisting of 100,480,507 ratings given by 480,189 users, leaving room for a lot of interesting new techniques to be tested.

When the Netflix Prize was awarded on September 18, 2009, 5169 different teams had submitted at least one entry to the competition. The winning team consisted of three different teams, that at one point decided to team-up and compete as a join-team.

Netflix originally wished to follow up the Netflix Prize, with another competition but decided to dismiss the idea, due to a lawsuit regarding privacy concerns related to the first Netflix Prize. \*

### 2.1.2 KDD Cup

Although the Netflix Prize gather a lot attention, it wasn't the first online machine learning competition. An example of an earlier competition, is the KDD Cup. The KDD Cup is a competition that is held every year, and that started back in 1997. The subject changes every year, and can be anything from mining purchase data from an online store, to computer aided detection of breast cancer (the 2000 and 2008 competitions respectively).

This year the KDD Cup is held in cooperation with Yahoo! Labs and the task is to predict user ratings of musical items (both tracks, albums, artists and genres). One note worthy detail about the 2011 KDD Cup is the huge data set, containing over 300 million ratings of more than 600,000 distinct items. †

---

\* The section about The Netflix Prize is based on [Wikipedia \(2011\)](#) and [Netflix \(2011\)](#)

† Read more about the KDD Cup history at [ACM \(2011\)](#). For more info about the 2011

### 2.1.3 And many others

The Netflix Prize and the KDD Cup are just two examples of online machine learning competitions. Many others exist, such as

- *The Hearst Challenge 2011* - Every year The Hearst Corporation hosts a machine learning competition. This year the task is to data mine the history of 1.8 million emails sent to subscribers of Hearst's publications, and then predict who will open emails in the future. Read more at <http://www.hearstchallenge.com>
- *The Reclab Prize* - RichRelevance is a company that specializes in online product recommendation. They offer a \$1M prize for the team that first improves their product recommendation algorithm by 10%. Read more at <http://www.overstockreclabprize.com>
- *The Heritage Health Prize* -

Since almost all the machine learning competition websites, need the same functionality, websites that specialize in hosting machine learning competitions have appeared. One example of a website that provides a hosting platform for machine learning competitions is the kaggle.com website, who hosts The Ford Challenge.

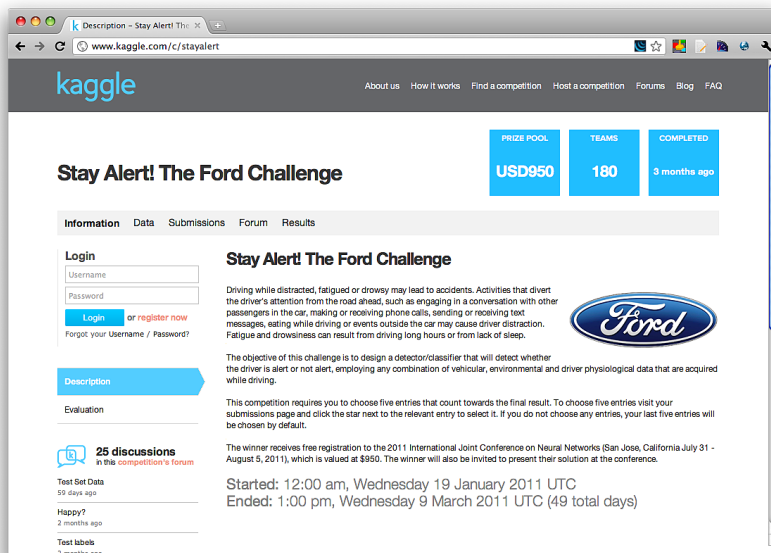
## 2.2 Kaggle.com and The Ford Challenge

**TODO: Mention that use of physical features should be minimized?**

As already mentioned, kaggle.com hosts machine learning competitions for universities and corporations. The first competition hosted by kaggle.com was started in April 2010, and since then a total of 18 competitions have been held. Every competition at kaggle.com has some background information, links to the data sets, a submission system and a forum, as seen in figure 2.1.

---

competition see [Labs \(2011\)](#)



**Figure 2.1:** The Ford Challenge frontpage at the kaggle.com website

The Ford Challenge began on January 19, 2011. The task was to create a classifier that is able to detect when a driver is about to get distracted while driving. The dataset that Ford made available for the competition consisted of measurements of 30 different features, measured on drivers along with a binary feature (IsAlert) that was 1 if the driver was alert and 0 otherwise. The 30 features was a mix of environmental, driver physiological and vehicular features<sup>‡</sup>. Based on this dataset a classifier should predict the IsAlert feature of a distinct test dataset held by Ford.

One detail that made the competition slightly different than many other competitions, was that Ford would not disclose any information about what the different features represented<sup>§</sup>. The official reason was that (Abou-Nasr, 2011c)

“We like to encourage the participants to pursue classification without preconceived notions based on prior knowledge of the

<sup>‡</sup>The dataset is explained in much more detail in the section 2.2.1

<sup>§</sup>see forum replies from the Ford spokesperson (Abou-Nasr, 2011b,a)

subject, focusing on variables which lead them (based on their own experiments) to better classification."

Doubts about the true motive behind the lack of details about the features, was expressed by, what later turned out to be, the winner of the competition ([Inference, 2011](#))

The performance of the classifiers was measured by calculating the AUC (see section 4.5.3) of the classifiers, on the test set. A limit of two submissions per contestant was set as a way to counteract the possibility of someone reverse-engineering the IsAlert-feature of the test dataset. This could of course be circumvented by one person registering more than once. And this was exactly what happened. On March 9, 2011 when the competition ended, two users had achieved exactly the same AUC (six significant digits) and other contestants immediately questioned the probability of two unrelated users getting exactly the same AUC ([Pardos, 2011](#)). One of the two leaders (Rosanne/Shen) quickly admitted that he and his friend had indeed used two accounts to get 4 submission per day ([Rosanne, 2011](#)), and after some discussion in the forum, the leaders were disqualified. The end result was that the user Inference in third position was declared the official winner of the competition.

Two weeks after the competition ended, Inference described the technique used to win the competition. This will be described in detail in chapter 5. But before that chapter, the data set will be described in details in the next section.

### 2.2.1 The data set

The dataset used to create a classifier was released on the competition website, the day the competition started. The dataset consisted of a number of trials and each trial was approximately 2 minutes of sequential data recorded every 100ms during a driving session on the road or in a driving simulator ([Kaggle.com, 2011](#)). As the interval between two rows was 100ms, each trial consists of approximately  $2\text{minutes} \cdot 60 \frac{\text{secs}}{\text{minute}} \cdot 10 \frac{\text{rows}}{\text{sec}} = 1200$  rows.

TrialID	ObsNum	IsAlert	P1	...	P8	E1	...	E11	V1	...	V11
0	0	0	12.2	...	1.2	4.3	...	33	12	...	7.34
⋮											⋮
0	1200	1	11.1	...	10.7	1.3	...	21	8	...	8.82
⋮											⋮
510	1198	0	11.1	...	10.7	1.3	...	21	8	...	8.82

**Table 2.1:** Structure of the data set

Each row has a total of 33 data columns structured as shown in table 2.1. Some details are worth noticing:

- The TrialID starts at 0 and the trials from 469 to 479 (both inclusive) are missing. The last trial has TrialID=510. This gives a total of exactly 500 trials.
- The ObsNum also starts at 0 there are not exactly 1200 observation for every trial.
- The total number of rows is 604,229
- The row number is not part of the data set, so a row is uniquely identified by the pair (TrialID, ObsNum).

As mentioned before no additional information about what the different features represent or what datatype (discrete, continous) they are, was disclosed by Ford. The only way to get these informations is by doing a thorough data exploration of the data set and that is what the next chapter is about.



## CHAPTER 3

# Data exploration

---

Here I describe the various data exploration techniques I have used. Lots of nice graphs.

### 3.1 A note about Test data, training data

### 3.2 Calculating common statistics

To start the data exploration, four common statistics, namely the mean, min, max and standard deviation, of every feature across the whole dataset was calculated. The standard deviation was calculated as (with  $n$  equal the total number of rows in the dataset, and  $x_i$  equal to the  $i$ 'th value of any of the features)

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$$

The source code for the calculations can be found in appendix [A.1](#) and all results in appendix [A.2](#). Most results did not tell that much, but a few results stood out. These are shown in table [3.1](#). The table shows that the features P8, V7 and V9, are zero throughout the whole dataset, and can be ignored. The features E9 and V5, could be binary, but that is only speculation at the moment. Finally the feature P6 has a mean of 843.73 and a standard deviation of 2795.32, but its maximum is 228812.00, which is many, many standard deviations away from the mean. This could be a sign of some serious outliers, but it could as well be a single trial with a mean far from the other trials. Further investigation are needed to conclude anything here. Finally it is seen that the mean of the IsAlert feature is only a little above 0.5, and therefore only a little over 50% of the time are the drivers alert. This may be a bit surprising.

The results of this first step have been to exclude a few features, and get some rough ideas about the shape of some other features. It is now time to resally get to know the different features.

### 3.3 Determining the datatype of features

Since Ford would not disclose any information about the different features, it is important to get a good picture of which features are discrete/-categorical and which features are continous. A natural first step to learn the datatype of the features, is to calculate the number of unique values each feature takes.

#### 3.3.1 Unique values

It requires a little thought to pinpoint exactly what needs to be calculated. On one hand it is natural to calculate the number of unique values a feature takes across the whole dataset. On the other hand it could happen, that a discrete feature might have only (eg.) 3 unique value within any trial, but the values differ between various trials. This way we would have a feature with 1500 unique values across the whole dataset, but

Feature	Min in trial	Max in trial	Whole dataset
E1	1	131	12265
E10	1	86	121
E11	1	79	116
E2	1	129	28502
E3	1	3	3
E4	1	97	249
E5	1	134	254
E6	1	116	247
E7	1	15	25
E8	1	10	10
E9	1	2	2
IsAlert	1	2	2
P1	967	1208	105715
P2	1155	1207	128666
P3	24	106	406
P4	24	106	406
P5	34	114	1070
P6	3	99	406
P7	3	99	406
P8	1	1	1
V1	1	969	12374
V10	1	5	5
V11	875	1206	166455
V2	1	76	90
V3	1	29	33
V4	1	204	324
V5	1	2	2
V6	1	775	2727
V7	1	1	1
V8	1	239	323
V9	1	1	1

**Table 3.1:** Highlights from the results of the summary statistics. See appendix A.2 for all results.

with max 3 unique values within any given trial. Therefore the number of unique values within a trial are calculated for every feature and every trial. Based on this result, the minimum and maximum number of unique values within a single trial, is calculated for every feature. The source code can be found in appendix A.3 and results are shown in table 3.2. **TODO: The result table in appendix or in report?**

A couple of things should be noticed about the results. Almost all features have some trials where they only take on one unique value. For categorical features, this could be perfectly normal, but for a continuous feature it seems to be pretty unlikely to have only one value in a trial spanning two minutes. Is a feature, that in some trials seems continuous, but then only have one value in other trials, just turned off in the latter trials? And how should the feature be handled in trials where it is “off”? This discussion is continued in section 3.4 about outlier detection.

From table 3.2 it is also seen, that only features P1, P2 and V11 is consistently having lots of unique value, across all trials. It seems fair to call these features continuous. Feature V1 have some trials where it takes on 969 different values, but it also have trials with only one unique value. Across the whole dataset it takes on 12374 unique values. This could be explained by feature V1 being continuous in a small number of trials (about 15-30), and turned “off” in all other trials. Further exploration will show whether it is true or not.

Another interesting detail is that there are two pairs of features ((P3,P4), (P6,P7)) in the results, that share exactly the same number of unique features. Both min and max and total. This indicates a possible relationship between the features, and later (section 3.5.2) it is shown that this is in fact true.

A final remark about the number of unique values, is that feature E9 and V5 indeed are binary. Also if a limit of max 40 unique values within a single trial is set, as an indicator for categorical features, E3, E7, E8, V3 and V10 are seen to be categorical.

Feature	Min in trial	Max in trial	Whole dataset
E1	1	131	12265
E10	1	86	121
E11	1	79	116
E2	1	129	28502
E3	1	3	3
E4	1	97	249
E5	1	134	254
E6	1	116	247
E7	1	15	25
E8	1	10	10
E9	1	2	2
IsAlert	1	2	2
P1	967	1208	105715
P2	1155	1207	128666
P3	24	106	406
P4	24	106	406
P5	34	114	1070
P6	3	99	406
P7	3	99	406
P8	1	1	1
V1	1	969	12374
V10	1	5	5
V11	875	1206	166455
V2	1	76	90
V3	1	29	33
V4	1	204	324
V5	1	2	2
V6	1	775	2727
V7	1	1	1
V8	1	239	323
V9	1	1	1

**Table 3.2:** The minimum and maximum number of unique values within the trials, for every feature in the dataset. Also the total number of unique values for each feature, across the whole dataset, are shown.

### 3.3.2 Plotting some features

To get a more nuanced picture of the various features, it is now time to visualize the data. The first thing of interest is to plot every feature for a couple of trials, to see the structure of the features, and how much they vary between different trials. For a start some random plotting was done in the interactive ipython-shell\*.

Some selected plots from the interactive plotting session are shown in figure 3.1. The main results from the introductory plotting of features are that:

- A feature varies a lot in structure between different trials
- Many features are constant in some trials (was already hinted at in section 3.3.1)
- Some features deviate a lot from their “normal” structure, in a few trials (more about that in section 3.4 about outliers).
- The feature P2 looks like pure noise **TODO: Is this right?**
- The features E3, E7, E8, V3 and V10 do indeed seem to be categorical features, although V3 does have some extra structure **TODO: What do I mean? Plot categorical features?**

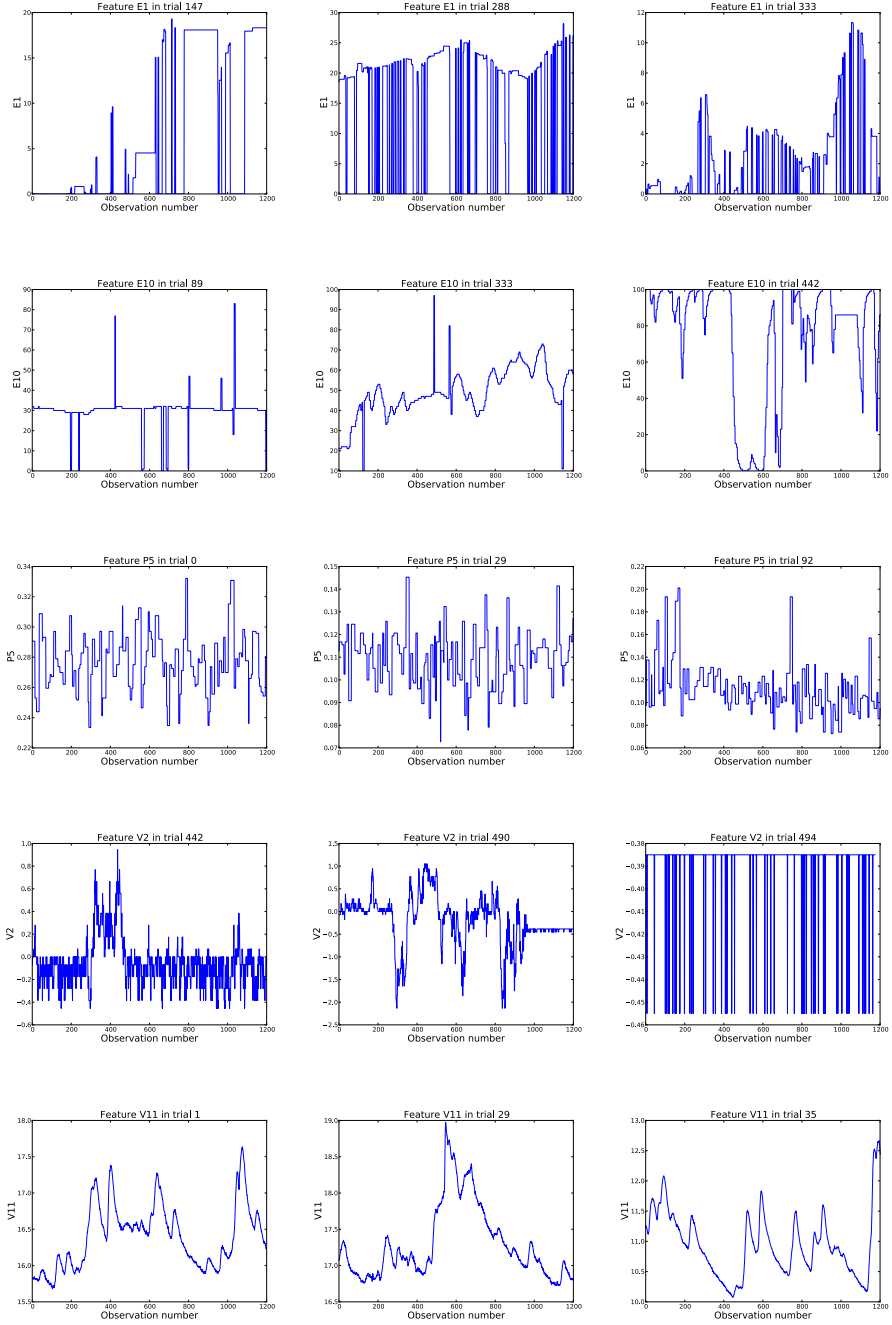
Apart from the observations above, a subjective conclusion from the plots of the features is that the data quality could be better. Since the data quality isn't perfect it is important to consider how outliers should be dealt with. This is what the next section is about.

## 3.4 Outlier detection

The textbook approach for detecting outliers is to create boxplots of the different features, and then argue that data points outside the 95-percentile

---

\* For details about the software setup used, see chapter 8



**Figure 3.1:** Plots of the features E1, E10, P5, V2, V11, for a couple of trials. The trials was selected to show the diversity a single feature exhibits between trials.

should be removed. Although this is a naive approach some useful information could be obtained from boxplots of the features. But it doesn't necessarily make sense to make boxplots of the features for the whole dataset. Some features have the same structure in different trials, but their mean varies. Eg. feature V11 in trials 29 and 35 have similar structure but the range in trial 29 is [16.75, 19] and [10, 12.6] in trial 35 (see figure 3.1). If a feature has a different mean in only a few trials, those datapoints could be seen as outliers, if we look at the feature across the whole dataset. Instead boxplots of the features for the 50 first trials are created. This also gives a visual idea about how feature means varies from trial to trial.

### 3.4.1 Making boxplots of features

The boxplots for 4 selected features are shown in figure 3.2. All boxplots are in appendix A.6, and the source code can be seen in appendix A.5.

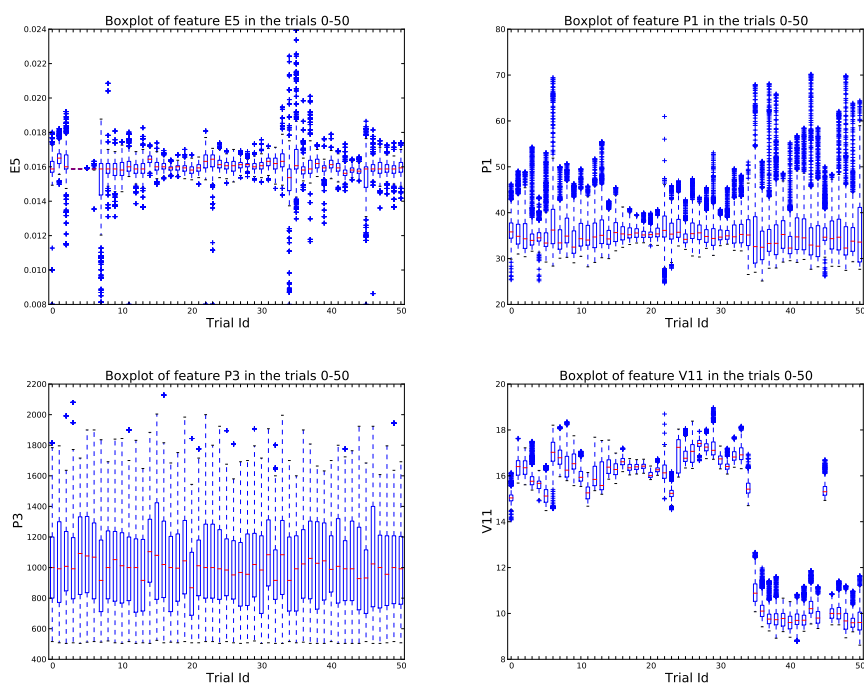
From the boxplots a couple of things are of interest.

- For some features the trial mean varies a lot between different trials  
**TODO: Remember to check with boxplots in appendix**
- Within a single trial some features still have a lot of data points outside the 95 percentile. See eg. P1 in figure 3.2 **TODO: Check the right name (percentile?)**

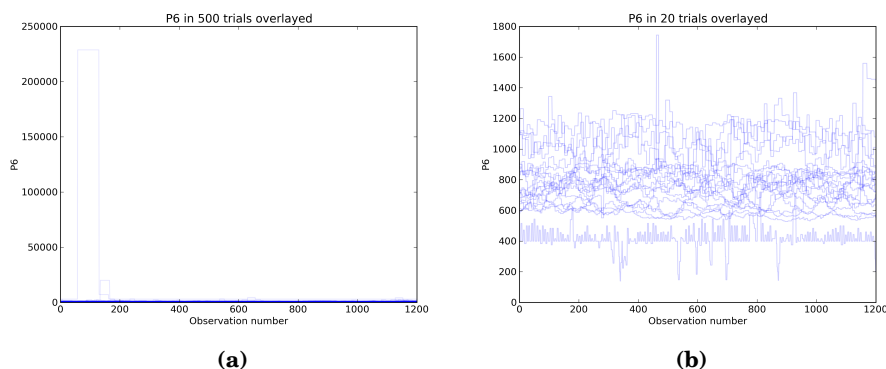
As many features have datapoints outside the 95 percentile even within a single trial, the next step could be to categorize these datapoints as outliers. But inspection of some examples of specific features in specific trials, shows that it would be wrong to remove these "outliers". Eg. feature V11 in trial 29 is plotted in figure 3.1 and seems to be perfectly normal. Careful inspection of the boxplot for feature V11 in trial 29 (figure 3.2) shows that the feature has many outliers in this specific trial.

The boxplots do not correctly identify the outliers in the dataset, and another method is therefore needed.





**Figure 3.2:** Boxplots of feature *E5*, *P1*, *P3* and *V11* in the first 50 trials



**Figure 3.3:** Layered feature plots of feature P6. All 500 trials are shown on the left. A couple of possible outliers are seen. On the right only 20 randomly chosen trials are plotted

### 3.4.2 Making layered feature plots

Instead of trying to categorize single datapoints as outliers, it may make more sense to define a specific feature in a specific trial as an outlier. The fact that some features deviates a lot from their normal structure, in a few trials was already mentioned in section 3.3.2 about exploratory plotting of features. Since the exploratory plotting was a rather unstructured process, a more thorough picture, of the structure of the different features across all trials, is needed. One way to get a better picture is by plotting all trials of a single feature, in a single plot. The graph of each trial is given a low opacity and then all 500 trials **TODO: update number based on number of training trials** are plotted on top of each other. That way the common structure of a feature across trials are visualized, and possible outlier trials can be detected. Two examples of this plot type can be seen in figure 3.3<sup>†</sup>

From the layered feature plots it is confirmed that many features have some trials that seems to deviate from the normal pattern of the feature. Using feature P6 as an example, it is seen in figure 3.3a that P6 in almost

<sup>†</sup>Source code for the layered feature plots can be found in appendix A.7 and the results are in appendix A.8

all trials lies within the range  $[0, 2000]$ . But one trial has a max value larger than 200000. This behaviour was hinted in section 3.2 and now there is a visual confirmation.

The question is, how should this “outlier” trial of feature P6 be handled? One approach is to exclude the trial from the dataset. But as nothing about the origin of the dataset is known, it can’t be justified with a common sense argument, and if this trial is excluded then how is the line between outlier and normal drawn? There is a couple of other trials where the P6 feature has values as high as 20000, but should this be seen as outliers? And are there other ways a trial could be an outlier than the scale? In figure 3.3b most trials have the same structure, but the trial with the lowest mean has a noticeable different shape than the other trials. It somehow has a baseline at the value 400 and then oscillates at a higher frequency than the other trials. Is this an outlier? If it is regarded as an outlier many other trials could possibly be classified as outliers, and we’re still only looking at one feature. If the same amount of cleaning is done for the other features, not much is left for the “normal” dataset.

The immediate conclusion is that the outlier detection easily becomes, either too all-encompassing or is done in an inconsistent manner. An alternative way to tackle the trials with datapoints with large values, could be to do some kind of variable transformation (eg. taking the log) of the feature. This could be a solution and the idea is investigated a little more in section 5.

### 3.4.3 Features that are constant in a trial

Another type of potential outlier trials, are all the trials where one or more features are constant throughout the trial. This could be a sign of a failure in the measurement equipment, and then maybe the trial should be regarded as an outlier? A great example is the IsAlert feature, that is the feature that should be predicted by the classifier. In 142 out of the 400 trials the IsAlert feature is constant and in 127 of the trials with a constant value, the driver isn’t alert. At first it may seem a bit odd, and indeed some of the contestants of The Ford Challenge revealed that they had excluded all trials where the IsAlert feature was 0 throughout the trial (see

David (2011)). But it is speculative to do, since the precise meaning of the IsAlert feature isn't known, let alone the fact that, measuring human alertness with a binary variable is speculative by itself.

As it seems to be speculative to exclude features with constant values, this idea won't be pursued any further. Instead the focus changes from "finding features to get rid of" till "finding features that could be beneficial for classification" as the next section is all about finding features that can discriminate between alert/not-alert drivers.

## 3.5 Finding possible discriminating features

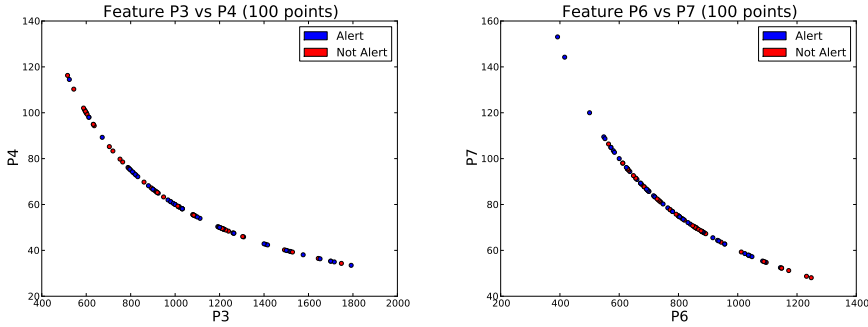
In this section all features are examined in an attempt to find one or more features that are in some way correlated with the IsAlert feature that should be predicted.

### 3.5.1 Testing binary features

A tempting first attempt at finding a feature that can be used to predict the IsAlert feature, is to look at the two binary features V5 and E9 and see if they are correlated with the predictor **TODO: What is the IsAlert feature called?**.

### 3.5.2 Scatterplots

A great way to visualize whether a pair of features can be used to discriminate between datapoints where the driver is alert and where he/she is not alert, is to create scatterplots of the possible pair of features. Since feature V7, V9 and P8 are removed and feature E3, E7, E8, E9, V3, V5 and V10 was found to be categorical, only 20 features are left for the scatterplots. Since the order of the two features in a scatterplot doesn't matter there is a total



**Figure 3.4:** Scatterplots of the feature pairs (P3, P4) and (P6, P7). An arithmetic relationship between the features is clearly present

of

$$\binom{20}{2} = 190 \text{ combinations}$$

All combinations of the 20 features was plotted<sup>‡</sup>, but no interesting results was found; at least not regarding feature pairs that could classify the IsAlert feature. One interesting result was found though. As can be seen in figure ?? the features pairs (P3, P4) and (P6, P7) both appear to have an arithmetic relationship. A quick calculation shows that P3 multiplied by P4 is  $60000 \pm 0.2$  for all rows in the dataset, and likewise for P6 and P7. We can therefore ignore one of the features from each pair. **TODO: Am I sure? Details can't hide in the decimal difference?**

## 3.6 Principal Component Analysis

### 3.6.1 Theory<sup>§</sup>

The idea in Principal Component Analysis (PCA) is to find a mapping from a  $d$ -dimensional input space, to a new  $k$ -dimensional space (with

<sup>‡</sup>Source code can be found in appendix A.9 and the results in appendix A.10

<sup>§</sup>Theory section is based on Alpaydin (2010, p.113)

$k < d$ ), while keeping as much of the variation in the input data, as possible. To begin with let  $k = 1$ . Then the subspace is identified by a  $d$ -dimensional vector  $\mathbf{w}_1$ , that without loss of generality is assumed to have length  $\|\mathbf{w}_1\| = 1$ .

For any vector  $\mathbf{x}$  in the input space, the projection onto  $\mathbf{w}_1$  is given by

$$z_1 = \mathbf{w}_1^T \mathbf{x}$$

and we now wish to maximize the variance of  $z_1$ . By the definition of variance we get (with  $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ )

$$\begin{aligned} \text{Var}[z_1] &= \text{Var}[\mathbf{w}_1^T \mathbf{x}] \\ &= \mathbb{E}[(\mathbf{w}_1^T \mathbf{x} - \mathbf{w}_1^T \boldsymbol{\mu})^2] \\ &= \mathbb{E}[(\mathbf{w}_1^T \mathbf{x} - \mathbf{w}_1^T \boldsymbol{\mu})(\mathbf{w}_1^T \mathbf{x} - \mathbf{w}_1^T \boldsymbol{\mu})] \\ &= \mathbb{E}[\mathbf{w}_1^T (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{w}_1] \\ &= \mathbf{w}_1^T \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] \mathbf{w}_1 \\ &= \mathbf{w}_1^T \Sigma \mathbf{w}_1 \end{aligned}$$

where  $\Sigma = \text{Cov}[\mathbf{x}]$ . So to maximize the variance of  $z_1$  is the same as maximizing  $\mathbf{w}_1^T \Sigma \mathbf{w}_1$ , subject to the constraint that  $\mathbf{w}_1^T \mathbf{w}_1 = 1$ . By introducing a Lagrange multiplier  $\alpha$ , we get

$$\max_{\mathbf{w}_1} \mathbf{w}_1^T \Sigma \mathbf{w}_1 - \alpha(\mathbf{w}_1^T \mathbf{w}_1 - 1)$$

This is a normal, unconstrained maximization problem. Taking the derivative with respect to  $\mathbf{w}_1$ , and setting this equal to 0, gives

$$\begin{aligned} 2\Sigma \mathbf{w}_1 - 2\alpha \mathbf{w}_1 &= 0 & \Leftrightarrow \\ \Sigma \mathbf{w}_1 &= \alpha \mathbf{w}_1 \end{aligned}$$

which is true exactly when  $\mathbf{w}_1$  is an eigenvector of  $\Sigma$  and  $\alpha$  is the corresponding eigenvalue. Giver det mening på binære variable?

### 3.7 Conclusions

## **Part II**

# **Modelling**





## CHAPTER 4

# Theory of classification

---

In this chapter some of the theory of classification is explained. The discussion is constrained to binary classification, of which the Ford Challenge classifier, is an example. First a general problem definition is given and notation is introduced. Then three different approaches to classification are introduced, and some pros and cons of each approach are mentioned. After this two concrete examples of classification models are introduced (Logistic Regression and Feed Forward Neural Network), and they are related to the three general approaches of classification. Finally different ways to measure the performance of different classifiers are discussed, including AUC that was used as the grading method in The Ford Challenge.

## 4.1 The binary classification problem<sup>\*</sup>

In a binary classification problem a binary outcome  $t$  must be predicted from a  $d$ -dimensional input vector  $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ . The input vector

---

<sup>\*</sup>This section is loosely based on [Wasserman \(2004, sec 22.1-22.2\)](#)

represents an event, and the outcome variable  $t$  represents the assignment of the event to one of two classes.

**EXAMPLE 4.1** *In The Ford Challenge the input is an instant in a driving situation, and this input should be assigned to either the class alert, or the class not-alert. The input is represented by an 30-dimensional vector and the assignment to a class is represented by  $t = 0$  meaning not-alert and  $t = 1$  alert.*

To make the prediction possible, a trainingset  $\mathcal{T}$  is given consisting of  $n$  pairs of input vectors and corresponding, known outcomes.

$$\mathcal{T} = \{(\mathbf{x}_i, t_i)\}$$

Based on this trainingset, a classification rule  $f$  is learned, that can predict the outcome, of yet unknown input vectors. Therefore a classification rule is a function  $f : \mathbb{R}^d \rightarrow \{0, 1\}$ , that is used to make the prediction  $t = f(\mathbf{x})$  on a new input.

A common way to obtain a classification rule is by the Bayes classification rule. First define  $p_k(\mathbf{x})$  as

$$p_k(\mathbf{x}) = P(t = k | \mathbf{x}), \quad k \in \{0, 1\}$$

and then the Bayes classification rule is defined by

**DEFINITION 4.1** *Let  $\alpha \in [0, \infty[$ . The Bayes classification rule  $f^*$  is given by*

$$f^*(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha p_1(\mathbf{x}) > p_0(\mathbf{x}) \\ 0 & \text{else} \end{cases}$$

Since  $p_0(\mathbf{x}) = 1 - p_1(\mathbf{x})$ , the Bayes classification rule can be written

$$f^*(\mathbf{x}) = \begin{cases} 1 & \text{if } p_1(\mathbf{x}) > \frac{1}{1+\alpha} \\ 0 & \text{else} \end{cases}$$

By the Bayes classification rule we have got a theoretical classification rule. In practice the posterior class probabilities  $p_0(\mathbf{x})$  and  $p_1(\mathbf{x})$  are unknown, so the trainingset must be used to find some approximation for

these probabilities. The problem of approximating a probability distribution from a given data set is a classic statistical problem. The next section gives one way to solve this problem.

## 4.2 Parametric models and maximum likelihood

Two distinct ways to approximate the probability distribution  $P(t|\mathbf{x})$  is using either a parametric or a non-parametric approach. The focus in this report is on a parametric approach. The distribution  $P(t|\mathbf{x})$ , is therefore approximated by a distribution  $\hat{P}(t|\mathbf{x})$  restricted to a class of distributions

$$\mathcal{F} = \left\{ f(t|\mathbf{x}, \mathbf{w}) \right\}$$

where the distributions  $f_{\mathbf{w}} \in \mathcal{F}$  are uniquely identified by their parameter  $\mathbf{w} = [w_1, w_2, \dots, w_k]^T$ . By restricting the approximating distribution  $\hat{P}(t|\mathbf{x})$  to the set  $\mathcal{F}$ , the problem of approximating  $P(t|\mathbf{x})$  reduces to the problem of estimating the parameter  $\mathbf{w}$ .

### 4.2.1 Maximum likelihood

There exists some different techniques to estimate the parameter  $\mathbf{w}$  in a parametric model.<sup>†</sup> In this section we look at the most common method for estimation of the parameter in a parametric model, namely the maximum likelihood method.

The approximation  $\hat{P}(t|\mathbf{x})$  of the posterior class probability is restricted to the class  $\mathcal{F}$ . Using the trainingset  $\mathcal{T}$ , the likelihood function is now defined as the probability of obtaining the data in the trainingset, given as a function of the parameter  $\mathbf{w}$ . More formally<sup>‡</sup>:

**DEFINITION 4.2** *The likelihood function is defined by*

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^n f(t_i|\mathbf{x}_i, \mathbf{w}), \quad (t_i, \mathbf{x}_i) \in \mathcal{T}$$

<sup>†</sup>See eg. Wasserman (2004, Sec.9)

<sup>‡</sup>Taken directly from Wasserman (2004, p.122)

and the log-likelihood function is defined by  $\ell(\mathbf{w}) = \log \mathcal{L}(\mathbf{w})$ .

It is worth noticing that the definition of the likelihood function assumes that the  $t_i$ 's are independent random variables. An assumption that in the case of The Ford Challenge may be a bit questionable.

Given the definition of the likelihood function, the maximum likelihood estimator is now defined by

**DEFINITION 4.3** *The maximum likelihood estimator  $\hat{\mathbf{w}}$  is the value of  $\mathbf{w}$  that maximizes  $\mathcal{L}(\mathbf{w})$ .*

Notice that as the logarithm is a monotonic increasing function, the maximum of  $\mathcal{L}(\mathbf{w})$  is equal to the maximum of  $\ell(\mathbf{w})$ . This turns out to be handy, since the log-likelihood often is easier to work with, than the likelihood function.

With these definitions it is now time to introduce two concrete examples of classification methods.

### 4.3 Logistic Regression

In the logistic regression the parametric form of the posterior class probability  $P(t = 0|\mathbf{x})$  is assumed to be

$$P(t = 0|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}} \quad (4.1)$$

where  $\mathbf{w} = [w_1, w_2, \dots, w_d]^T$ . The expression may seem a bit random, but it comes from the fact that (with  $p_0 = P(t = 0|\mathbf{x})$ )

$$p_0 = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}$$

can be written

$$\begin{aligned}
 p_0 &= \frac{e^{\mathbf{w}^T \mathbf{x} + w_0}}{1 + e^{\mathbf{w}^T \mathbf{x} + w_0}} && \Leftrightarrow \\
 p_0 + p_0 e^{\mathbf{w}^T \mathbf{x} + w_0} &= e^{\mathbf{w}^T \mathbf{x} + w_0} && \Leftrightarrow \\
 \frac{p_0}{1 - p_0} &= e^{\mathbf{w}^T \mathbf{x} + w_0} && \Leftrightarrow \\
 \log \frac{p_0}{1 - p_0} &= \mathbf{w}^T \mathbf{x} + w_0
 \end{aligned}$$

And therefore the parametric form in (4.1), comes from the assumptions that the log-odds of  $p_0$  is linear in the input vector  $\mathbf{x}$ .

Now to find an expression for the likelihood function, the probability of getting the data in the trainingset must be expressed. With  $p_i = P(t = 0 | \mathbf{x}_i)$  the likelihood function can be written as

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^n p_i^{t_i} (1 - p_i)^{1-t_i}$$

which gives the log-likelihood function

$$\ell(\mathbf{w}) = \sum_{i=1}^n t_i \log p_i + (1 - t_i) \log(1 - p_i)$$

The maximum of the log-likelihood function gives the maximum likelihood estimator of  $\mathbf{w}$ , so the derivative of  $\ell$  wrt.  $\mathbf{w}$  need to be derived. Using the fact that

$$\frac{\partial p_i(\mathbf{w})}{\partial \mathbf{w}} = p_i(1 - p_i)\mathbf{x}_i$$

gives that

$$\begin{aligned}
 \frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}} &= \sum_{i=1}^n t_i \frac{1}{p_i} p_i(1 - p_i)\mathbf{x}_i + (1 - t_i) \frac{1}{1 - p_i} (-p_i)(1 - p_i)\mathbf{x}_i \\
 &= \sum_{i=1}^n t_i(1 - p_i)\mathbf{x}_i - (1 - t_i)p_i\mathbf{x}_i \\
 &= \sum_{i=1}^n (t_i - p_i)\mathbf{x}_i
 \end{aligned}$$

The maximum likelihood estimator can now be found by solving  $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}} = 0$ , but since  $p_i$  isn't linear wrt.  $\mathbf{w}$  the equation must be solved numerically. An effective algorithm called iterative reweighted least squares can solve this problem (Bishop, 2006, p.207).

### 4.3.1 The decision boundary

An important property of a classification method is how complicated data sets the method is able to separate in the appropriate classes. The decision boundary of a classification method gives some information about how well it separates input data. The decision boundary is the set of points in the input space that lays at the boundary between input points that are classified as class 0 and input points that classify as class 1.

**EXAMPLE 4.2** *A trainingset consists of  $n$   $d$ -dimensional input data  $\mathbf{x}_i = [x_{1i}, x_{2i}, \dots, x_{di}]^T$  and corresponding classes  $t_i$ . Using linear regression, approximations of the posterior class probabilities  $\hat{p}_k(\mathbf{x}) = \hat{P}(t = k|\mathbf{x})$ ,  $k \in \{0, 1\}$  are given. Using these with the Bayes classification rule gives*

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \hat{p}_1(\mathbf{x}) > \hat{p}_0(\mathbf{x}) \\ 0 & \text{else} \end{cases}$$

*The decision boundary is given by all  $\mathbf{x}$  where  $\hat{p}_0(\mathbf{x}) = \hat{p}_1(\mathbf{x})$ . But by the definition of the linear regression*

$$\log \frac{\hat{p}_1(\mathbf{x})}{\hat{p}_0(\mathbf{x})} = \mathbf{w}^T \mathbf{x} + w_0$$

*So for all points at the decision boundary the following holds*

$$\mathbf{w}^T \mathbf{x} + w_0 = 0$$

*This shows that the decision boundary of the logistic regression is linear in the inputs.*

The classification methods with linear decision boundaries, defines a family of classification methods with simple analytical and computational properties (Bishop, 2006, p.179). This simplicity comes at the cost of the models having a limited capability to separate complex data sets.

### 4.3.2 Model complexity

Although the logistic regression has a limited complexity, its ability to separate complex datasets can be increased by introducing feature transformations. Instead of just fitting the logistic regression to the  $d$ -dimensional

input  $\mathbf{x} = [x_1, x_2, \dots, x_d]$ , the squared transformations of the elements  $x_i$  are added to the input. This gives the  $2d$ -dimensional input

$$\mathbf{x}_e = [x_1, \dots, x_d, x_1^2, \dots, x_d^2]$$

This gives nonlinear decision boundaries in the original input space, which increases the number of datasets that can be separated. Notice that the decision boundary is still linear in the extended  $2d$ -dimensional inputspace. Also there still exists datasets that can't be separated by a linear classifier, no matter what feature transformations are introduced **TODO: Morten: Is this correct? Where can I find a theorem stating this?**

### 4.3.3 Regularization

## 4.4 Feed Forward Neural Network

The Neural Network originated from an attempt to create a mathematical model of information processing in the human brain (Bishop, 2006, p.226). But the Neural Network turned out to be useful in many practical applications, and it is now seen as a standard statistical model (Hastie et al., 2009, p.392).

### 4.4.1 Model complexity

### 4.4.2 Regularization

## 4.5 Measuring classifier performance

When a classifier has been trained, an estimate of the classifiers expected performance on future inputs, should be calculated. To calculate this estimate, it needs to be specified exactly what is meant by a classifiers performance. For that a loss function is defined

**DEFINITION 4.4** Let  $t_0$  be the true class of a corresponding input  $\mathbf{x}_0$ , and  $f$  be a classification rule. The loss function

$$L(t_0, f(\mathbf{x}_0))$$

then gives the cost of making the prediction  $f(\mathbf{x})$ , when the true class is  $t_0$ .

The common way to define the loss function is by

$$L(t, y) = \begin{cases} 1 & \text{if } t \neq y \\ 0 & \text{else} \end{cases} \quad (4.2)$$

This assumes that the cost of predicting class 0 when the truth is class 1, and visa versa is the same. In many applications this may be questionable. In The Ford Challenge the cost of predicting alert, when the truth is not-alert, might be a more severe error, than the other way around; although it depends on what the Ford classifier is used for. In the rest of this section, the loss function given in (4.2) is assumed.

When a loss function has been defined, what would be interesting to know, is the expected loss on future data for a classification rule  $f$ .

$$\text{Err} = E[L(t, f(\mathbf{x}))]$$

where  $t$  and  $\mathbf{x}$  is random variables and the expectation is taken on their joint distribution. The expected loss isn't normally known so it needs to be estimated somehow. For that the average loss over a dataset for a given classification rule can be used. For an arbitrary dataset  $\mathcal{D} = \{(\mathbf{x}_i, t_i)\}, i = 1 \dots n$ , classification rule  $f$  and loss function  $L$ , the average loss is calculated by

$$\overline{\text{err}} = \frac{1}{n} \sum_{i=1}^n L(t_i, f(\mathbf{x}_i))$$

### 4.5.1 Splitting the dataset

It is extremely important that the dataset used to calculate the average loss, isn't in anyway overlapping with the dataset that was used to train the classifier. If there is a large dataset available for a classification task, the dataset should ideally be split into three separate datasets<sup>§</sup>

<sup>§</sup>Loosely based on [Hastie et al. \(2009, p.222\)](#)



- The training set  $\mathcal{T}$  - This dataset is used to train models. More specifically the training set is used to find the maximum likelihood estimator, for a parametric model
- The validation set  $\mathcal{V}$  - This dataset is used for model selection. As mentioned in section 4.3.2, the complexity of classification methods can be altered to make the method fit the training data better. The validation set gives an estimate of how well models of different complexity levels, predicts future datapoints.
- The publication set  $\mathcal{P}$  - This dataset is used, when the final model has been selected, and an estimate of the performance of this model.

There is not a general rule for how many percent of the original dataset that should be assigned to each of the three datasets. In [Hastie et al. \(2009, p.222\)](#), a split of 50% for training and 25% for both validation and publication, is suggested.

**EXAMPLE 4.3** *In The Ford Challenge a dataset of 500 trials is given. Following the suggestion by [Hastie et al. \(2009\)](#), the dataset is split in a training set of 250 trials and a validation set and publication set of both 125 trials. It is then decided to train a logistic regression model to classify drivers. The complexity of the model is increased by including various feature transformations. The average loss of the different models is calculated on the validation set, and the models with the lowest average loss is selected. Finally the average loss over the publication loss is used as an independent estimate of the expected loss on future data.*

The above method of making three distinct datasets out of the original, is only possible, when the original dataset is large. And even with a large dataset, it may seem a bit wasteful to only have tested the final model against 50% of the original dataset **TODO: Does this make sense?**. A way to circumvent this problem is by using cross validation.

### 4.5.2 Cross validation

Cross validation is a technique to squeeze out more information of a dataset, than by splitting the dataset in three parts as described in section 4.5.1 **TODO: Is this right? I have troubles defining cross validation? Is it only a technique for model selection? When assesing performance on test/publication set cross validation could be used?** Recommendation of 5- or 10-fold cross validation (Hastie et al., 2009, p.243)

### 4.5.3 AUC

Mentioning critiques of AUC?

## 4.6 Model selection

### 4.6.1 Over- and underfitting

### 4.6.2 Forward feature selection

### 4.6.3 Regularization

## CHAPTER 5

# Recreating winning approach

---

Here I describe how I have tried to recreate the winning approach. How I measure performance. The problem that I do not have access to the test data set used by Inference. My results. The scikits.learn library that I have used.



## CHAPTER 6

# Improving the winning approach

---

Here I try to improve the winning approach, by doing my own feature selection. Forward selection. Lasso. Cross validating with Lasso. Using window instead of running.



## CHAPTER 7

# Other classification methods

---

Here I describe some alternatives to the logistic regression used by Inference. Hoping to get a result or two from SVM or Neural Network.





## **Part III**

# **Workflow and discussion**



## CHAPTER 8

# Workflow and tools

---

Describing software used, workflow using github, writing sessions, evaluate my performance, mention the small improvement of roccurve method.



## CHAPTER 9

# Discussion

---

Bla, bla, bla.



## CHAPTER 10

# Conclusion

---

More bla, bla, bla.





CHAPTER **A**

# Appendices

---

A little introduction and then a new page

## A.1 Calculating common statistics - Code

```
# sessions/9-data-exploration/src/calculate_summary_statistics.py

from json import dump

from src.data_interface import trd, L
from src.utils import get_path

sess_root = get_path(__file__) + '/../scr'

summary_statistics = ['min', 'max', 'mean', 'std']
features_to_calculate = L[2:]
calculations = {}

for feature_name in features_to_calculate:
    calculations[feature_name] = {}
    for statistic in summary_statistics:
        stat_function = getattr(trd.get_feature(feature_name), statistic)
        calculations[feature_name][statistic] = stat_function()

f = open(sess_root + '/summary_statistics.json', 'w')
dump(calculations, f, indent=4)
f.close()
```

**Code Listing A.1:** [https://github.com/alphabits/ford\\_challenge/tree/master/sessions/9-data-exploration/src/calculate\\_summary\\_statistics.py](https://github.com/alphabits/ford_challenge/tree/master/sessions/9-data-exploration/src/calculate_summary_statistics.py)

A.2 Calculating common statistics - Result

Feature	Min in trial	Max in trial	Whole dataset
E1	1	131	12265
E10	1	86	121
E11	1	79	116
E2	1	129	28502
E3	1	3	3
E4	1	97	249
E5	1	134	254
E6	1	116	247
E7	1	15	25
E8	1	10	10
E9	1	2	2
IsAlert	1	2	2
P1	967	1208	105715
P2	1155	1207	128666
P3	24	106	406
P4	24	106	406
P5	34	114	1070
P6	3	99	406
P7	3	99	406
P8	1	1	1
V1	1	969	12374
V10	1	5	5
V11	875	1206	166455
V2	1	76	90
V3	1	29	33
V4	1	204	324
V5	1	2	2
V6	1	775	2727
V7	1	1	1
V8	1	239	323
V9	1	1	1

## A.3 Calculate unique values of features - Code

```
from json import dump

from src.data_interface import trd, L
from src.utils import get_path

sess_root = get_path(__file__) + '/../src'

features_to_calculate = L[2:]
unique_values = {}

for feature_name in features_to_calculate:
    unique_values[feature_name] = trd.get_feature(feature_name).unique_values()

f = open(sess_root + '/unique_values.json', 'w')
dump(unique_values, f, indent=4)
f.close()
```

**Code Listing A.2:** [https://github.com/alphabits/ford\\_challenge/tree/master/sessions/9-data-exploration/src/calculate\\_unique\\_values.py](https://github.com/alphabits/ford_challenge/tree/master/sessions/9-data-exploration/src/calculate_unique_values.py)

```
from json import dump
import numpy as np

from src.data_interface import trd, L
from src.utils import get_path

sess_root = get_path(__file__) + '/../'

features_to_calculate = L[2:]
trials = list(trd.trial_id_list)
calculations = {}

for feature_name in features_to_calculate:
    tmp = {"trial_results": {}}
    for trial_id in trials:
        unique_values = np.unique(
            trd.get_trial(trial_id).get_feature(feature_name).view())
        tmp["trial_results"][trial_id] = unique_values.size
    tmp["max"] = max(tmp["trial_results"].values())
    tmp["min"] = min(tmp["trial_results"].values())
    calculations[feature_name] = tmp

f = open(sess_root + '/src/unique_values_pr_trial.json', 'w')
dump(calculations, f, indent=4)
f.close()
```

---

**Code Listing A.3:** [https://github.com/alphabits/ford\\_challenge/tree/master/sessions/9-data-exploration/src/calculate\\_unique\\_values\\_pr\\_trial.py](https://github.com/alphabits/ford_challenge/tree/master/sessions/9-data-exploration/src/calculate_unique_values_pr_trial.py)

```
import json

from src.utils import get_path

path = get_path(__file__)

def get_dict(file_name):
    f = open(path + '/' + file_name, 'r')
    d = json.load(f)
    f.close()
    return d

unique_pr_trial = get_dict('unique_values_pr_trial.json')
unique_all_data = get_dict('unique_values.json')

values_combined = {}

for k in unique_all_data:
    tmp = {}
    tmp["all_data"] = unique_all_data[k]
    tmp["min_pr_trial"] = unique_pr_trial[k]["min"]
    tmp["max_pr_trial"] = unique_pr_trial[k]["max"]
    values_combined[k] = tmp

f = open(path + '/unique_values_combined.json', 'w')
json.dump(values_combined, f, indent=4)
f.close()
```

**Code Listing A.4:** [https://github.com/alphabits/ford\\_challenge/tree/master/sessions/9-data-exploration/src/unique\\_values\\_combined.py](https://github.com/alphabits/ford_challenge/tree/master/sessions/9-data-exploration/src/unique_values_combined.py)

A.4 Calculate unique values of features - Result

Feature	Min in trial	Max in trial	Whole dataset
E1	1	131	12265
E10	1	86	121
E11	1	79	116
E2	1	129	28502
E3	1	3	3
E4	1	97	249
E5	1	134	254
E6	1	116	247
E7	1	15	25
E8	1	10	10
E9	1	2	2
IsAlert	1	2	2
P1	967	1208	105715
P2	1155	1207	128666
P3	24	106	406
P4	24	106	406
P5	34	114	1070
P6	3	99	406
P7	3	99	406
P8	1	1	1
V1	1	969	12374
V10	1	5	5
V11	875	1206	166455
V2	1	76	90
V3	1	29	33
V4	1	204	324
V5	1	2	2
V6	1	775	2727
V7	1	1	1
V8	1	239	323
V9	1	1	1

A.5 Creating boxplots - Code

TODO: Include source code

## A.6 Creating boxplots - Result

**TODO: Include boxplots**

## A.7 Creating layered feature plots - Code

## A.8 Creating layered feature plots - Result

## A.9 Scatterplots - Code

```
import itertools as it

import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import numpy as np

from src.dataloaders import trainingset
from src.utils2 import get_path, c, L

L = list(L[4:])
D = trainingset()

path = get_path(__file__) + '/../'
savepath_template = '{0}/plots/scatterplots/{1}-{2}.pdf'
num_points = 100

rows = np.random.random_integers(0,D.shape[0]-1, num_points)
data = D[rows,:]

colors = map(lambda x: 'blue' if x==1 else 'red', data[:,c('IsAlert')])
blue = Rectangle((0,0),1,1,fc='b')
red = Rectangle((0,0),1,1,fc='r')

exclude = ['V7', 'V9', 'P8', 'E3', 'E7', 'E8', 'E9', 'V3', 'V5', 'V10']
features = [f for f in L if f not in exclude]

for f1, f2 in it.combinations(features, 2):
    #for f1, f2 in [['E4', 'E5']]:
        plt.title('Feature {0} vs {1} ({2} points)'.format(f1, f2, num_points),
                  {'size': 20})
        plt.legend((blue, red), ('Alert', 'Not Alert'))
        plt.scatter(data[:,c(f1)], data[:,c(f2)], c=colors)
        plt.gca().set_xlabel(f1, {'size': 18})
        plt.gca().set_ylabel(f2, {'size': 18})
```

```
plt.savefig(savepath_template.format(path,f1,f2), format='pdf',  
            papertype='a4')  
plt.cla()
```

**Code Listing A.5:** [https://github.com/alphabits/ford\\_challenge/tree/master/sessions/24-writing-helper-scripts/scripts/rewrite-scatterplot-script.py](https://github.com/alphabits/ford_challenge/tree/master/sessions/24-writing-helper-scripts/scripts/rewrite-scatterplot-script.py)

## A.10 Scatterplots - Result



# Bibliography

---

- Abou-Nasr, M. (2011a). Kaggle forum: About the parameter - Reply 3. URL <http://www.kaggle.com/c/stayalert/forums/t/317/about-the-parameter/1861#post1861>.
- Abou-Nasr, M. (2011b). Kaggle forum: Discrete or continuous values - Reply 2. URL <http://www.kaggle.com/c/stayalert/forums/t/266/discrete-or-continuous-values/1635#post1635>.
- Abou-Nasr, M. (2011c). Kaggle forum: Were units changed? - Reply 2. URL <http://www.kaggle.com/c/stayalert/forums/t/268/were-units-changed/1641#post1641>.
- ACM (2011). KDD Cup Center. URL <http://www.sigkdd.org/kddcup/index.php>.
- Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2 edition.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science + Business Media LLC.
- David (2011). Methods/Tips from non-top 3 participants. URL <http://www.kaggle.com/c/stayalert/forums/t/328/methods-tips-from-non-top-3-participants/1967#post1967>.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Science + Business Media LLC.

- Inference (2011). Kaggle forum: Ford - Reply 3. URL <http://www.kaggle.com/c/stayalert/forums/t/295/ford/1743#post1743>.
- Kaggle.com (2011). The Ford Challenge Data Files. URL <http://www.kaggle.com/c/stayalert/Data>.
- Labs, Y. (2011). KDD Cup 2011. URL <http://kddcup.yahoo.com/>.
- Netflix (2011). Netflix Prize Leaderboard. URL <http://www.netflixprize.com/leaderboard>.
- Pardos, Z. (2011). Kaggle forum: Top two teams with same AUC - Reply 1. URL <http://www.kaggle.com/c/stayalert/forums/t/327/top-two-teams-with-same-auc/1937#post1937>.
- Rosanne (2011). Kaggle forum: Top two teams with same AUC - Reply 4. URL <http://www.kaggle.com/c/stayalert/forums/t/327/top-two-teams-with-same-auc/1957#post1957>.
- Wasserman, L. A. (2004). *All of Statistics: a concise course in statistical inference*. Springer Science + Business Media Inc.
- Wikipedia (2011). Netflix Prize. URL [http://en.wikipedia.org/wiki/Netflix\\_Prize](http://en.wikipedia.org/wiki/Netflix_Prize).