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
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

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. 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 metioned 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 coorporation 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 exists, 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, needs the same functionality, websites that specializes in hosting machine learning competitions has 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)

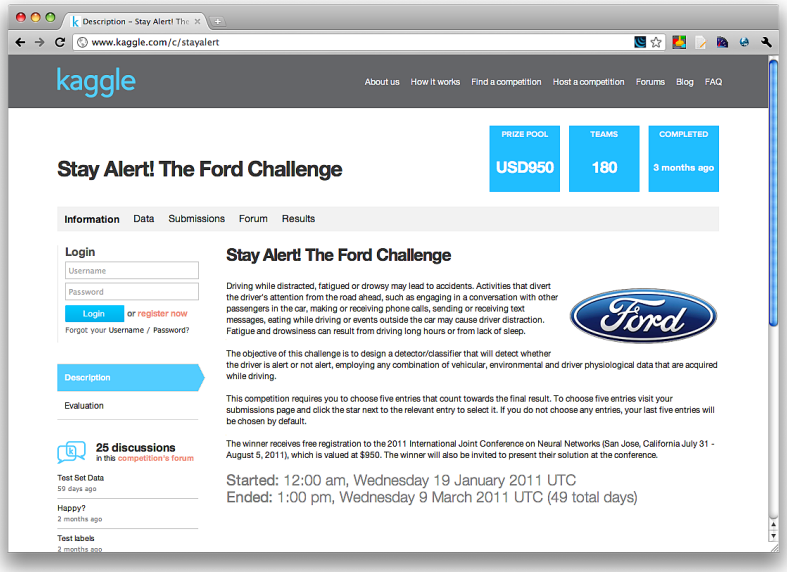


Figure 2.1: *The Ford Challenge frontpage at the kaggle.com website*

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[‡]. 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[§]. The official reason was that (Abou-Nasr, 2011b)

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

[‡]The dataset is explained in much more detail in the section 2.2.1

[§]see forum replies from the Ford spokesperson (Abou-Nasr, 2011a,c)

> 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 4.8) 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 cause 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 classfier 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)

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

Feature	Mean	Min	Max	Std.Dev
E9	0.87	0.00	1.00	0.33
IsAlert	0.58	0.00	1.00	0.49
P6	843.73	128.00	228812.00	2795.32
P8	0.00	0.00	0.00	0.00
V5	0.18	0.00	1.00	0.38
V7	0.00	0.00	0.00	0.00
V9	0.00	0.00	0.00	0.00

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

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 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 som trials where they only take on one unique value. For categorical features, this could be perfectly normal, but for a continous 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 continous, 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 continous. 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 continous 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.

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
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
V8	1	239	323

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.*

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.

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 i figure 3.1. The main results from the introductory plotting of features are that:

- A feature varies alot in structure between different trials
- Many features are constant in some trials (was already hinted at in section 3.3.1)
- Some features deviates 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 does indeed seems to be categorical features, although V3 do 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 i what the next section is about.

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

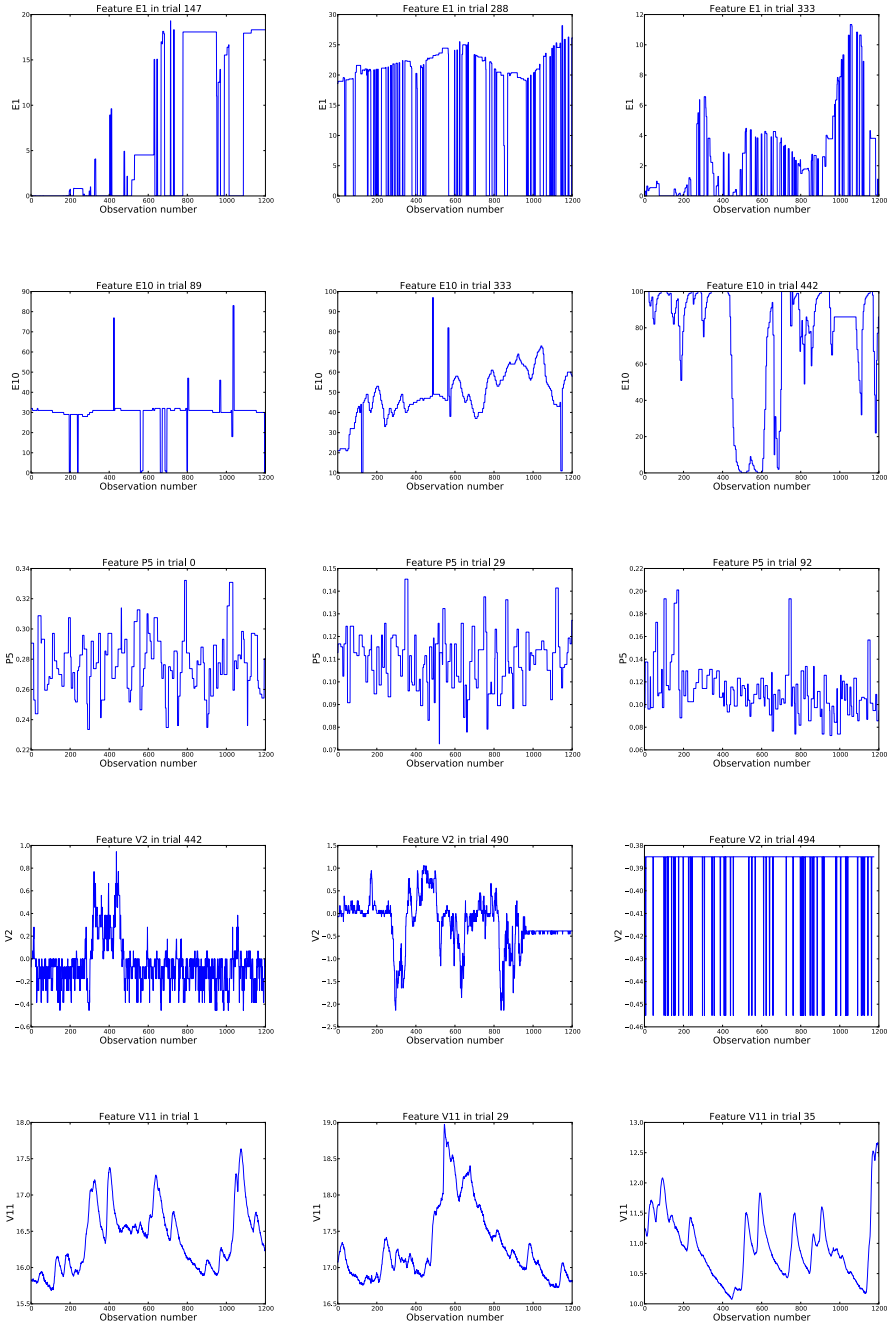


Figure 3.1: *Plots of the features* `E1`*,* `E10`*,* `P5`*,* `V2`*,* `V11`*, for a couple of trials. The trials was selected to show the diversity a single feature exhibits between trials.*

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 should be removed. Although this is a naive approach some useful information could be obtained from boxplots of the features. But i 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 i 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 boxpots 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.

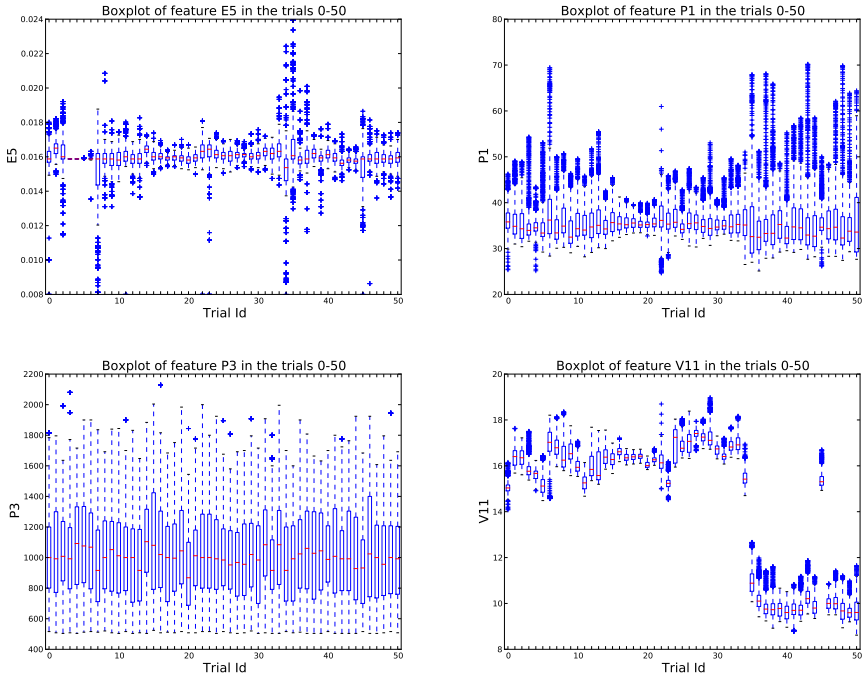
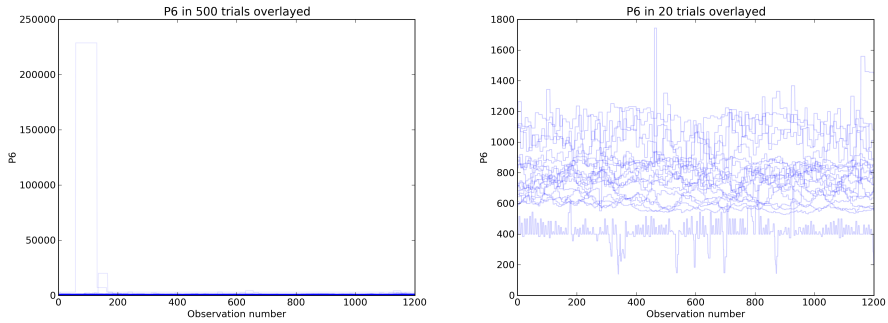


Figure 3.2: *Boxplots of feature E5, P1, P3 and V11 in the first 50 trials*



(a) **(b)**

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*

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

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 unstructered 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[†]

[†]Source code for the layered feature plots can be found in appendix A.7 and the results are in appendix A.8

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 all trials lies within the range [0,2000]. But one trial have 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 commen 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 a 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 are 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, measurering human alertness with a binary variable is speculative by itself.

As it seems to 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 benificial 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

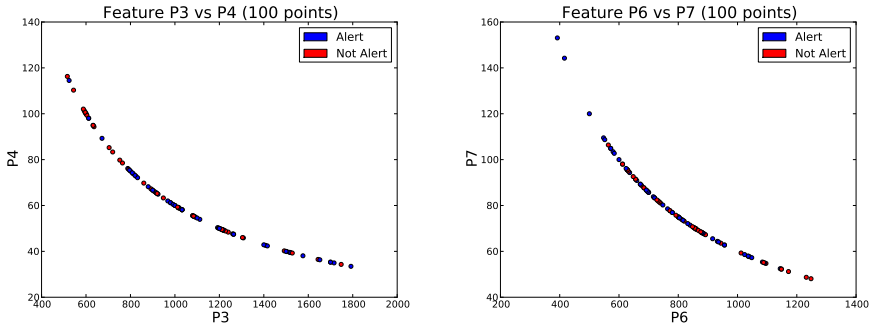


Figure 3.4: *Scatterplots of the feature pairs (P3, P4) and (P6, P7). An arithmetic relationship between the features is clearly present*

the order of the two features in a scatterplot doesn't matter there is a total of

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

All combinations of the 20 features was plotted[‡], 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 ± 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[§]

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

[‡]Source code can be found in appendix A.9 and the results in appendix A.10

[§]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 $\boldsymbol{w}_1$, that without loss of generality is assumed to have length $||\boldsymbol{w}_1|| = 1$.

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

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

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

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

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

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

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

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

which is true exactly when $\boldsymbol{w}_1$ is an eigenvector of Σ and α 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*

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

*This section is loosely based on **?**, 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.

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

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 $f : \mathbb{R}^d \to \{0,1\}$, that is used to make the prediction $t = f(x)$ on a new input.

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

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

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

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

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

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

By the Bayes classification rule we have got a theoretical classification rule. In practice the posterior class probabilities $p_0(\boldsymbol{x})$ and $p_1(\boldsymbol{x})$ 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 $P(t|\boldsymbol{x})$ is using either a parametric or a non-paramtric approach. The focus in this report is on a paramtric approach. The distribution $P(t|\boldsymbol{x})$, is therefore approximated by a distribution $\hat{P}(t|\boldsymbol{x})$ restricted to a class of distributions

$$\mathcal{F} = \left\{ f_{\boldsymbol{w}}(t|\boldsymbol{x}) \right\}$$

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

4.3 Logistic Regression

4.4 The binary classification problem

In a binary classification problem, a set of n inputs are given along with a set of n corresponding class labels. These two sets are called the trainingset. Based on this trainingset, a procedure is *learned*. The procedure should predict the class of a new input, with as few errors as possible. The input is represented by a d-dimensional vector

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

True \ Predict	Not Alert	Alert
Not Alert	0	100
Alert	1	0

Table 4.1: *An example of a loss table for The Ford Challenge classification problem.*

and the predicted class is represented by a binary variable $t \in \{0, 1\}$. The trainingset are then given by a set

$$\mathcal{T} = \left\{(\boldsymbol{x}_i, t_i)\right\}_{i=1}^{n}$$

In the Ford Challenge eg., the input is a driving instant and the classes are alert and not alert. The driving instant is represented by a 30-dimensional vector of physiological, environmental and vehicular features, and the alert/not-alert classes are represented by respectively $t = 1$ and $t = 0$.

A simple way to mathematically express the binary classification problem is: Given a trainingset $\mathcal{T}$, a discriminant function $f : \mathbb{R}^d \to \{0, 1\}$ must be learned such that errors on future inputs are minimized. Although this is a simple description, and concrete classifier methods using this approach exists[†], it is preferred (Bishop, 2006, p.43) to use a probabilistic description. In this perspective the classification consists of two separate steps, namely an inference step and a decision step.

In the inference step, the problem of determining the probabilities $P(t|\boldsymbol{x})$[‡], are solved**TODO: Should i use P for mass distribution and p for density distribution?**. These probabilities are called the posterior class probabilities.

In the decision step a loss table is set up describing the relative costs of predicting a class of an input given its true class. For The Ford Challenge, the loss table could eg. be as shown in table 4.1. The information in the loss table can be represented as a loss matrix L. In our example this means that eg. $L_{01} = 100$. Using the posterior class probabilities $P(t|\boldsymbol{x})$ a

[†]Eg. Bishop (2006, p.181)

[‡]In another approach the $P(t|\boldsymbol{x})$ isn't determined directly. Instead $p(\boldsymbol{x}|t)$ and $P(t)$ are determined and then using Bayes Formula $P(t|\boldsymbol{x})$ is calculated. This is called generative modelling.

decision about the predicted class label of an new input $\boldsymbol{x}_0$ can be made by the rule

$$\min_j \sum_k L_{jk} P(t = k|\boldsymbol{x}_0)$$

where $j,k \in \{0,1\}$. In words the decision step chooses the class label that minimizes the expected loss**TODO: Is this the right interpretation?**.

What is gained from making the two steps explicit? If we only have a discriminant function f, and the loss table is changed, we need to train a whole new discriminant function, using the trainingset again. If we instead have determined the posterior class probabilities, we only need to update the loss matrix, and don't have to retrain anything. Another reason is a common sense argument. When a class label is predicted for a new input, there will always be some level of uncertainty in the prediction. In the Ford Challenge the class label is predicted for a 30-dimensional input vector, but this vector is only a representation of the real input; namely an instant in a driving session. There is no doubt that some details that could influence the alertness of the driver aren't included in the 30 measurements, and this gives an uncertainty in the prediction. Probability is what is used to mathematically express uncertainty, and it therefore seems natural to take a probabilistic view on the classification problem.

The theory presented in this section has been on a general level. Two questions of practical importance needs to answered though. First of all, how can the posterior class probabilities $P(t|\boldsymbol{x})$ be determined from the training data? And secondly, how can the performance of the classifier on future inputs be determined, when training the classifier? The answer to the first question is delayed till the sections about specific classifier methods, and the second question is answered in the next section.

DEFINITION 4.2 Anders

4.5 Measuring classifier performance

4.6 Logistic Regression

4.7 Feed Forward Neural Network

4.8 AUC

Mentioning critiques of AUC?

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*

A.2 Calculating common statistics - Result

Feature	Mean	Min	Max	Std.Dev
E1	10.54	0.00	243.99	14.00
E10	63.38	0.00	127.00	19.23
E11	1.37	0.00	52.40	5.37
E2	105.05	0.00	360.00	128.33
E3	0.30	0.00	4.00	1.03
E4	-3.99	-250.00	260.00	34.80
E5	0.02	0.01	0.02	0.00
E6	358.50	260.00	513.00	27.84
E7	1.81	0.00	25.00	2.95
E8	1.39	0.00	9.00	1.62
E9	0.87	0.00	1.00	0.33
IsAlert	0.58	0.00	1.00	0.49
P1	35.45	-22.48	101.35	7.36
P2	12.01	-45.63	71.17	3.74
P3	1028.00	504.00	2512.00	309.70
P4	63.99	23.89	119.05	19.76
P5	0.18	0.04	27.20	0.39
P6	843.73	128.00	228812.00	2795.32
P7	78.40	0.26	468.75	18.79
P8	0.00	0.00	0.00	0.00
V1	75.58	0.00	129.70	44.94
V10	3.28	1.00	7.00	1.27
V11	11.59	1.68	262.45	8.43
V2	-0.04	-4.79	3.99	0.42
V3	569.78	240.00	1023.00	299.02
V4	21.24	0.00	484.49	67.21
V5	0.18	0.00	1.00	0.38
V6	1699.34	0.00	4892.00	626.27
V7	0.00	0.00	0.00	0.00
V8	12.46	0.00	82.10	11.54
V9	0.00	0.00	0.00	0.00

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*

```
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*

```
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*

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 featuer 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*

A.10 Scatterplots - Result

Bibliography

Mahmoud Abou-Nasr. Kaggle forum: Discrete or continuous values - Reply 2, 2011a. URL http://www.kaggle.com/c/stayalert/forums/t/266/discrete-or-continuous-values/1635#post1635.

Mahmoud Abou-Nasr. Kaggle forum: Were units changed? - Reply 2, 06 2011b. URL http://www.kaggle.com/c/stayalert/forums/t/268/were-units-changed/1641#post1641.

Mahmoud Abou-Nasr. Kaggle forum: About the parameter - Reply 3, 2011c. URL http://www.kaggle.com/c/stayalert/forums/t/317/about-the-parameter/1861#post1861.

ACM. KDD Cup Center, 06 2011. URL http://www.sigkdd.org/kddcup/index.php.

Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2 edition, 2010.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science + Business Media LLC, 2006.

David. Methods/Tips from non-top 3 participants, 2011. URL http://www.kaggle.com/c/stayalert/forums/t/328/methods-tips-from-non-top-3-participants/1967#post1967.

Inference. Kaggle forum: Ford - Reply 3, 06 2011. URL http://www.kaggle.com/c/stayalert/forums/t/295/ford/1743#post1743.

Kaggle.com. The Ford Challenge Data Files, 2011. URL http://www.kaggle.com/c/stayalert/Data.

Yahoo! Labs. KDD Cup 2011, 06 2011. URL http://kddcup.yahoo.com/.

Netflix. Netflix Prize Leaderboard, 06 2011. URL http://www.netflixprize.com/leaderboard.

Zach Pardos. Kaggle forum: Top two teams with same AUC - Reply 1, 06 2011. URL http://www.kaggle.com/c/stayalert/forums/t/327/top-two-teams-with-same-auc/1937#post1937.

Rosanne. Kaggle forum: Top two teams with same AUC - Reply 4, 06 2011. URL http://www.kaggle.com/c/stayalert/forums/t/327/top-two-teams-with-same-auc/1957#post1957.

Wikipedia. Netflix Prize, 06 2011. URL http://en.wikipedia.org/wiki/Netflix_Prize.