Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Obor: Fyzikální inženýrství
Zaměření: FTTF

Application of Machine Learning Tools to the Analysis of Tokamak Massive Databases

MASTER THESIS

Author: Bc. Michal Odstrčil
Supervisor: RNDr. Jan Mlynář, Ph.D.
Year: 2012

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 5.5.2012

.....
Bc. Michal Odstrčil

Název práce:
Aplikace učících se algoritmů v hromadné analýze dat z tokamaků

Autor: Bc. Michal Odstrčil

Obor: Fyzikální inženýrství
Druh práce: Výzkumná práce

Vedoucí práce: RNDr. Jan Mlynář, Ph.D.
ÚFP AV ČR v.v.i.
Konzultant: Dr. Andrea Murari
EFDA JET

Abstrakt: Tato práce se zabývá využitím učících se algoritmů pro hromadné zpracování dat. Pro otestování byly vybrány dvě úlohy určení pravděpodobností vzniku plazmatu na tokamaku GOLEM a hlavní část byla předpověď disrupcí na tokamaku JET. V práci byly nejdříve popsané použité algoritmy a dále byly popsány základní vlastnosti disrupcí potřebných pro jejich předvídání. V práci byly prozkoumány možné problémy vznikající při předpovídání a skóre a stability predikce kterých lze dosáhnout byly odhadnuty pro různě vybrané datové sady a vstupní parametry. V této práci se podařilo dosáhnout velmi slibných výsledků v predikci disrupcí, které by mělo být možno použit v budoucích tokamacích.

Klíčová slova: učící se algoritmy, hromadné zpracování dat, plazma , disrupce, průraz plazmatu

Title:
Application of Machine Learning Tools to the Analysis of Tokamak Massive Databases

Author: Bc. Michal Odstrčil

Abstract: This work is focused on use of learning machines for massive data processing. Two particular task were selected for testing: plasma breakdown probability in tokamak GOLEM and mainly disruptions prediction for tokamak JET. At the beginning of this work, the used learning machine algorithms were described. Furthermore, properties of disruptions necessary for the prediction were mentioned. Possible problems of the predictions were investigated and the score and its stability was estimated for several different training sets and different parameters. Many unique results were achieved in this work that could be possible to use in the future tokamaks.

Key words: machine learning, massive data processing, plasma, disruptions, breakdown

Contents

List of abbreviations

ARD	Automatic relevance Determination
AUC	Area Under Curve (in this work used only under DET curve)
ASDEX	large tokamak in Garching, Germany
CART	Classification and Regression Trees
CPD	Conditional Probability Distribution
CV	Cross-Validation
DAS	Data Acquisition System
DET	Detection Error Trade-off curve
EA	Early alarms
FA	False alarms
FS	Feature Selection
ITER	International Experimental Thermonuclear Reactor (built tokamak in Cadarache, France)
JET	Joint European Torus (large tokamak in Culham, Great Britain)
JPS	JET Prediction System
LogReg	Logistic Regression
MA	Missed alarms
MAP	Maximum A priori solution
MHD	Magnetohydrodynamics
ML	Maximum Likelihood
NN	Neural Networks
Q95	Safety factor (magnetic field helicity) at 95% of normalized distance from SOL
RBF	Radial Basis Functions
RFE	Recursive Feature Elimination
RV	Relevance vectors
RVM	Relevance Vector Machine
SOL	Scrape of Layer –The outer layer of a magnetically confined plasma, where the field lines come in contact with a material surface
SV	Support Vectors
SVM	Support Vector Machines
UFO	unidentified flying object

Abbev.	Name
ASD	Too little auxiliary power
GWL	Greenwald limit
IMC	Impurity control problem
IP	Too fast a current ramp-up
ITB	Too strong internal transport barrier
LON	Too low density (and low q)
NC	Density control problem
NTM	Neo-classical tearing mode
???	Nonclassified
INT	Intentional

Table 1: Types of plasma disruptions in used this work

Abbrev.	Diagnostics
IPLA	Plasma current
BTPD	Poloidal beta
LOCK	Mode lock amplitude
PIN	Total input power
PPOZ	Plasma vertical centroid position
DENS	Plasma density
WDIA	Derivative of stored diamagnetic E
POUT	Total radiated power
INDU	Plasma internal inductance
NGW	Greenwald density
TAU	Confinement time
BETN	Beta normalized
Q95	Safety factor

Table 2: The complete list of all the downloaded and tested JET database outputs

Introduction

In plasma physics, particularly in processing data from tokamaks, physicists are dealing with huge amounts of data to process. The learning algorithms can help in post-processing to identification of interesting events or classify data according to some criteria.

Learning machines have a wide field of use. They are used in physics, medicine, biology, economy, etc. Many different algorithms already exist for many common tasks such as optimization, regression or classification. This will be described in detail in following chapters.

This work is focused on use of learning machines for plasma physics especially for predictions of plasma disruptions. The plasma disruption is in a simplified way defined as a sudden and unexpected end of plasma. But generally, the disruption in tokamak is a dramatic event in which the plasma confinement is suddenly destroyed [1]. The disruptions are in detail described in Section 6.1.

The main topic of this work is investigation of potential issues of application of the learning machines in the case of a small and incomplete training database of the plasma disruptions. This will be an important issue for the new international ITER tokamak under construction in France. The aim was to develop and algorithm with a real-time prediction possibility, although the real-time prediction itself was out of this work. The real-time prediction possibility was the main limiting factor for choice of the input variables of the models because not all important and relevant plasma parameters are accessible with less than 1 ms time response. Other issues, studied in this work, were caused by huge amounts of data and a sparse information value contained in the database. It was necessary to find the most appropriate learning machine algorithm and the best preprocessing to maximize number of disruptions detected 30 ms before end of the plasma while the number of false alarms should be as low as possible. Moreover, influence of different parameters was tested and the best set of parameters in order to maximize the prediction score was proposed. Finally, properties of different types of disruptions were investigated because the origin of the disruption is a major factor determining the possible prediction time.

Chapter 1

Massive Data Processing in Fusion

The massive data processing and machine learning are becoming important branches in the fusion science. The massive data processing also called "data mining" is a popular discipline because it is a cheap way how to obtain new useful data from an already measured and processed database. Data mining is becoming even more popular with the growing size of the stored data during each discharge 1.1 and with development of more sophisticated data processing algorithms. Further, usually more than one algorithm can used for the data mining in order to improve results and processing time. For example in the video processing the first algorithm can be used to identify the most important frames, it ie. the key-frames in the compressed video, and the second slower algorithm is used for the video processing.

One of the most important tools used for data mining and signal processing are learning machines. The set of algorithms called learning machines is so wide that it is problematic to define them. Generally, the learning machines are algorithms able to extract useful information from the training data. Moreover, the algorithms are usually able to apply the learned information on new data.

The basic applications of the learning machines can be the classification of the data to several groups. It can be either automatically without any human inference (clustering, non-supervised methods) or according to an already (manually) classified dataset – supervised learning. Further, some algorithms are able to do the regression in order to either predict new values in the training range or to smooth the training data and remove noise

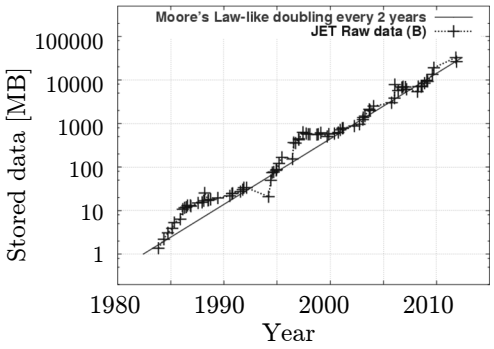


Figure 1.1: Total raw data stored per a shot to the JET tokamak database [2].

with minimal loose of information. The learning machines are sometimes used even for the extrapolation however the credibility of the prediction is an issue.

Furthermore, the applications in the nuclear fusion and in the physics can be significantly different from the general use of the learning machine such as handwriting recognition, speech recognition or market predictions. In the physical applications, the information about reliability of the prediction is often requested and also the creditability of the inputs is usually known, therefore it is possible to use this information (see Section 3.2.2). However, in many cases the uncertainties in the inputs are neglectable in comparison with the uncertainties in the physical background ie. hidden variables in the model and outliers.

Common applications of the learning machines for tokamaks are for example

- LH-transition detections – algorithms for automatic recognition of the exact LH transition time useful for study of conditions needed for the transition.
- image or video processing – video stabilization, detection of hotspots useful for the plasma facing components protection, object tracking – tracking of pellets, UFOs or turbulent structures in plasma
- Disruptions predictions – promising field of research, when the past plasma disruptions are used for the for real-time disruption prediction in tokamak. Moreover, some attempts to extrapolate the predictions for other tokamaks exist, however the results are rather poor.
- Fast radiation and magnetic reconstruction – learning machines can trained to duplicate results from the time demanding algorithms such as tomographic reconstruction to perform the predictions in real time. However, the learning machines still have issues with the extrapolation out of the training area.
- Model approximation – it is generally possible to perform different real-time predictions based on the learning machines trained on some sophisticated but slow models.

In spite of many advantages of the learning machines, the applications can often reach the limits of the computation power during the training phase. Therefore, many parallelized versions of the learning machines algorithms is being developed and also implementations for the graphical cards based on the CUDA drivers exist. On the other hand, the predictive phase can be very fast and it is often a great advantage compared to other more complex methods. Thanks to this property, learning machines are often used for real-time systems, for example real-time tomography [3], real-time L-H mode transition detection [4, 5], real-time plasma disruptions detection [6]. The advantage of learning machines is faster processing compared the regular analysis like tomography. On the other hand, learning machines are not able to reliably extrapolate the training data and thus if the observed problem is significantly different from the training data, the prediction can be useless.

Another, promising field of data processing is video processing. Almost 40% of the JET database consists from video data and 40 high speed cameras is currently installed. The

learning machines can be used either for an automatic classification of the events in video or for detection of the most interesting frames suitable for further processing.

The next hot topic that is being investigated are possible estimations if credibility and confidence intervals of the models. This allows to estimate reliability of the predicted values and base the decision on a risk threshold. Other use is "active learning" when the algorithm selects from a database only the points that gives to the model most of the new information, ie. from regions with the lowest predicted credibility.

And finally, many attempts exists to extrapolate the results from the current tokamak towards to the ITER or even DEMO. For these predictions, multi machine databases are usually used in order to include even the geometric properties of tokamaks and find a credible model over wide range of parameters. Learning machines or even the ordinary least squares fitting are used for example to estimate the LH transition power [7, 8], disruptions predictions [9], ELMs mitigations, ... The best model selection is crucial with the extrapolation. In the case of the best model for interpolating, many criterion exists (Akaie Information criterion (AIC), Bayes Information Criterion (BIC), Structural Risk Minimization (SRM)), however the extrapolation needs more strict criteria that penalize too flexible models, moreover the plasma physics do not fulfill many of the assumptions for this basic criteria such as i.i.d. (identically independently distributed) points, known noise distribution, no hidden parameters. Therefore, so called Model Falsification Criteria are being developed in order to find a simple models robust to the outliers are with minimal mathematical assumptions.

The future of the data processing based on learning machines directs towards processing even more data, therefore parallelization, graphical cards use and huge databases will be even more necessary. In particular, the applications for the ITER will be most probably based on hybrid models trained on the slow but complex simulations of the tokamak operations with some human a priory knowledge and later also with the first real data. The use of the learning machines allows to apply all the knowledge and simulations in real-time tokamak control. Moreover, possibility of model credibility estimations allows to stop the operation in the case that the probability of failure$\times$possible damage cross some threshold.

Nowadays, the learning machine algorithms can be tested on the current tokamaks and use the cross-machine learning to find the optimal methods that should work even for the ITER. Unfortunately, the results of this attitude for disruptions prediction are rather poor [9] because the necessary precision of the models is in order of percents of the parameters magnitude and the differences caused by the non included factors such as tokamak geometry or material of wall are not negligible.

Chapter 2

Learning Algorithms Background

2.1 Basic Properties of Learning Machines

Before introducing various algorithms, it is necessary to define some of the key differences among them. The number of different learning machines is huge and only some groups are used in this work:

- *Regression, Classification*: If the data labels can take values only from a small set then the goal of machine learning is to predict which group the data belong to. On contrary, in the case of continuous labels the aim is to perform regression. However, two cases are basically very similar and some learning algorithms can do both.
- *Supervised, unsupervised, semisupervised*: If learning machines are trained on data labeled by an expert, then it is called supervised learning. If the data are not labeled then it is called unsupervised learning. The unsupervised learning algorithms can for example detect separated groups or can be used for detection of outliers if there is only one group. However, the unsupervised learning is commonly much less reliable then the supervised. It is possible to use a combination so-called semisupervised learning, where some data are labeled by an expert and some data are unlabeled. The unlabeled values are classificated according to the labeled data. Finally, newly labeled data are used in the following learning. Learning machines, used in this work, belong to the group of supervised learning, the only exception is One Class SVM.
- *Dense, sparse model*: In case that the training data are used for prediction, the models are divided to two basic groups. If all data points are used for prediction then the model is cased dense, an widely used example is the Nearest Neighbors algorithm. The second group. sparse models, are using only limited number of the training points on order to increase prediction speed or avoid overfitting.
- *Online or offline learning*: Online learning means that each newly predicted value is used to improve the model. On the other hand, offline learning means that only one set of training points is used and once the algorithm is trained then the model is used only for predictions. Usually, the sparse models are faster for prediction however

there are usually slow in learning while dense models Nearest neighbors, are fast in learning and thus can be used for online learning but are slow in prediction. Only offline learning is used for purposes of this work.

- *Generative or discriminative models*: Discriminative models are usually better for data classification, because they focus mainly on the features that distinguish the groups. On contrary, generative models are focused on the main characteristics of the datasets and are able to generate new data points with the same properties as the original set. Generative models are usually easier and thus faster to fit and they can be trained for each class separately. The advantage of the discriminative models is the possibility to use different preprocessing of the input vector $\boldsymbol{\phi}(\mathbf{x})$ [10] (sec. 2.3).

- *Fully observed or partially observed*: The training data are partially observed if it has unknown variables in some dimensions. The training on partially observed data is much more problematic. Generally, it can be done only for generative models, where the missing values are replaced by the most probable ones. However, also the discriminative models were trained on partially observed data [11].

2.2 Probabilistic Learning

This section introduce some basic methods of probabilistic learning that are used in this work. In this section, the sources [12, 13, 14, 15, 16, 17] have been used.

The input vectors are denoted $\mathbf{x}_i$ and the observed values are denoted $\mathbf{t}_i$.

The regression function that assign t_i to $\mathbf{x}_i$ can be arbitrary but because of computation reasons a combination of non-linear functions is usually searched

$$y(\mathbf{x}, \mathbf{w}) = \sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi} \tag{2.2.1}$$

The parameterised function consists of sum of non-linear basis functions $\boldsymbol{\phi}(x)$ and the parameters of model w_m are generally called *weights*. The assumptions of this form gives enough flexibility and allows to use many optimisation algorithms described in this work.

Learning machines that are based on probability classification usually use a few common approaches to learn their parameters from data.

The prevalent approach is *maximum-likelihood* (ML). The goal of the ML is to identify the most probable parameters. This is also often called the frequentist model. If the model is defined as *conditional probability distribution* (CPD) then it can be written as $P(\mathbf{y}|\boldsymbol{\Theta}) = \prod_{i=1}^{N} P(y_m|\boldsymbol{\Theta})$, where y_m are the observed values and $\boldsymbol{\Theta}$ are computed parameters. The goal is to maximize the probability of the model

$$\hat{\boldsymbol{\Theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \left(P(\mathbf{y}|\boldsymbol{\Theta})\right)$$

the log-likelihood for the observed data is

$$\hat{\boldsymbol{\Theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \sum_{m=1}^{N} \log P(y_m|\boldsymbol{\Theta}) \tag{2.2.2}$$

ML is widely used because of simple computation and also often an analytical solution exists. For example, the least squares method is the analytical solution for Gaussian distribution. The minimised error E for least-squares (ML) solution of regression is

$$E = \frac{1}{2} \sum_{n=1}^{N} \left[t_n - \sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x_n}) \right]^2 \tag{2.2.3}$$

If Φ is defined as $\Phi_{mn} = \boldsymbol{\phi}_m(\mathbf{x_n})$ then the maximum likelihood $\mathbf{w}_{\mathrm{ML}}$ is

$$\mathbf{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t} \tag{2.2.4}$$

The maximum likelihood has many interesting asymptotic properties for large number of samples. For example consistency: if the number of samples tends to infinity then the ML probability converges to the true value.

Disadvantages are problems for low number of samples or high number of free parameters. In this case, the ML can get overfitted [14]. This can be mitigated by a minor change in the eq. (2.2.2) to include a prior $P(\boldsymbol{\Theta})$ on the parameters

$$\hat{\boldsymbol{\Theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\Theta}} P(\mathbf{y}|\boldsymbol{\Theta})P(\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\Theta}} (\log P(\mathbf{y}|\boldsymbol{\Theta}) + \log P(\boldsymbol{\Theta})) \tag{2.2.5}$$

This is called *maximum a posteriori* MAP solution. This assumption can avoid the the overfitting, on the other hand, it can introduce new problems through the apriori information.

Again, for Gaussian distribution and Gaussian prior, the ML has an analytical solution. If the expected noise is Gaussian with variance σ^2 and the prior distribution over parameters $\mathbf{w}$ is

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) = \prod_{m=1}^{M} \left(\frac{\lambda}{2\pi}\right)^{1/2} \exp\left(-\lambda w_m^2/2\right) \tag{2.2.6}$$

then the MAP solution of least squares method is

$$\mathbf{w}_{\mathrm{MAP}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^T\mathbf{t} \tag{2.2.7}$$

where constant λ is called *ridge regression parameter*. This parameter must be selected according to some other knowledge, for example cross-validation, see Section 4.1.

However, MAP is still only a small step toward the full Bayesian probability. If the probability distribution over parameters and noise in data is Gaussian as in the MAP solution,

it is theoretically possible to integrate out (marginalize over) the unknown parameters $\mathbf{w}, \lambda, \sigma$ and obtain the posterior distribution from the Bayes' rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalization}} \tag{2.2.8}$$

Distribution over parameters $\mathbf{w}$ is

$$p(\mathbf{w}|\mathbf{t}, \lambda, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\lambda)}{\int p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\lambda)\mathrm{d}\mathbf{w}} \tag{2.2.9}$$

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) \tag{2.2.10}$$

then the final analytical solution of the posterior for the Gaussian distribution is [15]

$$p(\mathbf{w}|\mathbf{t}, \lambda, \sigma^2) = \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{2.2.11}$$

where the mean μ and covariance matrix $\mathbf{\Sigma}$ denote

$$\mu = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\sigma^2\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{t} \tag{2.2.12}$$

$$\mathbf{\Sigma} = \sigma^2(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\sigma^2\mathbf{I})^{-1} \tag{2.2.13}$$

It is obvious that the mean value is identical to the MAP solution because the prior over weights have no bias. The advantage is the knowledge of the whole posterior distribution over weights $\mathbf{w}$. Moreover, it is possible to apply the Bayesian rule (2.2.8) and the posterior prediction distribution can be obtained directly. This is important because usually the new predictions are the expected result

$$p(t_n^*|\lambda, \sigma^2) = \int p(t_n^*|\mathbf{w}, \sigma^2)p(\mathbf{w}|\lambda)\mathrm{d}\mathbf{w} \tag{2.2.14}$$

The prediction (2.2.14) is independent of the training data because the distribution $p(t_n^*|\mathbf{w}, \sigma^2)$ already includes the information. Solution of the previous equation can be again found in closed-form. After marginalization the distribution over the weights $\mathbf{w}$ the final prediction 2.2.14 is independent of the weight vector $\mathbf{w}$ and also the probability distribution of the predicted value.

One problem of the previous derivation is the assumption of the Gaussian distribution of the values t_n around the true function. It is not possible to find an analytical solution for other distributions. Thus it cannot be used for classification, see Section 3.3, it is only used for regression.

The previous equation can be used also to predict the distribution over the model parameters

$$p(\mathbf{w}, \lambda, \sigma|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma)p(\mathbf{w}|\lambda)p(\sigma^2)}{\int p(\mathbf{t}|\mathbf{w}, \sigma)p(\mathbf{w}|\lambda)p(\sigma^2)d\mathbf{w}d\lambda d\sigma^2}$$

but this equation usually cannot be solved analytically even for Gaussian distribution. However, numerical integration can be used, i.e the Monte-Carlo method [15].

2.3 Kernel Based Methods

Use of the kernel based learning machines can significantly improve flexibility of the algorithms. The basic idea is to use training points to improve the model.

Kernel methods are based on so called *kernel trick* that allows to use the training points as a part of the learned model and transform the feature space to wide range of nonlinear spaces without significant increase of computational complexity..

Majority of classification learning machine algorithms search for the best hyperplane separating the training/validation data. When the hyperplane is known then the classification is done by a simple substitution of the inputs $\mathbf{x}$ into the formula of the hyperplane

$$y(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b \tag{2.3.1}$$

where $y(\mathbf{x})$ is the distance from the hyperplane, $\mathbf{w}$ is the normal vector of the hyperplane, $\boldsymbol{\phi}(\mathbf{x})$ is a non-linear regular feature space mapping. The classification depends on the sign of $y(\mathbf{x})$.

This is the basic principle of all the methods used in this work: Support vector machines (SVM), Relevance vector machines (RVM), Logistic sigmoid regression (LogReg). However, there is a way how to improve this approach, if the training data points, or some subset of them, are used also in the prediction phase. These data points are called *support or relevance vectors* and each of these vectors is a center of *kernel functions*. This can be done only if the learning machine model (2.3.1) can be estimated in the *dual representation* also called the method of potentials:

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}') \tag{2.3.2}$$

This kernel is a scalar product of two mapping functions where $\mathbf{x}$ is the projected data point and vectors $\mathbf{x}'$ are the support vectors. The result is so called *linear kernel* $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$ for the simplest case of mapping $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$. The idea of the kernel methods is simple. If the learning method can be estimated using the scalar product kernel 2.3.2, then the scalar kernel can be replaced by some different type of kernel. Some basic types of kernels are in the following table

$$k(\mathbf{x}, \mathbf{x}') = x^T x' \qquad \text{Linear kernel} \tag{2.3.3}$$

$$k(\mathbf{x}, \mathbf{x}') = (x^T x' + 1)^M \qquad \text{Polynomial kernel of M-th degree} \tag{2.3.4}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(\|x - x'\|^2 / 2\sigma^2) \qquad \text{Gaussian kernel (RBF)} \tag{2.3.5}$$

The purpose of the kernel is to estimate the similarity of the vectors $\mathbf{x}, \mathbf{x}'$.

The linear kernel is a basic example of the kernel method. Suppose that the normal vector $\mathbf{w}$ from eq. (2.3.1) can be expressed as a linear combination of a subset from training points $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^T$, then

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

with non-linear mapping $\boldsymbol{\phi}(\mathbf{x})$ it can be rewritten

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}) + b$$

and finally after the kernel substitution

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$$

The advantage of the kernels is the possibility to use more general shapes of boundary between class (fig. 2.1) compared to nonkernel methods without significant increase of computational complexity [14]. On the other hand, no analytical equation for the boundary exists, except the case of the linear kernel. The knowledge of the normal (weight) vector $\mathbf{w}$ of the linear hyperplane can be used to determine dependence of the model on different features, assuming that values in different dimensions were properly normalised (see Section 4.4).

The next challenge is the choice of the kernel suitable for the training data. The selection of the kernel type and its parameters is often more important than the choice of the learning machine algorithm itself (see fig. 2.1). Typically, it is recommended to use the linear kernel first. It usually gives the most stable results and has no kernel parameters to be tuned. It also gives more stable classification in high dimensional space when the number of samples is comparable with the number of dimensions. Gaussian and polynomial kernels are more flexible; on the other hand, the flexibility can lead to overfitting. If the number of dimensions is low and number of samples is much higher than the dimensions, then the Gaussian kernel is usually used. But it is necessary to determine the correct kernel parameter σ that corresponds to the size of the kernel. Usually, cross-validation introduced in the next section is used.

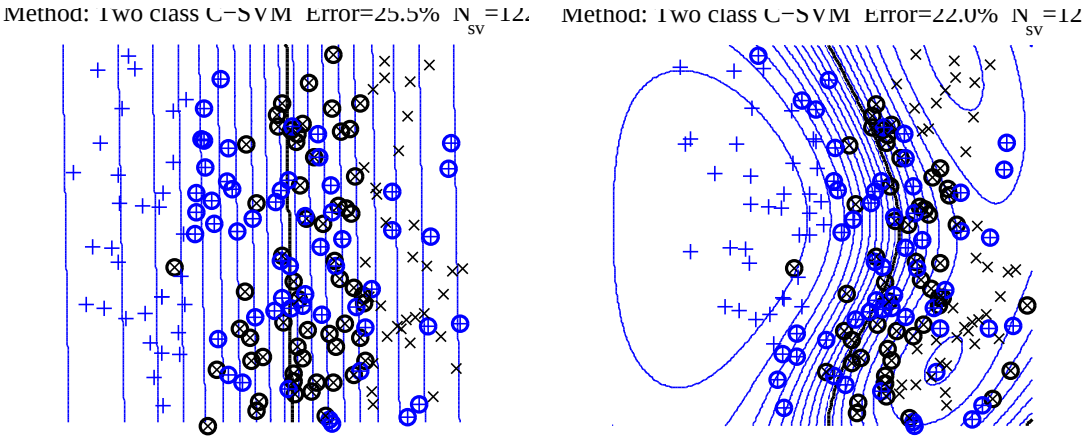


Figure 2.1: Examples of the prediction determined using the linear kernel (left) and the RBF kernel (right). Prediction was done using Support vector machines and both predictions were estimated from the same dataset. Points in circles are the support vectors.

Chapter 3

Learning Machines

This work is mainly focused on the classification algorithms that allows to estimate the probability of the result conditioned on the observed data. The knowledge of the probability can help to determine incorrect values in the database (outliers), see Section 6.9.2 for more details. Moreover, it can be used for improved model selection as is described in the Section 4.1 or used as feeding data for another advanced methods such as Bayesian networks that can connect many different prediction sources together and get some new information from the database.

However, the task of probability prediction can be ill-conditioned, for example outliers and noise in the training data can significantly influence the probability in the case of a small or informatively sparse learning dataset.

3.1 Logistic Regression

Logistic regression is a simple classification method based on data labels regression by the logistic sigmoid. It is basically a transformed ordinary linear regression

$$y(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x})\right) \tag{3.1.1}$$

The only difference is transformation of output by sigmoid function $\sigma(x) = 1/(1+\exp(x))$. The reason for the sigmoid shape can be obtained from the Bayesian theorem for the class C_1

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

where C_1, C_2 are unambiguous classes of data without intersection. After the following substitution

$$a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}$$

the probability can be rewritten

$$p(C_1|\mathbf{x}) = \frac{1}{1+\exp(a)} = \sigma(a)$$

for multiclass model, the probability can be written as

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln(p(\mathbf{x}|C_k)p(C_k))$. This function is called *softmax function.*

The logistic regression searches the probability for class C_1 of the form

$$p(C_1|\mathbf{x}) = y(\boldsymbol{\phi}(\mathbf{x})) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}))$$

However, this principle is not used only for transformation of linear regression, it commonly serves for transformation of the regression problem to the classification. The same principle can be applied on i.e. the Neural networks or the Relevance vector machines.

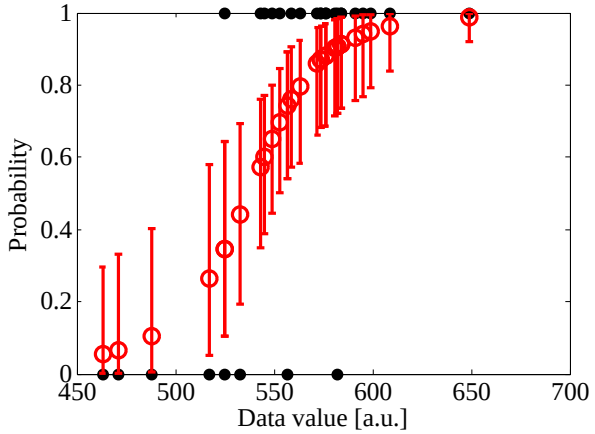


Figure 3.1: An example of the Logistic regression using the full Bayesian approach. Filled points are learning data, hollow points are predicted probability with errorbars.

The joint probability of the observed data for the target values $t \in \{0, 1\}$ and the learning machine outputs $y_n = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n}))$ is the Bernoulli distribution

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

This "regression" problem (fig. 3.1) can be optimized directly via the maximum likelihood estimation with the iterative reweighed least squares method [14, 17]. The disadvantages of this direct approach is possible overfitting if the data sets are linearly separable. In this case, the sigmoid gets infinitely steep in feature space. This can be prevented by adding some a priori assumption on the weights and finding the MAP solution.

The overfitting can be suppressed under the assumption the a prior Gaussian distribution over weights $\mathbf{w}$ is used:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$$

The prior and Bayes formula transform the maximum likelihood method to the standard optimization Maximum A Priory (MAP) method similar to the ridge regression derived in eq. (2.2.5).

In case that it is needed to use kernel, it is possible to define output y similarly to SVM machines (sec. 3.2)

Moreover, the Logistic Regression method can be improved if the kernels are used to estimate the output y, similarly to SVM machines (sec. 3.2).

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + b$$

The output of the Logistic regression is the conditional probability that the observed data $\mathbf{x}$ are in the class C_k : $p(\mathbf{x}|C_k)$. Furthermore, it is also possible to obtain precision of this prediction (fig. 3.1) using the Bayesian rule and the Laplace approximation of the probability distribution. More details can be found in [14, 17].

3.1.1 Regularization Methods for Logistic Regression

The loss function penalizing different values of model parameters – weights $\mathbf{w}$ – is used in two basic versions, so called L1-regularization and L2-regularization. These regularizes can be used generally but in this work they are used only for the Logistic regression.

- L2-regularization – ordinary least squares with the assumption that the distribution over the parameters $\mathbf{w}$ is Gaussian $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$. The L2-regularizer was already used in the introduction Section 2.2

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y_i} - \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_i})))^2 + \lambda\|\mathbf{w}\|_2^2$$

- L1-regularization – L1-norm is used for model parameters $\mathbf{w}$

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y_i} - \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_i})))^2 + \lambda\|\mathbf{w}\|_1$$

  This model has an important property: if λ is sufficiently large, some of the parameters gets equal to zero. This leads to a sparse model and the predictions are very similar to the Relevance vector machine called *Sparse logistic regression* [17].

3.2 Support Vector Machine – SVM

Support Vector Machine is a simple but powerful pattern recognition (classification) kernel based (see Section 2.3) method introduced by Vapnik [18]. SVM, basically, searches the best hyperplane that separates two data classes. The SVM is a kernel based method

that searches for a hyperplane that separates the data and the data with the highest distance from the hyperplane (boundary). The use of kernels allows the find a very flexible boundary between the groups thanks to the non-linear mapping of the hyperplane to the feature space. The main disadvantage of SVM are non-probabilistic predictions. Although it is possible to add a guess of the prediction probability [14, 19, 20], the probability estimations can be poor. This will be discussed later in the Section 3.2.1.

The maximization of the distance from the boundary can be reformulated as the maximization of the term $1/\|\mathbf{w}\|$ which is equivalent to minimizing $\|\mathbf{w}\|^2$. If we denote set of N target values $\{t_i\} \in \{-1, 1\}$, $i \in 1, \ldots, N$ and corresponding input vectors $\{\mathbf{x}_i\}$, it results to following quadratic optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.2.1}$$

with constraints

$$t_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, ..., N \tag{3.2.2}$$

It can be improved to allow – with some penalty – data points overlapping the boundary. This principle is called *soft margins*. New *slack* variables ξ_i are introduced to measure the overlapping. C is a new constant that controls the trade-off between data overlapping the boundary penalty and the size of margin. Moreover, the trade-off can be selected individually for each point if weights $\mathbf{w} > 0$ are used.

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} w_i\xi_i \tag{3.2.3}$$

with constraints

$$t_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N, \quad \xi_i \geq 0 \tag{3.2.4}$$

These equations can not be optimized directly if the kernel representation is used. The equation must be transformed to its dual representation:

$$L(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

where L is the minimized Lagrangian, a_n are the searched dual parameters, t_n are the target values normalized to the set $\{-1; 1\}$ and $k(\mathbf{x}_n, \mathbf{x}_m)$ is the value of the kernel. The Lagrangian L is minimized with respect to the transformed constrains

$$0 \leq a_n \leq w_n C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

The prediction output of the SVM algorithm is proportional to the distance from the boundary

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) \tag{3.2.5}$$

Only vectors $\mathbf{x}_n$ with nonzero parameters a_n are used for the prediction and these vectors are called support vectors. Support vectors are always points at the boundary or exceeding the boundary. The rest of the points is ignored after the learning phase, therefore it leads to a sparse model.

The SVM algorithm is not based on a probabilistic model, therefore the only information about the uncertainty of the prediction is the distance from the hyperplane. However, the distance is normalized to one for the support vectors laying on the boundary, hence the absolute value of the distance from the boundary does not directly corresponds to the uncertainty of the prediction.

LibSVM algorithm [21] was used in this work because of a good training speed and sufficient flexibility. The algorithm is implemented in C++ and contains many optimization improvements in order to improve speed and limit the memory usage.

3.2.1 Different Precision of Data Points

The original version of the SVM algorithm presumes that all points and also all classes have the same weight, precision and classes have similar number of points. However, in practice it is not often satisfied.

In the case that the training classes are significantly unbalanced, the learning machine algorithm often completely ignore the smaller group.

The simplest solution is to choose different weights C_i eq. (3.2.3) for each class. The weights w_i can be selected as i.e inverse number of data points in classes. It results to a worse total classification score but a better score for the smaller group. A disadvantage can be significantly worse convergence of SVM algorithm (LinearSVM).

Another common problem is the different precisions (errorbars) for each data point. One solution is to use different weights C_i for each point. The weights can be selected as inverse points precision.

The main advantage of this solution is the fact that the weighting is fully supported by the LIBSVM library. The disadvantage is that the algorithm ignores spatial distribution of the data point with errorbars.

The best solution is to use the original data from which the errorbars were obtained. The SVM algorithm can process all the data and get more realistic results.

In the case that the original data are not accessible or only the probabilistic distribution of the points is known, another approach can be used. Each probabilistic distribution can be approximated by a random set of points. For the Gaussian distribution, the set of the points corresponds to the position original point and variance is set according to the errorbars. For high numbers of the artificial points, the properties converge to the expected behavior (see Fig 3.2, 3.3). As result, the boundary correctly reflects the data distribution. Disadvantage of this approach is significant increase in the number of the training points that leads to the increased computational time and memory demands.

Moreover, the spatial distribution is negligible for high number of the training points.

3.2.2 Confidence Region

One of the problems connected to the probability estimation is determination of the confidence interval or "novelty regions" around the boundary. These are the regions where the predictions are not reliable and more values is needed.

In the case of the linear kernel, the exact equation of the boundary hyperplane can be estimated. However, in the case of the other kernels, the boundary can be estimated only by appropriate root searching algorithm and the analytical equation is unknown.

The confidence intervals cannot be determined directly from the SVM output. Instead, a method based on fitting the SVM outputs with the logistic sigmoid was proposed in [19]

$$p(x|C_1) = \sigma(Ay(\mathbf{x}) + B)$$

where $y(\mathbf{x})$ is the distance from the boundary (the output of the trained SVM algorithm) and the values of the parameters A, B are found by cross-entropy method. The data used for the sigmoid fitting should be independent of the data used for learning in order to avoid severe overfitting. The advantage of the algorithm is good implementation in LIBSVM [20]. LIBSVM uses 5-fold crossvalidation instead of the new independent data to train the sigmoid, therefore if the probability prediction is used inside another cross-validation (i.e for optimal parameters selection), it can significantly delay the training phase. Moreover, the SVM algorithm is not designed for probability estimation, therefore the results can be poor [14].

If the probability estimation is known, it is straightforward to identify the uncertain (novelty) points inside the probability range [threshold; 1-threshold], however the best threshold value must be selected artificially. Examples of the results for different size of errorbars are in figures 3.2, 3.3.

3.2.3 One Class SVM

Although the SVM algorithm is essentially two class algorithm, Scholkopf proposed [22] a modification for one class only. This method is usually used for identification of outliers or novelty detection. The basic idea of this algorithm is to find a hyperplane that separates majority of data points from the outliers such that the of a training point getting beyond boundary is equal to ν. It can be done, if the origin after kernel transformation is threated as the only member of the second class and than the standard ν-SVM [23] algorithm is used. The result will be that majority of the points will be inside of the border and only minority of points will lay outside.

The following quadratic optimization is used to separate the data set from the origin.

$$min_{\mathbf{w},\boldsymbol{\xi},\rho} \frac{1}{2}\|\mathbf{w}\| + \frac{1}{\nu l}\sum_i \xi_i - \rho$$

with constrains

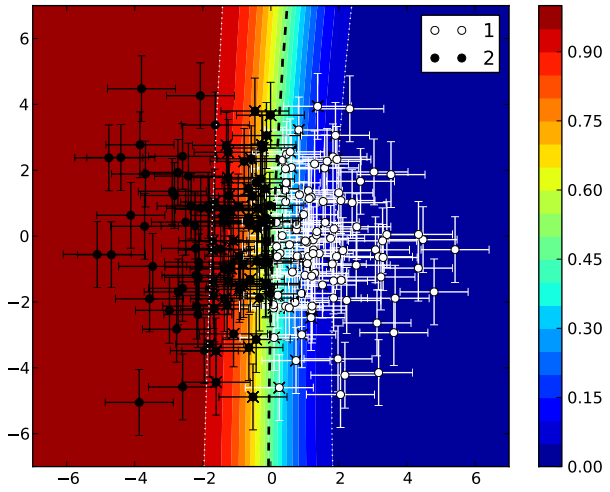


Figure 3.2: An example of the border and probability distribution for the binary classification. The training data set was generated with class boundary $x = 0$ and each point was described by points with the Gaussian distribution with variance $\sigma^2 = 1$ (errorbars).

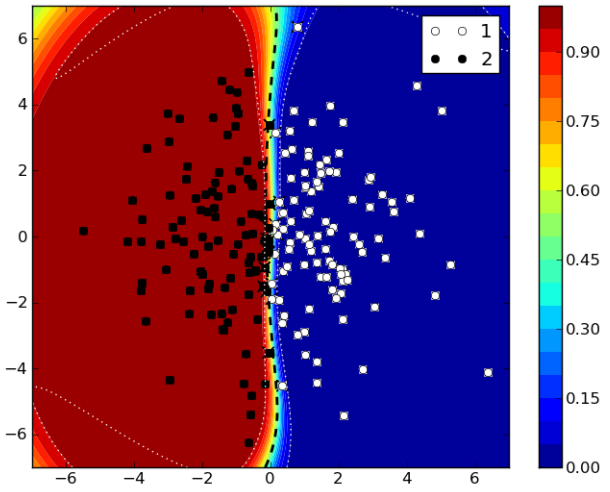


Figure 3.3: An example of the border and probability distribution for the binary classification. The training data set was generated with class boundary $x = 0$ and each point was described by points with the Gaussian distribution with variance $\sigma^2 = 0.01$ (errorbars).

$$\mathbf{w}\mathbf{x}_i \geq \rho - \xi_i, \quad \xi_i \geq 0$$

this quadratic problem is then transferred to dual problem in the same way as the SVM. The final decision rule (distance from boundary) is

$$f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho$$

When the parameter ν goes to zero, the boundary should behave as *hard margin*, therefore no outliers on training set should be allowed.

The main advantage of this algorithm is possibility to a use only one class for training or a very limited size of the second class for cross-validation to estimate free parameters of the model: ν and kernel parameters. However, in case of plasma disruptions the results depends only weakly for the ν therefore, only the kernel parameter must be estimated moreover the added threshold mentioned in the last section, is another free parameter.

In this paper, the implementation of One Class SVM from the LibSVM library [21] was used.

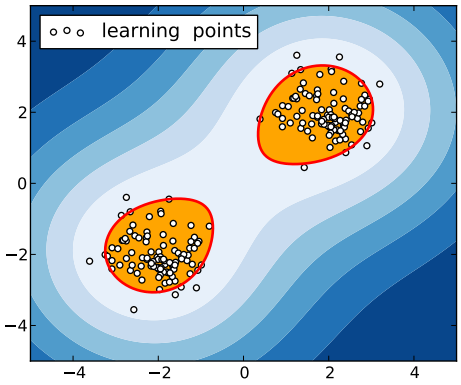


Figure 3.4: An example of the One Class SVM prediction for two Gaussians. Kernel parameter $\gamma = 0.1$ and ratio of outliers is ν=0.1

3.3 Relevance Vector Machine – RVM

In spite of state-of-the-art results in many tasks, where SVM was used, this algorithm suffers from several disadvantages.

- The first disadvantage is that the predictions are not probabilistic. Although some attempts to add an estimation of the prediction probability exist (see Section 3.2.2 or [14, 19, 20]), the results can be often imprecise.
- SVM algorithm uses unnecessarily high number of the support vectors and although the resulting model is sparse, number of the support vectors can be a significant fraction of the training set. Moreover, the number of the SV usually grows linearly with the number of the training points, although some postprocessing method exist that can decrease number of SV [24].
- No straightforward and universal way exists, how to determine C and kernel parameters for a nonlinear kernel.
- The kernel function $K(\mathbf{x}_i, \mathbf{x})$ must satisfy the Mercer's condition [25].

Relevance Vector Machine algorithm was introduced by Tipping [12] to obtain some advantages of the Support Vector Machine (see Section 3.2) algorithm such as the sparse

model, kernels and to remove the disadvantages such as the native non-probability output and additional model parameters C or ν. Another advantage is the sparsity because the number of complexity of the model is not growing linearly with the number of training points.

The derivation of the RVM can be found in [12] but an introduction to the problem is in this section. The derivation is based on the probabilistic regression introduced in Section 2.2, but some steps and assumptions are different.

The final equation for the prediction is identical to the SVM (eq. 3.2.5).

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{M} w_m K(\mathbf{x}_i, \mathbf{x}) + b \tag{3.3.1}$$

where $\mathbf{w}$ is the vector of weights and b is the intercept.

However, the Bernoulli distribution should be adopted to obtain the probabilistic prediction $p(t|\mathbf{x})$ because only values 0 and 1 are possible. Therefore, the logistic sigmoid function $\sigma(y) = 1/(1+e^{-y})$ must be applied to transform the regression 3.3.1 that is linear in coefficients to a fully nonlinear probabilistic function. According to the definition of the Bernoulli distribution, the conditional probability for N points under assumption of known $\mathbf{w}$ can be written as

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(x_n, \mathbf{w})\}^{t_n} [1 - \sigma\{y(x_n, \mathbf{w})\}]^{1-t_n}$$

for targets $t \in \{0, 1\}$. Maximum likelihood estimation of weights leads to overfitting. Tipping suggested to introduce prior constrains on the weights $\mathbf{w}$ penalizing the complexity of the model. Following prior distribution in combination with the the Bayes rule was introduced to avoid overfitting:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} \sqrt{\frac{\alpha_i}{2\pi}} \exp\left(-\frac{\alpha_i w_i^2}{2}\right)$$

where $\boldsymbol{\alpha}$ are hyperparameters that controls the distribution of the associated weights $\mathbf{w}$. This separated hyperparameters for each weight are the most important property of the RVM algorithm.

Given the prior, new predictions are made using so called predictive distribution:

$$p(t^*|\mathbf{t}) = \int p(t^*|\mathbf{w}, \boldsymbol{\alpha}) p(\mathbf{w}, \boldsymbol{\alpha}|\mathbf{t}) d\mathbf{w} d\boldsymbol{\alpha} \tag{3.3.2}$$

However, as for many Bayes methods, this integral cannot be solved analytically. The posterior $p(\mathbf{w}, \boldsymbol{\alpha}|\mathbf{t})$ is not known, instead it can be decomposed as

$$p(\mathbf{w}, \boldsymbol{\alpha}|\mathbf{t}) = p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\mathbf{t})$$

The posterior over the weights is according to the Bayes rule equal to

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) = \frac{p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\boldsymbol{\alpha})}{\int p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\boldsymbol{\alpha}) \mathrm{d}\mathbf{w}} \tag{3.3.3}$$

the integral in the denominator of the previous equation has no analytical solution for the Bernoulli distribution, therefore the Laplace approximation should be applied.

The second term in eq.(3.3.2) $p(\boldsymbol{\alpha}|\mathbf{t})$ is approximated by the most probable point estimated values. The distribution $p(\boldsymbol{\alpha}|\mathbf{t})$ is therefore approximated by $\delta(\boldsymbol{\alpha}_{MP})$. The hyperparameters α_i are obtained from the type II maximum likelihood method as is shown in [15].

During the optimization, most of the α_i gets large and the corresponding weights $\mathbf{w}$ are almost infinitely peaked around zero and because the corresponding kernel are pruned, the models becomes to be sparse. This is called ARD prior – Automatic Relevance Determination. The remaining vectors are called Relevance vectors and ,on the contrary to SVM, these vectors are the most representative points of both groups. The optimization continues till the change in α is below certain threshold or maximum number of iteration is reached.

Despite the described advantages of RVM, this algorithm suffers from several disadvantages. Firstly, the RVM optimization can often reach a local maximum on contrary to the SVM optimization where the global optimum is always guaranteed because the optimized function is convex. This can lead to a significantly worse model, however it is possible to detect these corrupted models and remove them. Other disadvantage is the computational complexity and high memory demands. The memory limitation restricts the training set to less than 10000 points for large kernel sizes. The LibSVM algorithm can be effectively computed using more than 60000 points thanks to implemented heuristic algorithms and caching [21].

In this work, the implementation of RVM for MatLab was used – SparseBayes 2.0 [16] with a few speed and memory usage improvements based on the sparse matrices.

3.4 Testing

The algorithms described in the previous section are compared in the following figures. Note that all the predictions are very similar. It is not so obvious, because the principle of all algorithms is quite different:

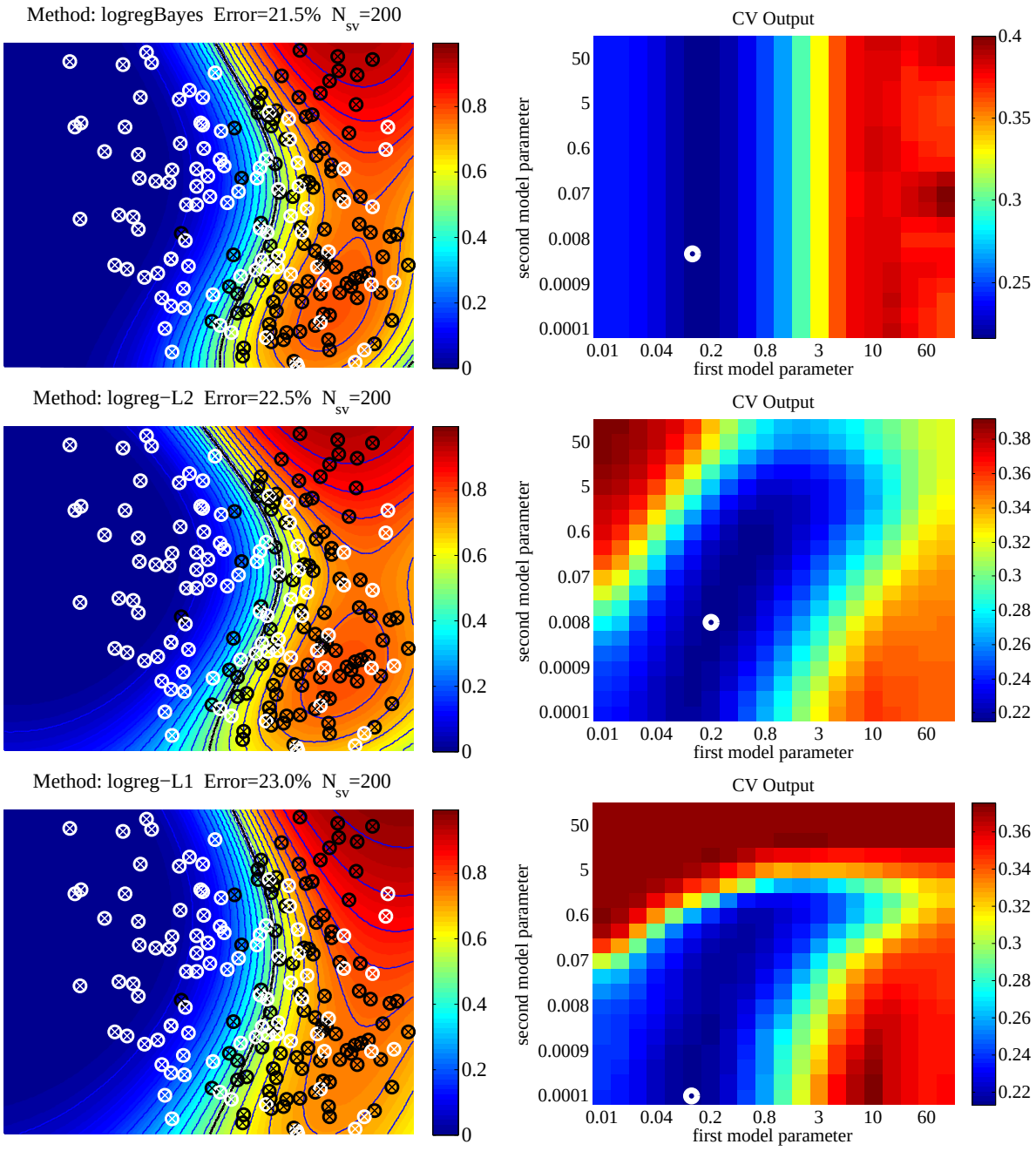
- LogReg–Bayes, LogReg–L2 – dense models
- LogReg-Bayes, RVM use full Bayesian probability
- SVM sparse and maximize margin between the classes

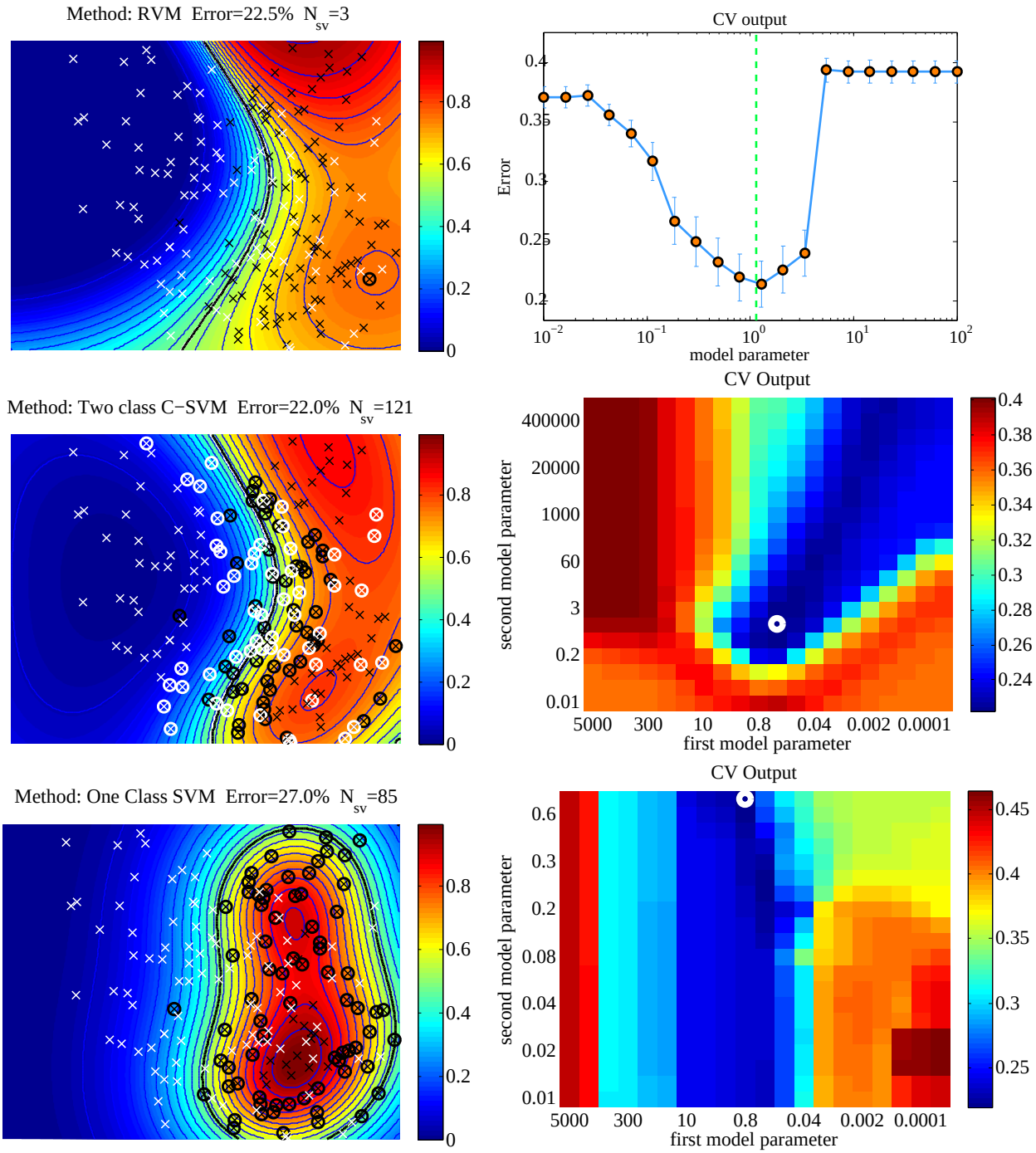
Moreover, the optimal kernel sizes for different algorithms oscillates from 0.05 to 1.3. And even quite a small change in the parameters significantly changes the resulting probability although the classification rate stays almost the same.

Testing data were loaded from `http://research.microsoft.com/~cmbishop/PRML/webdatasets/datasets.htm`. Data were generated from a mixture of 3 Gaussians. Points in circles are support/relevance vectors. The corresponding Gaussian mixture model had the parameters:

```
mix.priors = [0.5 0.25 0.25];
mix.centres = [0 -0.1; 1 1; 1 -1];
mix.covars(:,:,1) = [0.625 -0.2165; -0.2165 0.875];
mix.covars(:,:,2) = [0.2241 -0.1368; -0.1368 0.9759];
mix.covars(:,:,3) = [0.2375 0.1516; 0.1516 0.4125];
```

Examples of predictions (left) and CV outputs (right) using different learning machines
Note that the resulting probability predictions (left) are very similar for all methods
although the principles of used methods are different




Method: RVM Error=22.5% N_sv=3
CV output
Error
model parameter
Method: Two class C−SVM Error=22.0% N_sv=121
CV Output
second model parameter
first model parameter
Method: One Class SVM Error=27.0% N_sv=85
CV Output
second model parameter
first model parameter

Chapter 4

Model Selection

The direct approach to the model selection is a choice of the model that gives the best classification score on the training data according to some loss function. Unfortunately, this approach does not work, the most complex model is always chosen, for example a polynomial of the highest degree in regression. This selected model has low predictive value it is so called overfitting. Two basic ways, how to avoid this problem, exist: the Bayesian and the non-Bayesian.

4.1 Cross-Validation – CV

The crossvalidation (CV) is a non-Bayesian (frequentist) statistical method how to avoid the overfitting caused by the choice of model parameters [26]. The aim of the crossvalidation is to generalize the estimated results for an independent data set. Bayesian methods should be able to choose the parameters automatically [12]. However, it is often possible only for some special cases such as the regression (see Section 2.2). So even for the Bayesian methods the kernel parameters must be determined by the CV. Also the choice of the correct kernel is either based on some a priori knowledge or the kernel with the best results in the CV is used.

CV is basically similar to the concept of training and validation on separated data set, however it can use more information from the data.

The standard way to perform crossvalidation is following: training data are randomly separated to N–folds, usually to 5–folds [17], $N-1$ folds are used for training and the last one is used for testing, however many other types of CV exists. The training process is repeated N-times for different combinations of learning/testing folds. The result of the CV is mean classification score for the given parameters. Moreover, it is possible to estimate the variance of the CV and use so called "S1-rule" [17]in the case of the flat bottom (fig 4.1) of the loss function. According to the S1-rule, the most optimal parameters have the score higher than the best score minus the standard deviation. Futher, there is used some a priori knowledge to selected the best one. For example, the least complex model can be preferred.

A common way to find optimal solution is to use a grid search in the N_p-dimensional

space, where N_p is the number of the free parameters and estimate the parameters that give the best prediction. An optimization algorithm can be used instead of the grid search. However, there are issues with the local minimums and also the grid search can be easily parallelized to increase the computational speed.

The problem of the CV is that it is a non-probabilistic (non Bayesian) method and all the introduced learning machines compute the output probability under the assumption that the used kernel parameter is correct. Therefore, the probability prediction can be significantly different for different kernels and their parameters (see Fig 4.3).

In this work, the apriori knowledge used for the S1-rule is that the larger Gaussian kernel size σ or the lower number of the support vectors usually means a less complex model. This knowledge generally leads to more reliable predictions.

The second issue of the CV is connected to the way how the CV score is computed. So called *loss functions* are used to success rate of the models, usually "zero-one" loss function is used. It returns the mean number of the misfitted points

$$L = \frac{1}{N} \sum_{y_i \neq \tilde{y}_i} 1$$

where $\tilde{y}_i$ is the predicted value and y_i is the expected value. If the sizes of the classified groups are not similar, the loss function should be normalized in order to prefer the smaller group

$$L = \sum_{i \in Y} \sum_{j=1}^{N} \frac{y_{ij} \neq \tilde{y}_{ij}}{N_i}$$

The disadvantage of this loss function is that it gives no importance to the probability only to the class selection. Therefore, the resulting model can prefer all the predictions close to 50% (see the binary classification in the Fig. 4.3). Better results can be obtained using function that penalizes predictions regard to the probability:

$$L = \sum_{y_i \neq \tilde{y}_i} P(x_i|C_1) + \sum_{y_i = \tilde{y}_i} (1 - P(x_i|C_1)) \tag{4.1.1}$$

where $P(x_i|C_1)$ is the conditional probability that the data point x_i belongs to the first group for binary classification. This loss function results in a less flat shape of the CV curve compared to the standard zero-one loss function (Fig. 4.1). Moreover, this probabilistic loss function uses more information from the system compared to ordinary zero-one loss function. The main difference is expected for small datasets as was already tested for a similar loss function in [27].

4.2 Bayesian Model Selection

The cross-validation has several major drawbacks. The first and the most obvious is the necessity to repeat the model training several times to find the best parameters. Moreover,

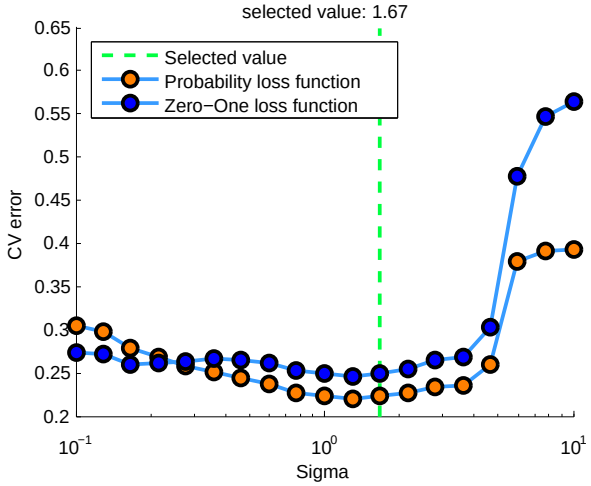


Figure 4.1: An example of the CV tuning of the kernel size for the Relevance vector machine. The "S1-rule" was used for selection of the optimal value. The blue points correspond to the ordinary zero-one loss function, the red points to the probability loss function. (4.1.1).

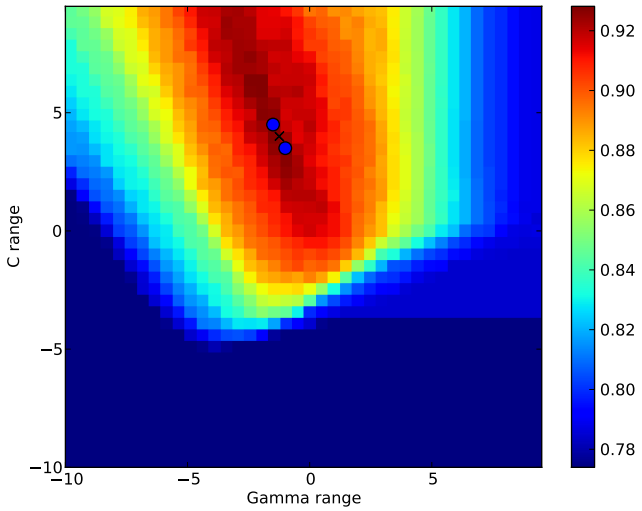


Figure 4.2: Another example of the CV tuning of the kernel parameters (C and Gamma) for SVM algorithm. The used loss function is $L = \frac{1}{N} \sum_{y_i \neq \tilde{y}_i} 1$. The best values are signed by blue circles.

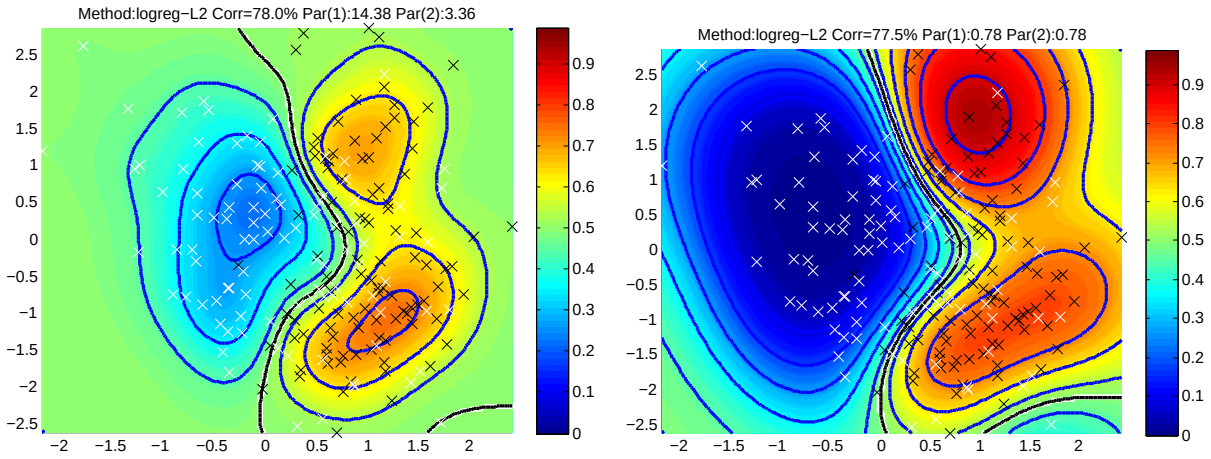


Figure 4.3: Examples of the different predictions for the optimal parameters selected according to the zero-one loss function (left) and the probability loss function (right). L2 Logistic regression was used for the prediction. Note that the boundary is similar but the probability values are significantly different

the grid size grows exponentially with the number of parameters. The use of the grid also implies that the searched parameters are distributed to a finite number of values, although the logarithmic scale is usually used. Another issue of the CV is the estimation of the uncertainty for the selected parameters. It means that the probability estimations given by the learning machines are based on the assumption that the parameters learned by the CV are the only correct values.

A better approach is to use the Bayesian rule (2.2.8) and integrate out the model parameters $\mathbf{\Theta}$ as it is shown in Section 2.2. The total probability $p(y|y_0, C_k)$ that the vector $\mathbf{x}$ is in the class C_k and y_0 denotes labels the training data, is generally

$$p(y|y_0, C_k) = \int p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)d\mathbf{\Theta}$$

The term $p(y|y_0, C_k)$ is the marginal likelihood or sometimes also called *evidence* for the class C_k.

The probability over the searched parameters can be estimated directly from the Bayes rule

$$p(\mathbf{\Theta}|y_0, C_k) = \frac{p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)}{\int p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)d\mathbf{\Theta}}$$

It is also possible to optimize the likelihood $p(\mathbf{\Theta}|y_0, C_k)$ numerically.

In the case that the distribution $p(\mathbf{\Theta}|y_0, C_k)$ can be solved analytically over some set of parameters $\mathbf{\Theta}_i$ while for other parameters, for example kernel size σ, cannot be directly solved, then it is possible to approximate the kernel parameters by its maximum likelihood value and use only the distribution over the rest of the parameters. This approach is called type-II maximum likelihood approximation and is used for the RVM regression [15] (see Section 3.3).

The great advantage of the Bayesian method is the possibility to integrate out the model parameters and thus automatically avoid overfitting. It follows the principle of the Occam's razor.

4.3 Dimension Reduction

The dimensionality reduction is an important step for the machine learning. It is important to note that even a lot of the data does not guarantee that the overfitting will be avoided. The amount of the data must be measured relatively to the number of free parameters in the model and also the amount of the information contained in the data. The number of the free parameters grows with the dimensionality of the problem and also with the choice of the kernel. The Gaussian kernel (RBF) is very flexible therefore it provides a huge almost unlimited degree of freedom. The linear kernel is less flexible but still, in the case of high dimensionality or insufficiency/sparsity of the training data: It can lead to the overfitting. Another advantage of the dimensionality reduction is possibly better understanding to the physical context if only the most important variables are selected.

It is possible to perform "variables pruning" from a priori knowledge or via learning machines. Many different algorithms were developed to provide some feature ranking and elimination [28].

A basic way is to use a single variable classifier and thus to classify the individual predictive power of each dimension separately and choose dimensions with the best score. This method is called *univariate feature selection*. It statistically rank differences in distribution using ANOVA filter[1] and identifies the most relevant dimensions; the results are similar to the linear separation [29, 30]. The advantage of this attitude is a good theoretical background, on the other hand, it works only for special cases when the data are easily separable and number of points are low, on the other hand, this method is very quick.

Another more advanced method is to separate the data with a linear hyperplane. In the case that the data are correctly normalized then the dimensions with the lower absolute weight $\mathbf{w}$ are less relevant for the linear separation. This can be used to remove the least successful dimensions. The disadvantage is the expectation of linear separability. The advantage is quite fast feature selection compared to i.e. wrappers described in the following text.

Some learning methods can be improved to automatically remove the irrelevant dimensions. For example, if the l_0-norm or l_1-norm for the weights $\mathbf{w}$ is used in the MAP solution without the kernels use. The l_0-norm prefers low number of non-zero weights. The number of the variables ($N_p = \|\mathbf{w}\|_0$) can be added as the next regularization term to the optimized error size. But it needs radical changes in the learning machine algorithm. In work [31] a simple method was proposed to solve l_0-norm approximately for the SVM algorithm. The process is the following:

1. Train linear SVM
2. Rescale the input data with their weights $\mathbf{w}$
3. Repeat until convergence

This procedure prunes the variables approximately similarly to the l_0-norm.

[1] Basic statistical analysis of variance

Another way was proposed in [27] where the SVM with Gaussian kernel was tested and each dimension used different kernel size. The optimal kernel sizes were identified iteratively using CV with non-grid optimization. Finally, the dimensions with the smallest kernel size are the most important.

4.3.1 Wrappers

Wrappers denote a class of powerful methods for features elimination regardless the used learning algorithm. Wrappers are algorithms searching the optimal set of variables in feature space and propose different combinations of variables according to some loss function. This loss functions is usually some learning machine algorithm trained on the data proposed by the wrapper and returns a number corresponding to the mean loss of the testing phase. The learning algorithm serves as a black box and thus it is easy to implement such a wrapper for many different learning machine algorithms. Moreover, the wrappers use the predictive performance of the learning machines or other ranking algorithms such as ANOVA filter [28] and only need to know how to interpret the results and how to search in space of all possible subsets of variables. The problem is that checking all subsets combinations is NP-hard[2]. This cannot be solved for larger datasets using a brute-force method and thus many strategies were developed [29].

However, the most often the forward feature selection (FS) and the recursive feature elimination (RFE) algorithms are used. The basic principle of the forward selection is selection of the features according to their predictive power. Only the most important features are added to the training subset. On the contrary, the backward elimination algorithm starts with all features and removes dimensions with the smallest additional information. Advantage of the backward elimination is better resistance to the variable correlation, on the other hand this algorithm is more computationally demanding.

These algorithms can search in possible subsets of variables, but it is also possible to rank each dimension (variable) separately according their influence to the total classification score, i.e use linear weights, and remove only the least significant variable.

Great advantage of the wrappers is their universality and also, compared to the previous methods, possibility to deal with non-linear feature dependencies if they are supported by the used learning machine. Moreover, it is possible to handle multi-class problems easily. The disadvantage are generally higher computational demands..

4.3.2 Embedded Methods

Some methods have built-in selection of the most relevant features. One example is decision trees algorithm – CART (Classification And Regression Tree). Tree methods need to decide which feature split the data best and sort layers in the tree according to this. The main disadvantage of the CART method is that it rank the classification performance separately for each dimension.

[2]the complexity is higher than polynomial

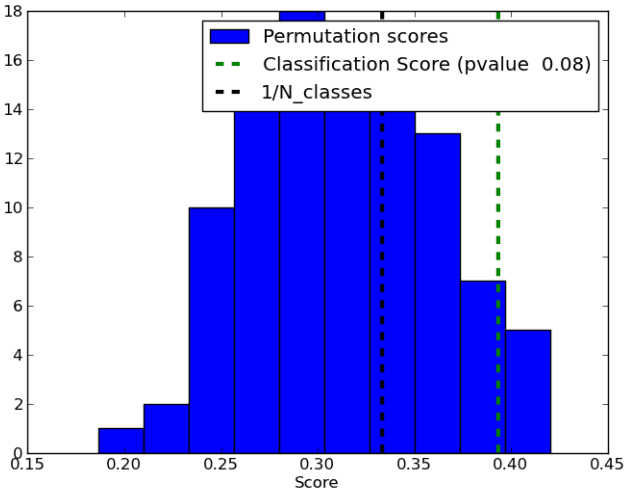


Figure 4.4: An example of a relevance histogram determination using the random permutations of the labels. The maximum of the histogram denotes the median p-value, 1/N line denotes random guess (for 3 classes), and Classification Score line denotes p-value for our selection

4.3.3 Model Validation

The last step of the variable elimination is checking if the number of the selected dimensions is optimal or more dimensions should be removed. If the pruning leads to better predictive results (classification score) then the algorithm should continue. But even if the pruning decrease predictive accuracy it still can improve generalization of the predictions and remove overfitting.

One way is to repeatedly randomly permute labels of the data and perform classification. The statistical p-value of the original classification is equal to percentage of random permutations that have higher classification score then the score of the original data. The final p-value should be almost zero (fig. 4.4) This approach can give only the upper limit for the number of dimensions and also it can be used only for smaller data sets because of high demand on computing performance.

The classification success also must be compared relatively to random selection. It means that classification score 60% for binary classification is worse than 60% for ten classes classification.

The next problem that must be considered is the correlation between dimensions. It is possible that two dimensions has low predictive power itself, but together the score can be significantly better. It is possible the the forward feature selection would not select these dimensions.

One solution is use of the correlation analysis to detect these redundant dimensions. The second way is to use another elimination algorithm for example the recursive feature elimination that should completely ignore this problem.

Another interesting method of the model validation uses artificial “fake” dimensions with

random Gaussian noise. These dimensions in the training data set are used as a probe. If weight of some "real" variable is lower or similar to weights of the probes then this dimension can be removed. This method can be further improved if a random permutation of a real variable is used as the fake variable instead of the Gaussian noise. On the other hand, the probles significantly deteriorate the convergence of the algorithm.

4.4 Scaling

Kernel learning algorithms are not generally independent of scaling and translation. Some kernels are invariant to the translation, i.e. $k(x, x') = c(x - x')$, some kernels depend only on the magnitude (e.g. *radial basis functions* – RBF). The linear kernel parameters are not invariant to the translation or scaling, but the prediction stays the same, if all dimensions are scaled or shifted with the same number. RBF kernel is invariant to the translation and scaling in the case that the kernel parameter σ is scaled too as it is denoted in following equation:

$$k(a\mathbf{x} + b, a\mathbf{x}' + b) = \exp(-a^2\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$$

The size of the kernel is the only element that depends directly on the scale of inputs x, the rest of the SVM algorithm is independent. However, this is valid only if the scaling constant a is the same for all dimensions.

It is commonly recommended [14] to rescale the data to be more similar in the training dimensions. However, it is important to normalize the training and the testing data in the same way. Either normalization in form $(x_i - \max(x))/(\max(x) - \min(x))$ or normalization to reach zero mean and standard deviation equal one is usually used. Latter normalization is slightly more robust to the outliers. In order to avoid problems with outliers, it is possible to use a robust statistics such as the median instead of the mean and the median absolute deviation instead of the standard deviation. The final normalization would be

$$\tilde{X}_k = \frac{X_k - \text{median}_i(X_i)}{\text{median}_i(\|X_i - \text{median}_j(X_j)\|)} \tag{4.4.1}$$

If the data have heavy tails, it is also possible to renormalize the quantiles of the data as was proposed in Chapter 6.

The purpose of the scaling for the linear kernel is mainly to reach the same importance for all dimensions. On the contrary, for the RBF kernel the aim is to achieve similar distance between patterns in the data points for each dimensions.

4.5 Data Preprocessing

Creation of a good data representation is the art of the learning machines. All learning machines algorithms described in this work expects data properties as the amplitude,

therefore all the important features must be converted to the magnitude. It is also possible to include an a priori knowledge to the feature extraction, i.e. extract frequency via the Wavelets or Fourier transformation, size of derivation, variation of data or many others features. Also removing of the outliers from data and performing data smoothing in the case of time dependence data can improve the prediction.

It is also possible to use some generic feature extraction methods such as clustering or singular value decomposition to increase the orthogonality of the data features.

4.6 Toolboxes

Many algorithms described in the previous sections are already implemented in open-source toolboxes. Five toolboxes were used in this work: PMTK3, scikits.learn, SparseBayes, pMatlab and standard statistic toolbox from MatLab.

PMTK3 is an open-source toolbox developed by K.P.Murphy and it is available for MatLab and Octave. PMTK3 implements many learning machine algorithms and also includes other toolboxes as plugins. The number of supported algorithms is impressive. The most important are: Logistic regression L1, L2 type and implementation of Logistic regression with Bayesian probability, Linear regression, K-th nearest neighbor, Neural networks, naive Bayes. But the most important are Support vector machine implemented in LIBSVM, LinearSVM and other SVM implementations, the second important plugin is Relevance vector machine algorithm based on the SparseBayes toolbox implemented by Tipping [12].

Futher, the scikits.learn package is an open-source toolbox in Python. Among others it implements support vector machines (LIBSVM). The advantage, compared to the PMTK3 toolbox, are implementations of many feature selection algorithms and also a better support of the multiprocessing.

Finally, due to high computation demands of learning machine algorithms, pMatlab toolbox [32] was used to parallelize the computations over more processors and computers. Also LIBSVM was used with OpenML library that allows parallelization but only for one computer.

Chapter 5

Breakdown prediction on tokamak GOLEM

The learning machines and other algorithms described in the previous sections were applied on data set based on database of shots from tokamak Golem **??** but some algorithms for the feature elimination were used only for the SVM algorithm.

The main aim of this section is to test methods introduced in the previous sections on a relatively small data set with simply comparable results and quite with well known physical background. This simple model shows problems described in previous sections such as many unimportant dimensions, hidden (unknown) dimensions such as amount of plasma impurities and outliers – machine failure.

Moreover, the learning machines could be applied directly without any problems with evaluation of score that will be in the next Chapter 6 about prediction of disruptions in tokamaks.

The Golem database was used to determine probability of the plasma breakdown based on parameters that were set up before the shot. These available parameters are

Shortcut	Description	Range
H2	Enabled H2 filling	0,1
U_b	charge of capacitors of magnetic field	0-800 V
U_{cd}	charge of capacitors of current drive	0-1200 V
U_{bd}	charge of capacitors of breakdown field	0-500 V
U_{st}	charge of capacitors of vertical stabilisation field	0-500 V
P	pressure of H2	0-250 mPa
T_{cd}	delay of current drive trigger to main trigger	0-10 ms
T_{bd}	delay of breakdown trigger to main trigger	0-10 ms
T_{st}	delay of stabilisation field trigger to main trigger	0-10 ms
PreIonisation	Preionisation electrode	0,1

5.1 Preprocessing

The raw data had to be slightly preprocessed in order to improve prediction and add a priori knowledge. The first problem is a high number of the dimensions and their correlations. It is important to remove less important dimensions to avoid overfitting. Here, it could be done from a priori knowledge. The first property $H2$ filling is necessary condition to make breakdown thus it can be removed and also all shots when there was reached breakdown without the filling gas, because it is only a minor unimportant group – outliers.

The next step was to join some dimensions to ensure the independence of variables. In this case U_b plus T_{cd} or U_b plus T_{bd} gives magnetic field in time of maximal current drive or breakdown field. Furthermore, instead of absolute values of T_{cd} and T_{bd} only their time shift is important, thus only the difference should be used.

Finally, the gas pressure was transformed using the logarithmic substitution in order to allow faster changes near to the "vacuum" pressure – $\approx 5\,\mathrm{mPa}$.

The results of the SVM algorithm with RBF kernel before this feature elimination using a priori knowledge are:

Group	Precision	Recall	F1-score	CV score	N_{SV}	N_{VEC}
0	0.834	0.627	0.648	0.338	265	367
1	0.885	0.990	0.467	0.786	230	1064
Total:	0.858					
Total SV:	34.591%					
Best parameters:	C=1000	G=0.005				

and after the these changes were the results

Group	Precision	Recall	F1-score	N_{SV}
0	0.93	0.67	0.78	302
1	0.91	0.99	0.95	1034
avg / total	0.92	0.91	0.91	1336

It is important to notice two things: firstly, the dimensions cut-off can lead to worse results in the total precision. This was expected and it even quite successful result when the dimensionality of the problem was decreased by 40% and the prediction success rests almost the same. The second issue is a common property of the described learning algorithms: they usually give worse results to the smaller class. In this case, it was group 0 (*no breakdown*), because breakdowns were usually requested by operators. The number of shots with breakdown is 1034 and shots without breakdown is only 302. This disproportion should be compensated. It is possible to use two ways: change threshold (bias) or change weight of the classes. The methods were described in the section 3.2.1. Moreover, a weighted probability loss function was used (see eq. 4.1.1).

Results for the same dataset using the linear kernel and SVM method are much worse. The total precision decreased to mere 80% but it is important to note that value 50% is only a random guess and value 77% if all thee points would be predicted as breakdown. This problem is caused by linear non-separability of the data, as is shown in Fig. 5.1.

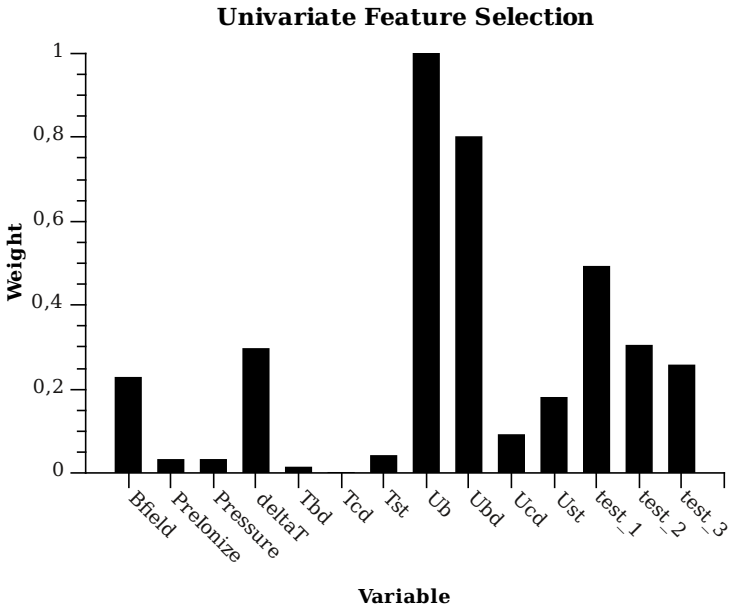


Figure 5.1: Results of univariate feature elimination with ANOVA method applied on data from GOLEM tokamak.

5.2 Random probes

The method of random probes introduced in Section 4.3.3 was used. Three probes were created: the first probe *test_1* was random permutation of numbers from 1 to number of samples, the second *test_2* was random permutation of U_b and the last *test_3* was random permutation of $U_c d$. These probes should have the weight much lower than the rest of the valid variables.

5.3 Univariate feature elimination

The first tested method was univariate feature elimination based on statistical ANOVA test (see Section 4.3). The ANOVA filter returns weight corresponding to statistical difference between groups for given variable. The great advantage is almost immediate speed of the filtering. Disadvantage is quite poor performance compared to other methods. The resulting weights are in fig. 5.1. According to the ANOVA, the random probes are one of the most important variables and, on the other hand, the pressure – theoretically the crucial variable – is negligible. This is significantly different from the expected results. It is caused because the classes in database are not linearly separable variable in each dimension. For example, the breakdown fails for too high pressure and also for too low pressure.

The unexpected weights of the random probes could be caused by violation of of the ANOVA model assumptions, i.e. independence of cases, normality of the residuals, homogeneity – variance of the groups should be the same.

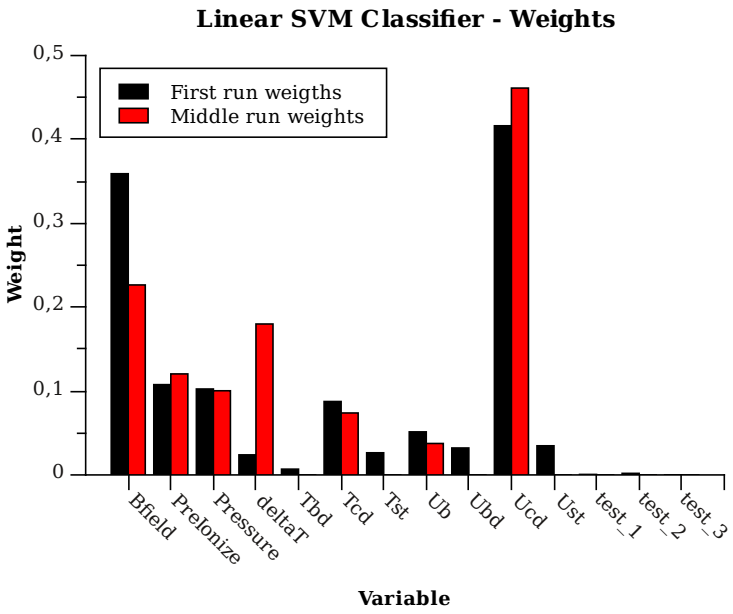


Figure 5.2: Weights for input dimension used in the GOLEM tokamak when the recursive feature elimination with linear SVM was applied. The black bars are weights for all variables. The red bars are weights for seven the most important variables.

5.4 Recursive feature elimination with linear SVM

The next tested method is based on the Recursive Feature Elimination (RFE) with linear SVM, but the results are similar for all the described methods with the linear kernel. The solving speed was slower than the ANOVA filter but still it took only a few minutes. The weights are normalized absolute values of the normal vector of the SVM linear hyperplane. It is possible to make prediction for all variables and use only n of the most important or iteratively remove only the least important dimension and perform the training and prediction/pruning again. The first way is faster but the second way gives result that should be less affected by random variable that were removed in previous steps.

The results are shown in Fig. 5.2. The results are significantly different from the ANOVA weights, the linear weighting successfully ignored random probes. Also it is important to note that the weight stayed very similar during elimination, the only exception is ΔT variable.

Basic statistical characteristics of the fitted model are shown in the following table. Recall for the class 0 is very low; it means that linear model almost ignored the smaller class. However, it was expected since the model is not linearly separable.

Group	Precision	Recall	F1-score	CV-score
0	0.77	0.29	0.21	0.39
1	0.80	0.97	0.44	0.78
Total:	0.80			

The next step is to use the weights from linear hyperplane estimate the influence of each dimension.

keys:	Bfield	PreIon	P	ΔT	T_{bd}	T_{cd}	U_{bd}	U_{cd}
weights:	0.34	0.34	0.33	0.12	0.08	-0.13	-0.20	0.53

Almost all variables increased breakdown probability with increasing variable size, the only interesting exception is voltage in breakdown coils – U_{bd}. However, the breakdown field should always improve the chance to reach breakdown. This was probably a non-physical effect caused by the fact that the highest fields were tested in the cas of machine failure.

5.5 Recursive feature elimination with RBF kernel and SVM

This method was described in the Section 4.3.1. The weights correspond to increase of loss function when the belonging variable was removed. Full cross-validation was performed in order to find the best parameters for each variable combination – each parameters combination was solved 5 times. The results from the cross-validation were used to estimate errorbars of the weights.

The advantage of the method is that it is fully nonlinear method and thus the weights are not biased by assumption of linear separability, the disadvantage are high computational demands.

The results are shown in Fig. 5.3. The results fit very well to the expected behavior. Weights of the random probes are not zero, but within the errorbars they are correct. Also, on the contrary to the other method, pressure was determined as a very important variable together with the magnetic field and current drive. It is also quite interesting that ΔT, delay between breakdown and current drive, have no importance although linear SVM and ANOVA filter predicted quite high importance to this variable.

In Fig. 5.4 is plotted the order of the eliminated variables. The variables eliminated at the beginning have very similar weights, so the order is not reliable. Interesting is that probe variable *test_3* stayed quite long.

Finally, the evolution of the predictive probability for each step of the RFE algorithm is in Fig. 5.5. During the first four steps, the prediction is almost constant then the classification rate is decreased although the *test_3* variable was still in training set, so it was still overfitting.

The optimal number of variables is 5-6.

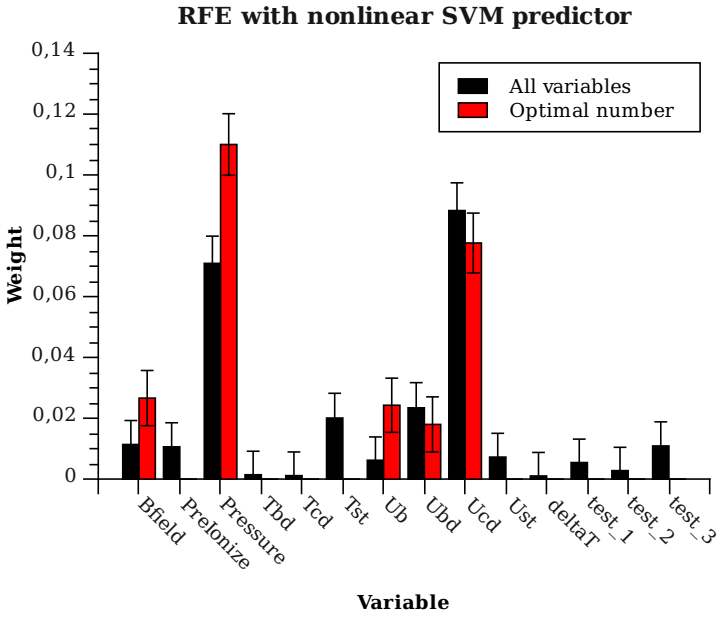


Figure 5.3: Recursive Feature Elimination (RFE) using the nonlinear SVM predictor. Errorbars are estimated from the cross-validation

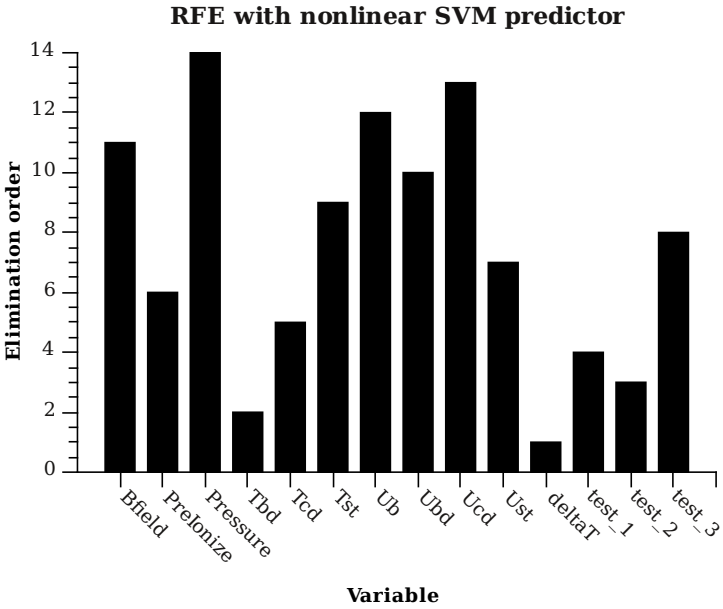


Figure 5.4: Elimination order of variables using the Recursive Feature Elimination (RFE) and nonlinear SVM. The lower number, the less important variable.

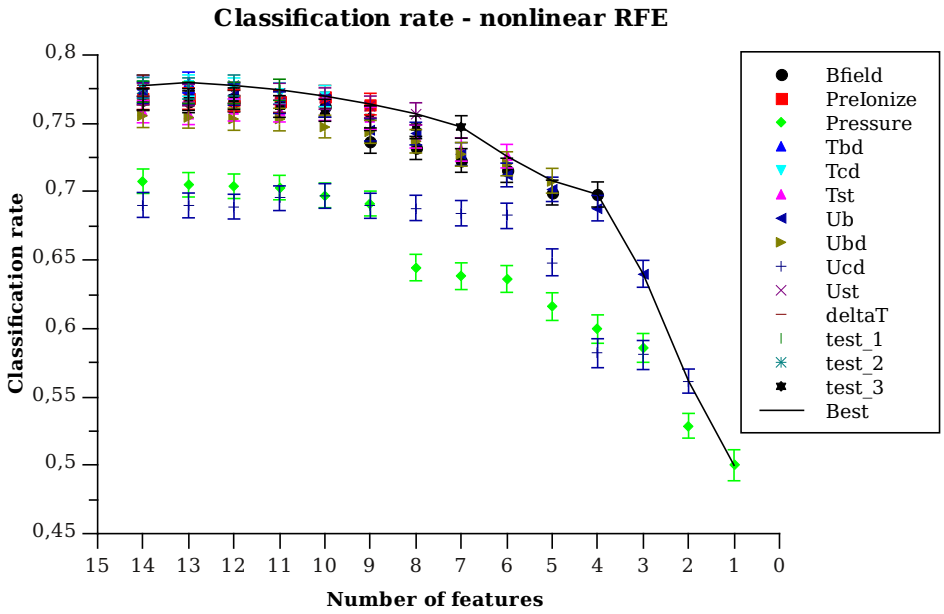


Figure 5.5: The classification score for a different number of inputs selected by the non-linear RFE

Results for different learning machines when only four dimensions were used are in following table. The differences between the misclassification are not significant for the different algorithms because random changes between each run can produce different results. One important property is sparsity of the model. The sparsest and thus usually the less complex model is the RVM. Interesting is that the linear SVM have less sparse model but it is caused by linear non-separability of the model and thus by high misclassification rate and the plane can be still described by 5 parameters.

Machine	Total Missfit	Main vectors
RBF kernel - 4 dimensions		
SVM	0.941	34%
LogReg L1	0.926	100%
LogReg L2	0.959	100%
RVM	0.912	5%
linear kernel - all dimensions		
SVM	0.67	46%

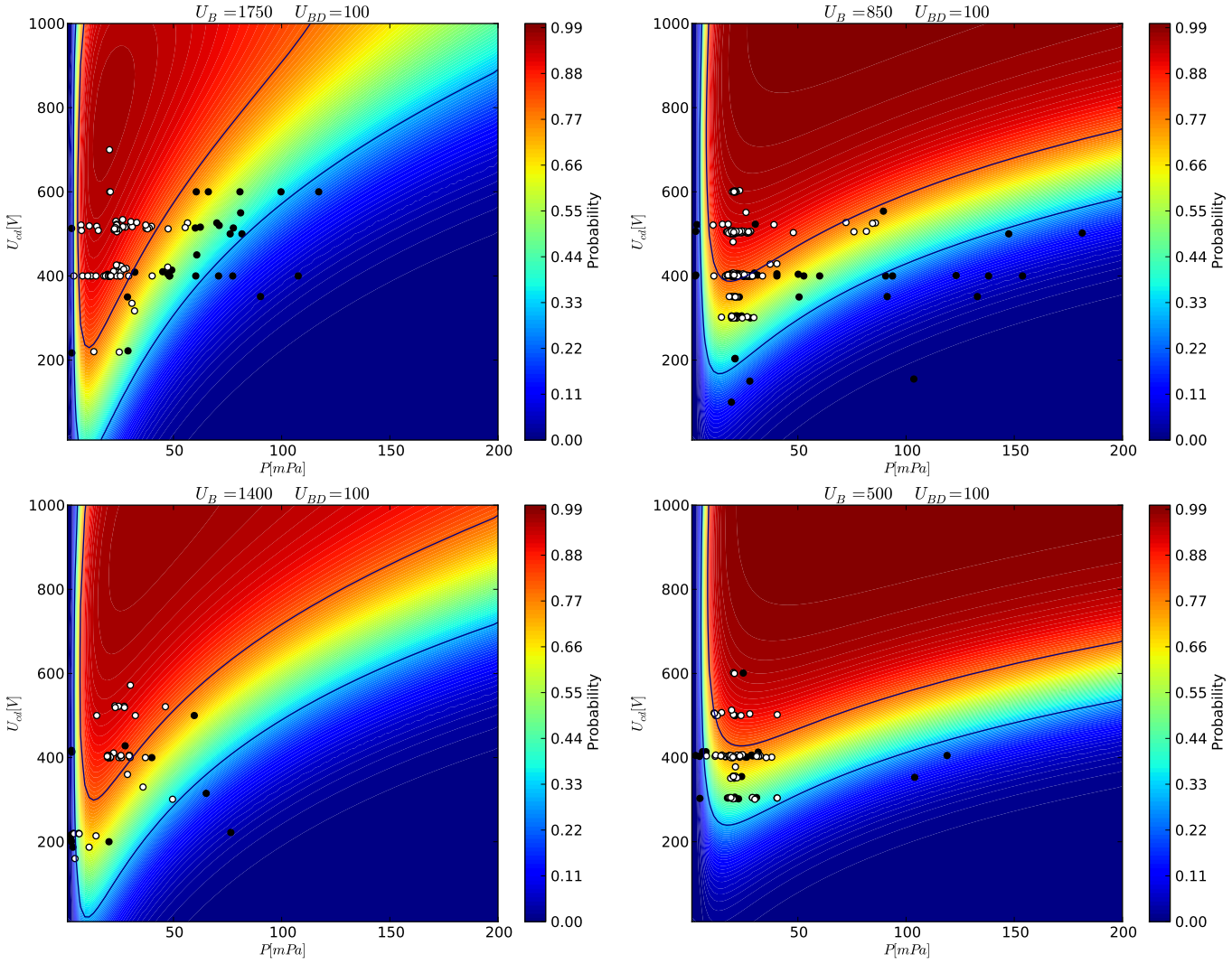


Table 5.1: Cuts through the predicted probability of breakdown. The probability prediction is the output of the SVM algorithm LibSVM. Black points are shots without breakdown and white points are with breakdown. Contour lines denote 30% and 80% decision boundary. It should be noted that the boundary shape is very similar to the Paschen's curve [33].

Chapter 6

Disruption Prediction on Tokamak JET

The aim of this work is to use machine learning tools to understand the physics of disruptions in the perspective of predicting them.

The work is particularly focused on three problems: the possibility of extrapolation of predictors from low current disruptions to high current disruptions, the performance of predictors trained using intentional[1] disruptions for unintentional disruptions (and vice versa) and the prospects of training predictors using non dimensional parameters. These issues must be studied if learning machines are to be used for the future thermonuclear machines. Moreover, many minor issues were studied.

6.1 Introduction to Disruption Prediction

The disruption is defined as sudden loss of the plasma confinement that results in the abrupt termination of the plasma discharge. Disruptions are potentially serious threats to the integrity of the ITER[2] tokamak [34] chamber integrity. Disruptions usually consist of several phases [1]: the first phase of the disruption is called thermal quench. During this phase huge amounts of heat are released to the plasma facing components. This phase is followed by a current quench. A fast current quench with high internal inductance can cause high negative loop voltages and it can lead to production of the relativistic "runaway electrons" with energy up to 100 MeV. These relativistic electrons can cause severe damage of the plasma facing components and also damage the tokamak first wall. Almost 70% of the initial plasma current (10 MA) can be converted to the runaways during a disruption or a fast current quench in the ITER [35]. It is also possible that the vertical control system loses control of the plasma position and this can result into a VDE (vertical displacement event) and, at high currents, VDEs can cause high forces and damage the vessel. They can also induce large halo currents and again cause damage to the vessel or other components.

[1] disruptions cased by a human intention under controlled conditions

[2] ITER is the largest tokamak that is being currently build in Cadarache, France

The main problem with the disruptions is that a minor damage in the present tokamaks can cause major integrity issues in the future thermonuclear devices. The magnetic energy will be increased from 11 MJ in JET to 400 MJ in ITER, plasma current will be increased from 5 MA in JET to 15 MA in ITER [35]. Because disruptions seems to be unavoidable in tokamaks, although the disruption rate on JET in recent campaigns was limited to 3.4% [36] , in ITER the expected disruptivity is 5-10% [34]. It follows that it is necessary to predict and then mitigate or suppress the disruptions in ITER. The success rate of the disruption prediction should be >95% [37] to prevent too fast erosion of the divertor.

The idea of using learning machines to predict the disruptions is quite old and examples of the real-time prediction applications based on Neural Networks already exists [38]. With newer computers and better learning machines algorithms, it should be possible to reach better results. However, most of the current articles in the literature, and also this work, are focused only on the proof of concept and not on the real-time deployment of the predictors in actual devices.

In many experiments, the prediction was based on monitoring one or two diagnostics i.e [39] locked modes or global plasma energy equlibrium (see the next Section) and the trigger was started with some appropriate threshold. The problem of these methods is limited ability to detect disruptions in different plasma configurations and at the same time achieve low false alarm rate. Another possibility is to use Neural Networks (NN) with multiple diagnostics as inputs. The neural network output can be trained to predict the time to the disruption [6] or a number proportional to the probability of the disruption. Usually, an appropriate threshold is also selected to launch an alarm. The disadvantage is the necessity to choose the network topology artificially, the main advantage is the high speed of the predictions, although it is not a critical issue for disruption prediction, nowadays. An example is the real-time application of the NN that is able to predict the density limit disruptions for the ASDEX Upgrade [38]. 13 diagnostics were used as input and the alarm must be triggered 50 ms before disruption. The used training database contained 99 disruptive and 386 nondisruptive shots. They reached 15% missed rate on 65 disruptions and 1% false alarm rate for 500 non-disruptive shots.

Another example is NN with 9 inputs that was used for the JT-60 [40]. The NN was trained with merely 12 disruptive and 6 nondisruptive shots that were manually selected and manually classified to determine different stability levels ("precursor time"). Two iterations were used in the training. The second iteration used the predictions of the first step to identify the "precursor time" and also some additional synthetic artificial points were added to improve the prediction. The synthetic data were placed on positions that were not covered by measurements, however the disruptive character of that part of the operational space was known from a priory knowledge. The results were 97% success rate 10 ms in advance for 300 disruptive shots with 2.1% false alarm rate for 1008 nondisruptive shots.

Also for the JET tokamak, several NN based algorithms were developed. An example is published in in [6], authors reached a success rate of 80-90% with zero false alarms because the effort was to minimize the false alarms.

Next interesting topic is the cross-machine prediction. The NN predictor trained for ASDEX reached 69% success on JET and a predictor trained for JET reached 67% on

ASDEX data [9]. The article reports that the portability of the learning machines is an challenging issue and other approaches need to be developed.

Furthermore, other machine learning tools have been already used. In [41], three SVM machines solved in parallel were used with a second linear SVM layer that was used instead of the threshold. With 13 classical disruptive input variables resampled at 1 kHz, the success rate >90% was obtained.

A direct compare of the results from different articles is difficult since each author measures success rate in a different way, moreover as it is shown in this chapter, success rate depends more on the data – disruption types – than on the used learning machine.

The following notation for the score is used in this work: % FA (false alarms) is the percentage of triggered alarms on the nondisruptive shots, %MA is percentage of non triggered disruptive shots at least 30 ms, %EA (early alarm) is percentage of alarms triggered more than 1 s before the disruption. [3] It should be noted that it is always possible to minimize FA or MA separately by an appropriate threshold but the sum stays usually almost constant near to the optimum.

6.2 Operational Limits

The operational space of the tokamak is restricted by several limits. The most known are current limit, density limit and pressure limit but more empirical limits exist [1]

Current limit – limit on the safety factor $Q95$ [4]. The value is limited to $Q95 \geq 2$ otherwise it results in a low-q disruptions. When the $Q95$ is near to 2, the tearing/kink instability starts to grow, get locked with the wall and results into a disruption.

Density limit – the plasma density should not significantly exceed the empirical Greenwald density limit $\mathrm{n}_{GW}\ [10 \cdot 20\,\mathrm{m}^{-3}\ ] = \mathrm{I[MA]}/(\pi \mathrm{a}^2\ [\mathrm{m}^2])$ for circular plasma of diameter a. If this limit is exceeded the impurity radiation at the plasma edge is increased and this can lead to the radiative disruptions. The radiation is mainly caused by not fully stripped impurities or by intense hydrogen gas fueling. The edge plasma cooling contracts the plasma current profile and if the $Q95$ is not high enough, the current contraction leads to the low-q disruption. This limit is only a soft limit and it can be improved by decreasing impurity content and increasing additional heating [42]. It follows that the edge parameters of the plasma are important: electron and impurity densities, the edge plasma temperature. The limits for the disruptions can be estimated by Murakami parameter [43] $nR/B_{fi} \cdot 10^{19}$ and the inverse safety factor $1/Q95$. The final diagram is called the Hugill diagram [44] and it was verified on many tokamaks i.e for JET [45].

[3] The expected minimal mitigation time for JET is 30 ms because the later mitigations are not enough effective. Therefore, the learning machine algorithms should predict the disruption at least 30 ms before a major disruption otherwise the shot is treated as a missed alarm. The thermal and current quench or vertical instability are detected usually too late and they cannot be used for the prediction and therefore other quantities must be used. The minimal mitigation time will be longer in the case of the ITER because of its size.

[4] measurement of magnetic field helicity

$Q95$	Safety factor
I_p	Plasma current
n_{GW}	Greenwald density
n_e	Electron density
B	Toroidal magnetic field
B_p	Beta poloidal
LOCK	Locked modes amplitude
l_i	Plasma internal induction

Table 6.1: The list of the important disruptive parameters regards to the operational limits

Pressure limit – the normalized toroidal beta $B_N = \langle B_p \rangle / I[\mathrm{MA}]/(a[\mathrm{m}]B[\mathrm{T}])$ should be lower than the Troyon limit ≈3.5. For high $Q95 > 3$ the beta limit is only a soft limit but for $Q95 < 3$ is problematic to cross the boundary (rollover) [34]. The B_N limit is often observed for peaked pressure profiles [40]. The evolution of the beta-limit involves the development of NTMs (neoclassical tearing modes) and ballooning modes. According to the numerical models [35], convections cells develop between the hot plasma core and the edge plasma, therefore the heat conduction across the magnetic lines is increased and the plasma confinement is decreased. The simulations indicate that the energy loss is caused by ballooning modes. In the next step an internal minor disruption appears and it is usually followed by the major disruption. The Beta limit disruptions are not simple to detect because the first precursors appear a few ms before the disruption [40].

Locked modes – the tearing modes or islands do not rotate with the plasma fluid. The main causes are error fields or eddy currents caused by big islands near to the plasma surface. The rotating modes are slowed down and locked; therefore the modes start to grow rapidly and cause a disruption. The amplitude of the 2/1 mode is often used as a disruption precursor.

Too fast plasma current ramp up – destabilizing MHD (magneto-hydrodynamic) instabilities can be produced in plasmas quite often if a rational value of safety factor is near to the plasma surface [1]. It can be aided by a hollow density profile. It results in the locked modes and disruptions.

Plasma energy equlibrium – plasma output power should not significantly increase the inout power for a longer period.

Plasma current ramp down with high l_i – a disruption can often appears if the plasma current profile is strongly peaked during the ramp down phase. The peaking of the current profile can be estimated from increase in the plasma internal inductance. Precursors for this type of disruption are usually not clearly observable [40] but the disruption depends on crossing a boundary in q_{eff}(effective safety factor) vs l_i plot. It is called Cheng's $l_i - q$ diagram [46].

In the table 6.1 the important parameters for the prediction of disruptions induced by operating too close to the operations limits are summarized:

These diagnostics are quite similar to the inputs that we have finally selected (see Tab

JET Disruption Classes			%
Impurity (control problems)	NC		18.7
Density control problems	IMC		15.6
Auxiliary power shut-down (H-L)	ASD	0.04-0.8s	10.0
Fast emergency shut-down	FSD	0.1->2s	9.6
Neo-classical tearing mode	NTM	0.1->2s	8.2
Shape control problems	SC		6.0
Current ramp-up	IPR		5.9
(Low density) Error field mode	EFM	0.1-1s	5.6
Strong internal transport barrier	ITB	0.01-0.05s	5.1
Vertical stability control problem	VSC	0.02-0.1s	4.6
Greenwald limit	GWL	0.05-0.8s	2.4
No clear classification			8.2

Table 6.2: Ratio of different type of disruptions in the JET tokamak [36]

6.6). However, it is important to minimize the number of the input parameters to the learning machines, therefore it can be expected that the magnetic field is usually similar for all the shots and can be neglected. Moreover, if the plasma center position is added to the parameters set, the $Q95$ (or plasma current) can be removed because it depends approximately only on plasma current ($Q95$) and plasma surface.

The input set in the Tab 6.1 contains some important disruptive parameters. In addition to the table 6.1, it contains the radiated power, the input power and the derivative of the stored diamagnetic energy. The derivative of the stored diamagnetic energy is claimed to be an important precursor [34].

The main disadvantage of our choice of inputs is that none of the parameters is non dimensional and therefore the results are not applicable to other tokamaks. On the other hand, a study about cross machine training [9] proved that the cross machines results are very weak even the with non dimensional parameters.

The problem is even more complicated because the parameter importance depends on the number of disruptions in each type (Tab 6.2). Therefore, the best set of the signals is selected to maximize the score on a general training dataset (see Section 6.10)

6.3 Shot Database

The total processed database contained 2309 disruptive shots and 1654 of the disruptions were unintentional. We have also used a database that contains 1187 shots with disruptions of known physical cause. The list of disruptions types is in Tab 6.3.

The database contains shots from campaigns C15-C27 but we have not used all campaigns because major changes of some tokamak diagnostics have been implemented before the campaign C19. Therefore, the training dataset contains campaigns: C19, C20, C21, C22 and the testing database contains shots from the campaigns C23, C24, C27b.

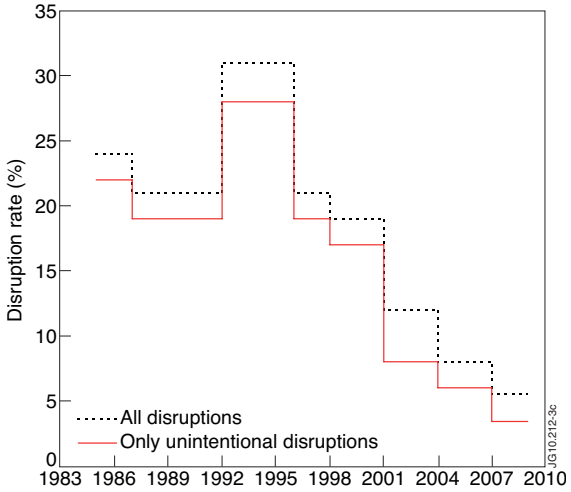


Figure 6.1: Disruption rate during JET lifetime. source [45]

Type	
Too little auxiliary power	ASD
Greenwald limit	GWL
Impurity control problem	IMC
Too fast a current ramp-up	IP
Too strong internal transport barrier	ITB
Too low density (and low q)	LON
Density control problem	NC
Neo-classical tearing mode	NTM

Table 6.3: Types of disruptions in the used database, campaigns C15-C27

Name	Date	Pulse Range
C15a	18 April - 10 May 2006	65985 - 66355
C15b	12 May - 25 May 2006	66361 - 66572
C16	5 July - 7 July 2006	66954 - 66976
C16-C17	25 September - 15 December 2006	67641 - 69148
C18	8 January - 12 February 2007	69227 - 69795
C19	12 February - 4 April 2007	69806 - 70750
C20	11 April 2008 - 6 June 2008	72150 - 73854
C21	9 June 2008 - 18 July 2008	73130 - 73854
C22	21 July 2008 - 29 Aug 2008	73855 - 74463
C23	1 Sept 2008 - 3 Oct 2008	74464 - 75115
C24	6 Oct 2008 - 6 Nov 2008	75116 - 75599
C25	10 Nov 2008 - 16 Dec 2008	75600 - 76329
C26	12 Jan 2009 - 7 Apr 2009	76395 - 78157
C27a	15 Jul 2009 - 14 Aug 2009	78296 - 78884
C27b	7 Sep 2009 - 23 Oct 2009	78995 - 79853

Table 6.4: Date and pulse range for each used campaign

	Training data			Testing data		
Type	All	High Current	Intentional	All	High Current	Intentional
Unknown	95	20	58	71	18	36
ASD	12	3	0	5	4	0
GWL	4	0	0	3	3	0
IMC	22	1	0	9	3	0
ITB	4	0	0	2	2	0
LON	0	0	0	10	1	8
NC	26	4	0	15	6	0
NTM	14	7	0	4	0	0

Table 6.5: List of disruption types used for training and testing. Names of the disruption types are listed in Tab. 6.3

178 disruptive shots were used for training and internal validation. However, 42% of these shots were intentional disruptions and therefore they were removed from the database except the Section 6.12.2. The reason is explained in the next Section. From the rest of the 120 disruptive shots only 35 shots were at high current >2MA at the time of the disruption.

The disruption types used for the training and testing are reported in the following table:

The available diagnostics were another limiting factor for the number of used shots because all the shots with some missing input diagnostics were removed. The used diagnostics were selected to minimize the number of missing data and also we preferred to include only the signals that should be available in real-time. Therefore, a compromise between the number of shots and the number of used diagnostics had to be found. The input diagnostics used in this work are in Tab 6.6 and histogram of each input is in Fig 6.2.

6.4 Intentional Disruptions

Intentional disruptions are caused by human intention under controlled conditions. Usually us used a gas injection in combination with a low safety factor. These disruptions are quite often in the used JET disruptions database and represent more than 42% of disruptions in the campaigns C19-C23. However, these disruptions behave very differently from the non intentional. The output of the Grand Tour algorithm [47] – a multidimensional projection from the space of all inputs and their preprocessing (derivations, variations) to a 2D image – is shown in Fig 6.3. The data are in the time range [-60 ms, 0 ms] before the disruption. Each dimension was normalized to the 95% quantile. In the Fig 6.3, it is visible that the intentional and unintentional disruptions are clearly separated.Moreover, the intentional disruptions are usually very similar and the precursor behavior is very short and this causes problems during the learning and the testing using these disruptions.

Abbrev.	Diagnostics	PMDS address	dt [s]	dt_{final} [s]
IPLA	Plasma current	jpf/da/c2-ipla	2e-4 – 1.5e-2	1.5e-2
BTPD	Poloidal beta	jpf/gs/bl-bpdiam<s	1e-2	
LOCK	Mode lock amplitude	jpf/da/c2-loca	2e-4 – 1.5e-2	1.5e-2
PIN	Total input power	jpf/gs/bl-ptot<s	1e-2	
PPOZ	Plasma vertical position	jpf/gs/bl-zcc<s	1e-2	
DENS	Plasma density	jpf/df/g1r-lid:003	1e-3	1e-2
WDIA	Derivative of diamagnetic E	jpf/gs/bl-fdwdt<s	1e-2	
POUT	Total radiated power	jpf/db/b5r-ptot>out	1e-4	1e-2
NGW	Greenwald density	jpf/gs/bl-gwdens<s	2.5e-3	
TAU	Confinement time	SCAL/TAU	1.5e-2	
BETN	Beta normalized	SCAL/BETN	1.5e-2	
Q95	Safety factor	EFIT/$Q95$	1.5e-2	
INDU	Plasma internal inductance	jpf/gs/bl-li<s	1e-2	

Table 6.6: The complete list of all the downloaded and tested JET database outputs. These variables were used to compute the final set of parameters. Time resolution of some dimensions was reduced on order to limit the data size and remove problem with incorrect preprocessing (see Section 6.6)

6.5 Data Preprocessing

The first step of the implementation was the processing of the original data. The aim was to compress the information and minimize the loss of the important information. The data for disruptions are in form of time sequences of several signals (Tab 6.6). The data are already down-sampled in the JET database to limit the database size. The time resolutions of different diagnostics are not the same and also the time resolution can be different when some important action happens i.e. before the disruption. Several ways, how to process the differently sampled data, exist. The linear interpolation of the signals to 1 ms was used in articles [6, 41]. This approach has several disadvantages. The most of the diagnostics are saved in >10 ms resolution; therefore it increases the amount of data that need to be processed. And the second issue appears when the time resolution of the diagnostics is not constant during the discharge. This problem is discussed in the Section 6.6.

In this work, the following preprocessings were proposed:

mean	Mean value in each window
diff	Mean value smoothed derivative (“derivation”)
var	Standard deviation of the derivative (“variation”)

Table 6.7: List of different preprocessing used in this work

These preprocessings detect absolute value, increase and fluctuations of the inputs. It is theoretically possible to use even higher order derivatives but the signals are usually

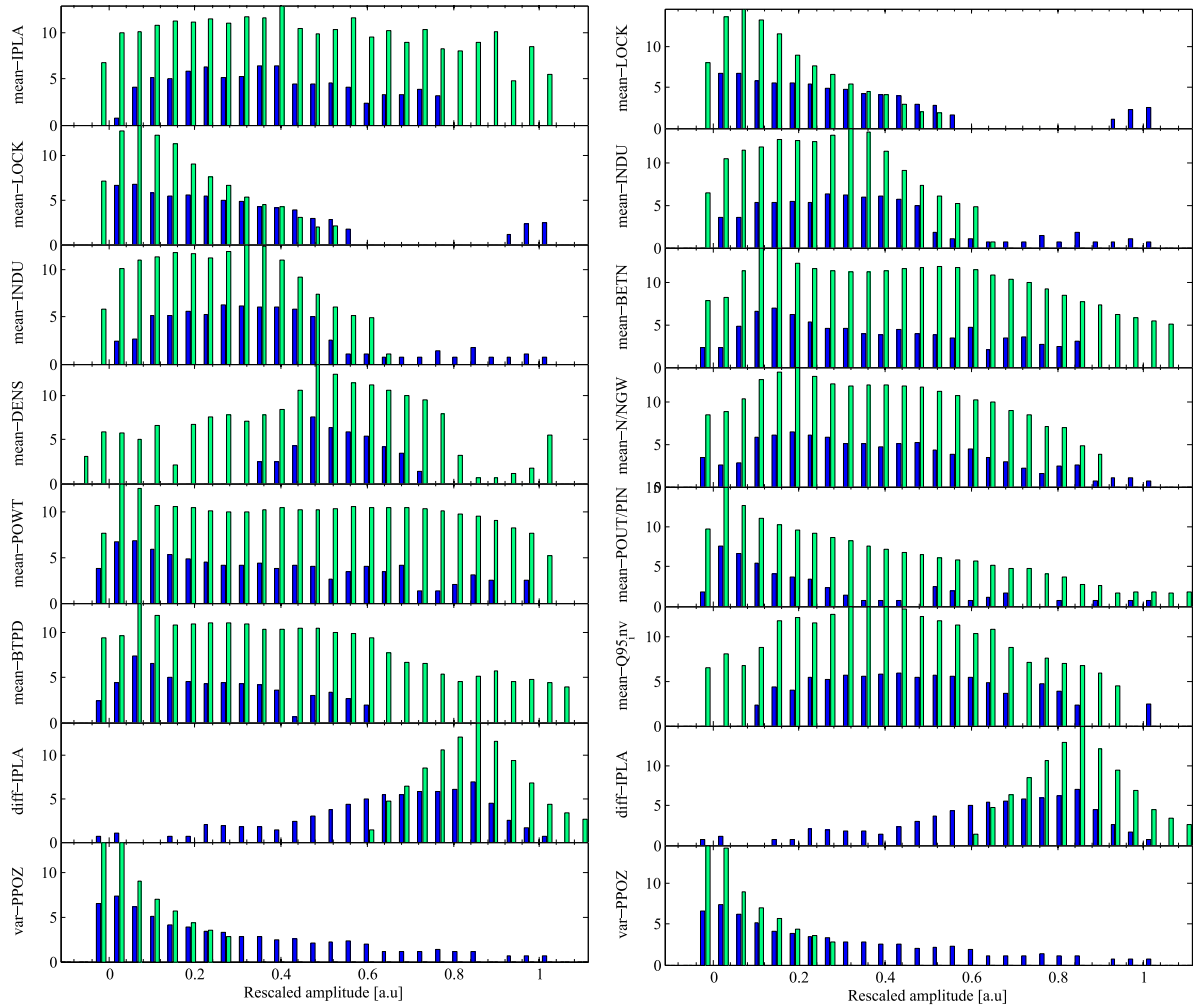


Figure 6.2: The histograms of the input variables. The inputs were rescaled to the same range, histogram values are plotted with the logarithmic scaling, green bars are the nondisruptive and blue are the disruptive values ($t_{disr} + [-150; -30]$ ms). Histograms on the left shows the set of the ordinary parameters, histogram on the right side are predominantly dimensionless parameters with some important ordinary parameters.

very noisy and even the first derivative is very noise with no smoothing. However, in [40] the second derivative was used to detect the start and the end of the plasma current rump-up/down from the reference plasma current.

Also the variance sensitive preprocessings (std(fft), std(diff)) have very similar results as the ordinary absolute value of the mean time derivative (see the next Section).

The length of the windows was chosen 30 ms because the data were usually saved to the database with sampling >10 ms and therefore usually the each window contained only four values (two real and two interpolated on boundaries). We have used overlapping windows shifted by 10 ms to increase the number of the disruptive points.

It should be noted that only the density and the radiated power are saved at high time resolution in all shots and the plasma current and the locked modes are saved at higher time resolution only when some more important events happen (i.e disruption).

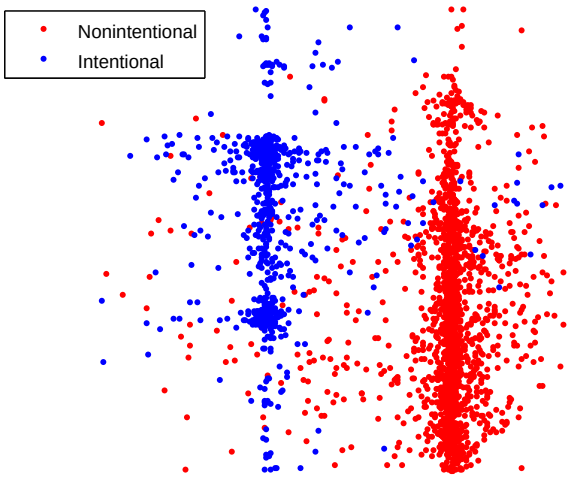


Figure 6.3: The Grand Tour projection of the input data. Only the data from the time interval -100-30 ms are shown. Dimensions are rescaled so that the 5–95% quantile would be equal 1, and each group contains the same number of the points.

The next issue were inputs failures that led to unphysical results such as zero input power after NBI shutdown, negative density, negative radiated power, negative Greenwald density etc. Some of these failures were possible to remove with the median filter, another were removed by a priori knowledge i.e minimal input power $\approx 500\,\mathrm{kW}$.

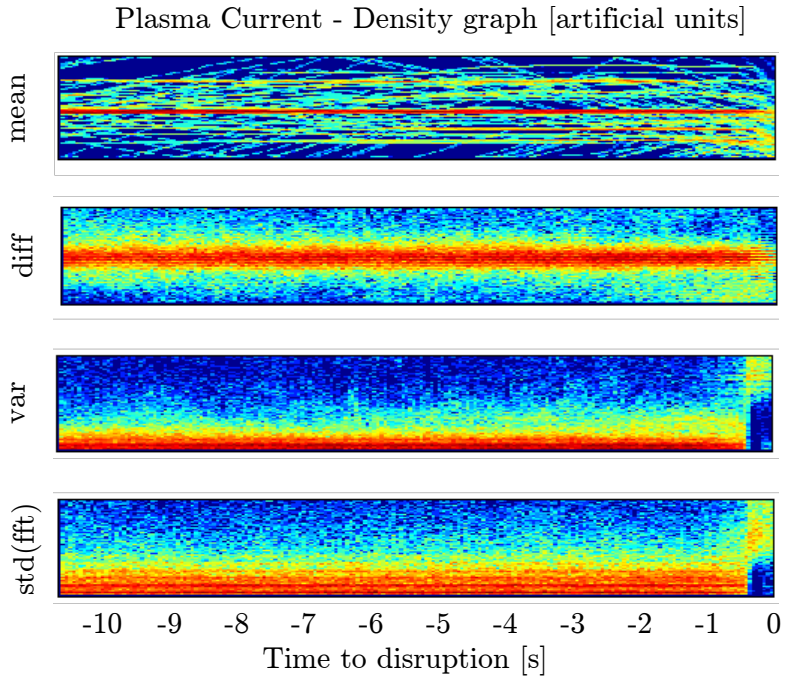


Figure 6.4: Examples of signal density of IPLA when is used the wrong downsampling. The time vector of all signals were shifted such that the disruption is in time 0. Results for LOCK are very similar.

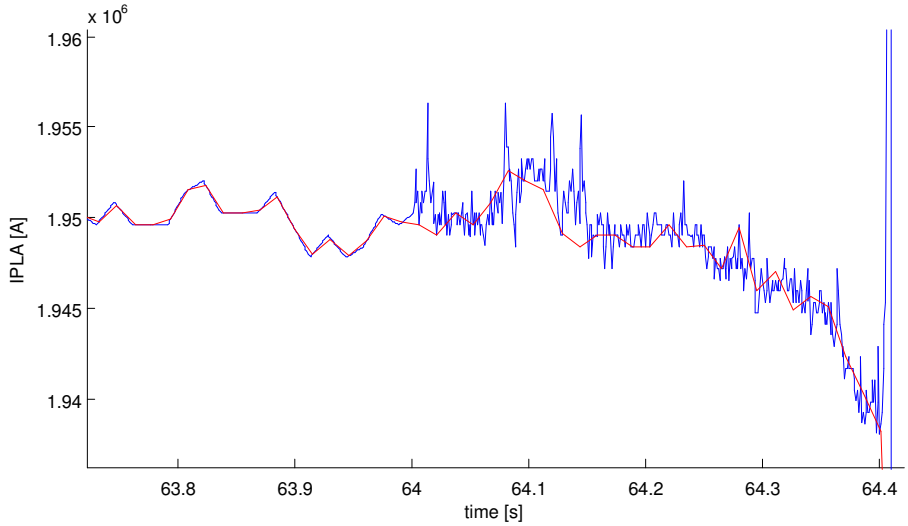


Figure 6.5: An example of plasma current for a disruptive shots.Disruption is in time 64.41 s. Blue curve are data interpolated at 1 ms, red curve are data interpolated at 15 ms. The incorrect downsampling cause artificial change of signal 400 ms before the disruption, because the IPLA is saved in JET database with different time resolution.

6.6 Problems of Preprocessing

One major problem was found during the data processing. The IPLA and LOCK diagnostics are saved in the JET database with a different time resolution in some shots and $\approx$ 300 ms before each disruption the sampling rate is increased from 15 ms to 0.2 ms. If the linear interpolation (`interp1`) is used for the IPLA and LOCK diagnostics to interpolate/downsample at 1 ms the result is shown in Fig 6.5 (blue curve) and if an appropriate preprocessing is used i.e std(fft) the frequency transition is very clear The only correct fix is performing the same downsampling as the JET Data Acquisition System (DAS). The data must be downsampled at the lowest frequency in each diagnostics separately. The original JET DAS undersampling used the nearest neighbor without any smoothing. It is necessary to use this pessimistic way and perform a possible information loss otherwise all preprocessings sensitive to the data variation as std(fft), std(diff) are able detect the sampling frequency change and the used learning machine is able to use the information for the prediction. The sensitivity of the variation preprocessing (6.7) is plotted in the Fig. 6.4 and the corrected version is in the Fig. 6.6.

The smoothing is performed only before the derivative preprocessing. The inputs were smoothed using the Parzen window filter with a semi (unsymmetrical) Gaussian weight (to prevent the information shift) with the kernel size of 80 ms.

6.7 Data Smoothing

Although we have used 30 ms windows for data processing, the noise in the data is still significant when derivatives are used. Therefore, each signal should be smoothed before derivation. The problem is that incorrect smoothing can shift in time the information

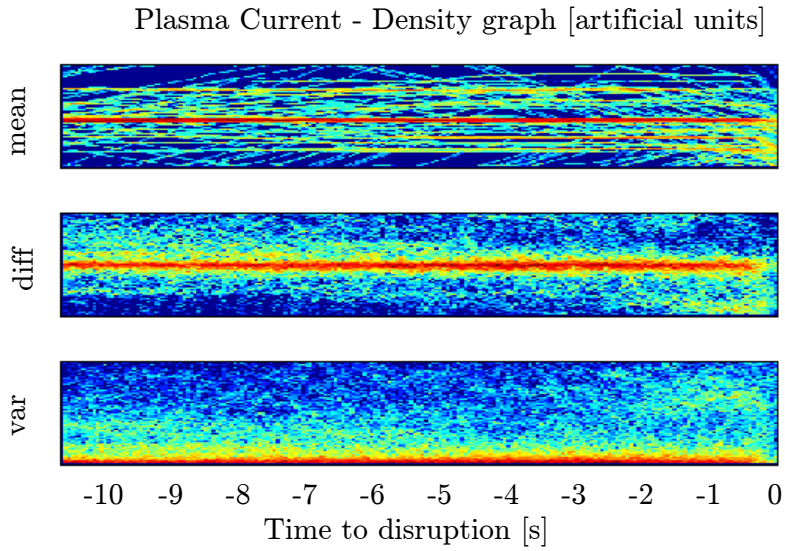


Figure 6.6: Examples of proper downsampling - graph of lines density for time before disruption for IPLA. There is no significant change is the signal 400 ms before disruption.

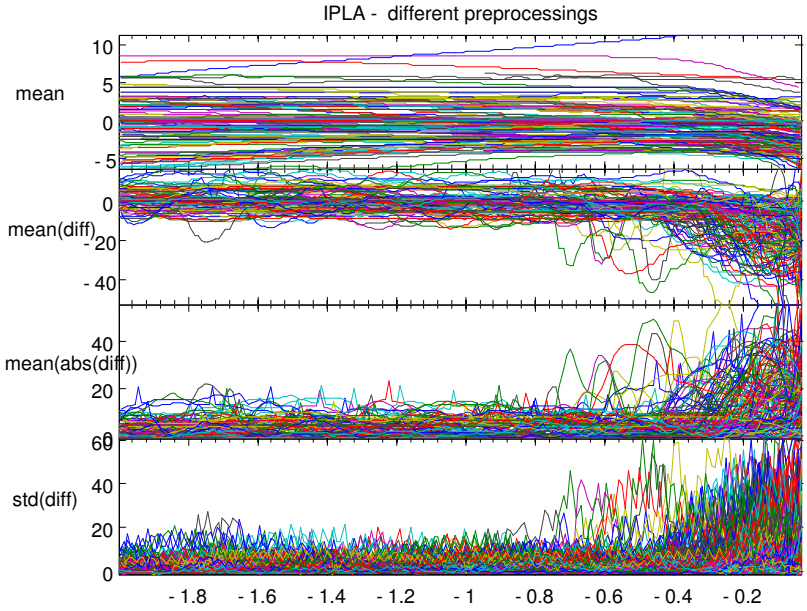


Figure 6.7: Examples of the different processing of the plasma current input. The mean and derivative is smoothed over 30 ms windows and the last two are examples of variations of the signal. The shown signals are from disruptive shots from training campaigns and the time vector was shifted so that time 0 is time of disruption.

about disruptions information and unrealistically improve the score of the prediction. In [40] a low pass filter were used without any specification. No filter/smoothing was used in the articles [6, 41]. We have used asymmetric Gaussian filter to prevent information shift with smoothing σ equal 80 ms.

The only exception is the total radiated power. This diagnostic suffers from unphysical peaks caused probably by wrong deconvolution of the signal from the bolometers (i.e. shot #66053). The median filter over 30 ms was used for the correction.

6.8 Data Postprocessing

In the next step, the data must be preprocessed for the learning machine algorithms, a correct rescaling of the inputs is especially important. This step is essential for the kernel based learning machine algorithms because the normalization gives information about the size of the patterns in each dimension. In the literature it is usually recommended to rescale minimum/maximum of the data to the [0,1] range. However, many outliers are present in the processed data and the outliers can be several orders of magnitude higher than normal data. The same problem remains if the data are rescaled to unit variance or unit norm. Therefore, some robust method must be used. One possibility is to use α-quantile, i.e rescale data from $q(0.01)$ to $q(0.99)$. The best approach would be to set the scaling of each dimension as a free parameter and find the best ratios using cross-validation as it was tested [27] but it would increase the number of free parameters in the system and significantly increase the learning time, because the time of the cross-validation increases exponentially with the number of parameters.

It is also possible to manually introduce a priori knowledge by scaling i.e. by increasing the size (magnitudes) of the more significant dimensions and therefore allow sharper transitions between groups. The outliers could be rescaled by an appropriate transformation i.e. $\mathrm{atan}(x)$ or removed as in [40] but this is not recommended for disruptions prediction because the outliers are usually the most important disruptive signals.

An example of the processed diagnostics (plasma current) is in Fig 6.7.

6.9 Learning Machines

Three machine learning tools were used in this work: Support Vector Machine (SVM) – one and two class – and Relevance Vector Machine (RVM). These algorithms are presented in Chapters 3. All the machine learning tools are sparse and the two class SVM and RVM are able return estimates of the probability that the the data points is disruptive. However, this probability is a very poor estimate because the algorithm is not able to directly use the knowledge from the data time-sequence and because of the outliers.

6.9.1 Total error

In this work, the total error is defined as (%FA+%MA+%PA*disruptivity), however it is only our choice, generally any error in form $A{\cdot}$%FA+$B{\cdot}$%MA+$C{\cdot}$%EA[5], where A, B, C are appropriate positive constants can be selected in order to penalize each type with different weight. The score is then defined as 1−total error. The constants A, B, C were selected in many different ways in other articles, i.e the constants were selected $A = 1, B = 1, C = 0$ in [38, 40] , $A = 1, B = 1, C = 1$ in [41, 48], $A \gg 1, B = 1, C = 0$ in [6] and $A = 1, B = 1, C =$ disruptivity ≈ 0.1 was selected in this work.

The early alarms are less important than the false/missed alarms and often some $\%EA$ can be unavoidable if strong precursors appear too early before the disruptions. On the other hand, the early alarms can be triggered in rare but important shots i.e high current discharges and avoid operation continuity. Our selected constant $C =$ disruptivity ≈ 0.1 gives to the %EA the same importance as to the %FA.

6.9.2 Training

Several problems need be solved before the training phase. The first is the number of the data points. Four campaigns (C19-C23) were chosen for training. It means 2300 non-disruptive shots and 125 disruptive shots that contain more than 10^7 learning points. However, with the available PC, the support vector machines are unusable for more than 10^5 points and the relevance vector machine is limited by 10^4 points.

The SVM algorithm is able to use heuristics algorithms and caching [21] to speed-up the learning. These options significantly increase the learning speed but not enough.

This issue can connected to the problem of the sparse information in the data. It can be solved if the unimportant points are removed from the training data set. In the first step, the data must be removed artificially. We have removed majority of data points that were in 0.1-99.9% quantile range in all dimensions. This step reduced the dataset to <0.5% of the original size. When the learning machine was trained on this reduced set, the estimated probability can be used to remove the unimportant data even more precisely in the next iteration. The training time improvement is shown in Fig 6.8. One important question is the effect of this pruning on the final score, Fig 6.9. shows that there is no significant influence. The training is done on a limited number of the nondisruptive shots.

This allowed us to use all the non-disruptive shots for the learning and with all shots (1246 shots, originally 6108417 learning points, after removing 30619) is the training time on a modern computer < 10 minutes for the two class SVM. The speed could be further improved if more data are removed but the prediction time on the validation/testing data set will not be further neglectable and the overall time improvement is not so significant.

[5] %MA – percents missed alarms, %FA – percents false alarms, %EA – percents early alarms, see 6.1 for more details

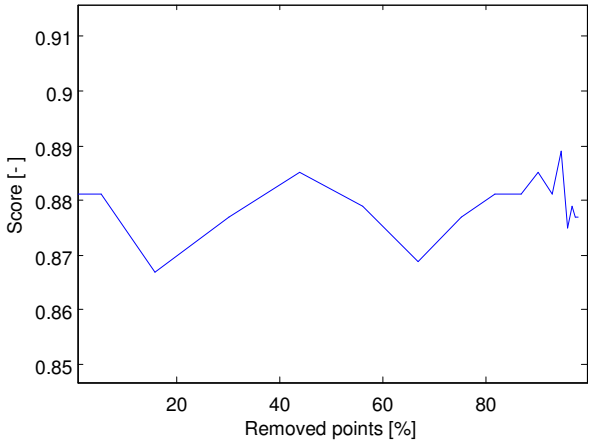


Figure 6.8: The total score as a dependence on the removed data percentage. The score has no significant decline although the number of the training points was significantly decreased. The method of removing unimportant points is described in the previous text.

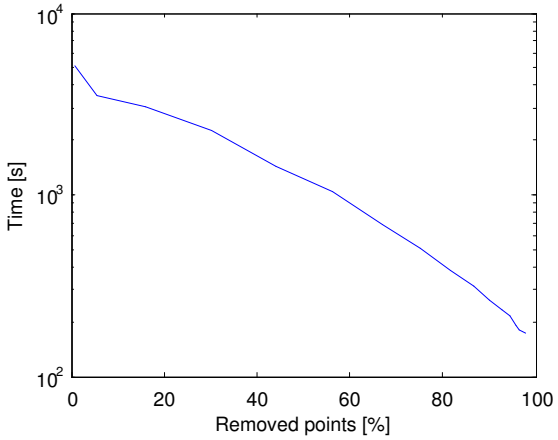


Figure 6.9: Dependence of the training time on the removed data. The training time is significantly decreased when less points were used for the training.

The learning machines have to deal with two issues: the "zero-one" loss function do not corresponds to the score and the data classes are significantly unbalanced – the disruptive points are less than 0.01% of the nondisruptive points. Although the previous step significantly decreases the imbalance, still the nondisruptive outliers can cause that many disruptive data is missed. Both issues were solved by a selection of an appropriate threshold (bias) on the learning machine output (the distance from boundary or the "probability") and we have selected the best threshold that correctly minimize the total error rate. An example of the total error dependence on the threshold is in Fig 6.10. The threshold was searched in a outer cycle and the selection was very fast, it was not used as an additional cross-validation parameter.

Furthermore, the SVM, RVM and LogReg algorithms allow setting of a different weight for each point/class and therefore it should be logical to increase the weight of the disruptive group. The weighting for SVM was performed by choice of different C_i for each group so that $C_i = C \cdot w_i$, where w_i is the weight. However, our tests show, that there is no significant affect of the weighting on the score when the custom optimal threshold is used.

6.9.3 Loss Function

The most important step is the correct evaluation of the learning machine performance during the learning and the testing phase. The used learning machine tools assess each point separately (independent) but this is not valid in the case of time sequences because one misclassified nondisruptive point means one lost shot but one missed disruptive point can be ignored. Also, if the shot is already a false alarm, more misclassified windows do not change the total error. Therefore, the best boundary determined by a learning machine is

not the same as the best boundary that maximizes the number of correctly predicted shots. The same arguments are the reason why the ordinary zero-one loss function ($y_{pred} \neq y_{test}$) do not minimize the total disruption prediction error.

Furthermore, the training and validation set for internal cross-validation must be selected from different shots because the sequential data are strongly correlated and the test/validation data from the same shot leads to the over-fitting. Note that it is necessary to classify all the nondisruptive data points in the selected shots to get the correct false error.

However, the best total error can be an imprecise indicator if a low number of the disruptive shots is used. Therefore, more constrains and penalization functions should be added. We added the area under Detection error trade-off curve (DET), see Fig 6.11 that shows how well are the training groups separated. The score, however, still depends on the number of the disruptions in the validation set, therefore the "optimal threshold" is not selected as to minimize the score but as a mean threshold in the range of the minimal error within the bounds equal to the inverse number of the disruptions (min(Error)+$1/N_{dist}$. Within this bound, the optimal threshold is selected as the center.

It is also possible to add some condition on maximal FA because as it is shown in Fig 6.10 the optimal value can include wide range of MA/FA. We do not recommend to add condition on the maximal MA due to low correlation as is explained in Section 6.15 in detail.

A similar principle to the threshold selection was used in [41] in the second layer. The standard "threshold searching" was done indirectly via SVM with linear kernel and the optimization of number missed/premature shots was done iteratively. However, this solution has no advantages compared to the direct optimization.

Moreover, the threshold optimization is used even during the cross-validation in this work.

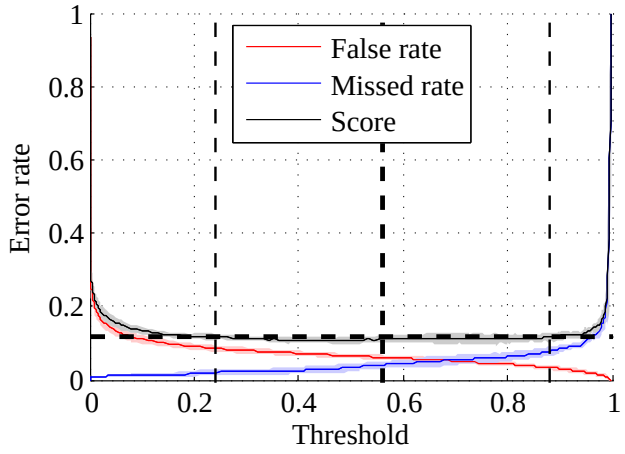


Figure 6.10: An example of the score dependence on the threshold. In this case, the optimal threshold that minimise error is between 0.2 to 0.9. The best threshold was selected in middle equal to 0.55.

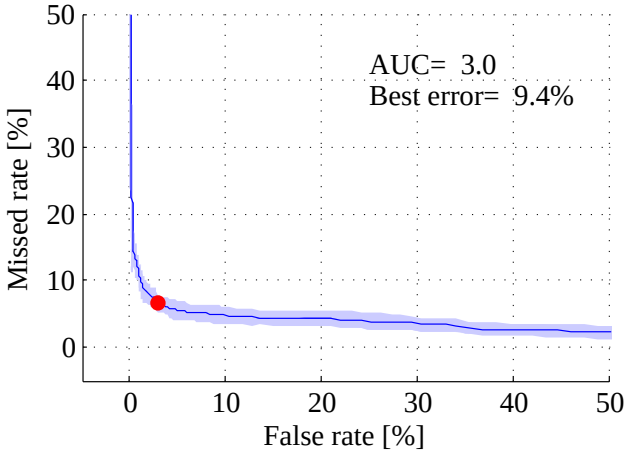


Figure 6.11: An example of the Detection Error Tradeoff (DET) curve. Note that the MA under 5% leads FA >20% for this particular model.

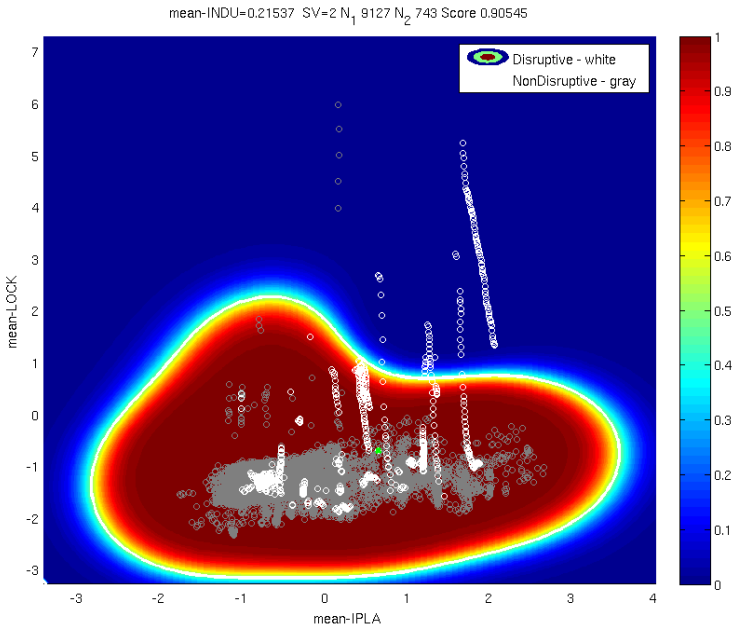


Figure 6.12: An example of the cut through the input space with plotted disruptive (white) and nondisruptive (gray) points and the probability of non-disruptivity. The scaling was selected to show 95% of the disruptive points. Values in all hidden dimension are equal to the median value. The optimal border for disruption recognition is rescaled to be 0.5

6.9.4 Artificial Data

One problem of the kernel based learning machines is that the support vectors are selected from the training data. However, the training data can be very sparse in the interesting area, in our case of the disruptions, and in the multidimensional space big gaps with no data can be present in the disruptive area. It can be solved if artificial points are added to clearly disruptive areas, usually so disruptive that no data were measured in these regions. It can be for example very strong locked modes or very fast decrease of the plasma current. These artificial points can be used by the SVM or RVM algorithms as the support/relevance vectors to get more realistic shape of the probability regions. The best results were obtained when a limited number of these points were used (hundreds) but their weight is increased ($> 10^5$) otherwise the SVM algorithm uses all these points as the SV and number of the SV can be significantly increased.

6.9.5 Identification of Disruptive Data

Another interesting issue in the disruption prediction is the selection of the disruptive data. The problem is that it is simple to identify nondisruptive data i.e. nondisruptive

shots or data more distant from disruption then 1 s. But there is no simple way to select the disruptive/precursor data points. In [6] data were selected from the range [-440 ms; -40 ms] as disruptive. In [41] 3 windows in ranges [-120,-90],[-90, -60],[-60,-30] . In [40] the training shots were classified manually by the operator in order to select the precursor points.

The first possibility is to use the principle described in the previous section and remove some percentile of the "disruptive points". Usually, more than 60% of the "disruptive" points can be removed because there is clearly no precursor. This solution has a big advantage, because all missed vectors are used as support vectors in SVM, therefore the number of support vectors of the disruptive group can be significantly decreased.

The second possibility is an iterative way: use importance weighting with the probability estimated by a learning machine. Therefore, the weight of the "disruptive points" that are predicted to be in the nondisruptive area with high precision is decreased. The weights were used to change SVM penalty $C_i = C \cdot w_i$ separately for each point. The main advantage of this method is the smooth transition between clearly disruptive and clearly nondisruptive. Disadvantage is the higher number of the support vectors for the disruptive class. However, it is also possible to combine it with the previous method.

It is also possible to increase the weights of the important points instead of the penalization as was suggested in Section 6.9.4. One way of the possible implementation is performing the first training without weighting to get an estimation of probability (or distance from the boundary) and in the second step increase the weight to the most disruptive (according to probability) data points before -30 ms in each disruptive shot. It would be also possible to increase the weight of the last window before the disruption or use some weighting depending on the time of the disruption but not always the time window in -30 ms must include the strongest precursor behavior.

6.10 Dimension Selection

The most important issue of the disruption prediction using learning machines is the low a priory knowledge about the importance of dimensions. This problem is even more significant if several preprocessing are to be used (see Section 6.5) and the number of the possible inputs can be hundreds. It should be logical to select the parameters corresponding to the disruptions limits such as $Q95$, n_{GW}, β_N, ... However, these parameters do not have to be optimal because the tokamak operators usually try to avoid the disruptions limits.

Several approaches were used in other articles. In [6] sizes of NN output derivation for each variable were used as significance of importance. Therefore, the authors removed all time derivatives because they were not significant according to this method. Classification And Regression Trees (CART) were used to identify the most significant dimension in [49]. And finally in [48] the authors used genetic algorithms to find the best set of parameters.

We have used a simple Forward Feature Selection principle described in Section 4.3.1. Also other presented method (Section 4.3) were also tested, however issues such as a huge amount of data, nonlinear separability and significant unbalance in the data prevented us

to get reasonable results. The previously discussed diagnostics (Tab 6.6) were used as the original set of parameters with three preprocessings listed in the table 6.7.

The locked modes diagnostics was used as the default input because this variable alone is able to get the success rate more than 60%. Then a signal that the most improves the score was searched. Because no clear rule for the maximal number of the variables exists, we have used the "testing probe" from pseudo random variables introduced in Section 4.3.3. These variables were created from the normal inputs but they have a randomly permuted time vector) and thus the time before the disruption was random. If no dimension is more important than the probes then clearly this is the top limit for the the number of inputs. The top limit estimation is around 10 variables, but it is very imprecise and the limit is rather overestimated.

An example of the FS output is in the Fig 6.13. The white fields are the already selected variables; the color of fields is the score that is reached if the variable is added to the inputs and the left axes is number of the step (number of variables in the model).It should be noted that some variables significantly decrease the results i.e TAU or POUT/PIN, but with more variables in the model, the differences are vanishing. Higher number of inputs usually do not decrease the score. Moreover, if more than 7 variables are chosen, the score remains almost the same for the further additional variables.

Moreover, the dimension selection order shows the physical background. According to the FS the physical limits such as BETN, NGW, Q95 are not relevant for the prediction. The most important are the variables determined from magnetics: LOCK, PPOZ, INDU, IPLA other variables are significantly less important.

6.11 Filtering of Results

The final output from the learning machine algorithm is a time evolution of the disruption probability or other output proportional to the probability. Therefore, it is possible to apply different smoothing on the output to remove outliers. The filters must be applied so that it cannot transfer the information in time and therefore improve the results. The filters are basically similar to the method applied in [41] where the linear SVM behaves as simple weighted moving average over three windows.

Our algorithm was tested with several filters: weighted mean, median. The results are in the following figures 6.15, 6.16. The results implies that the filters decrease number of the false alarms but also increase number of the missed alarms. The explanation is that the filter removes the rapid increase of the disruption probability short time before the disruption itself. Therefore, the disruptions with a short precursor cannot be triggered early enough.

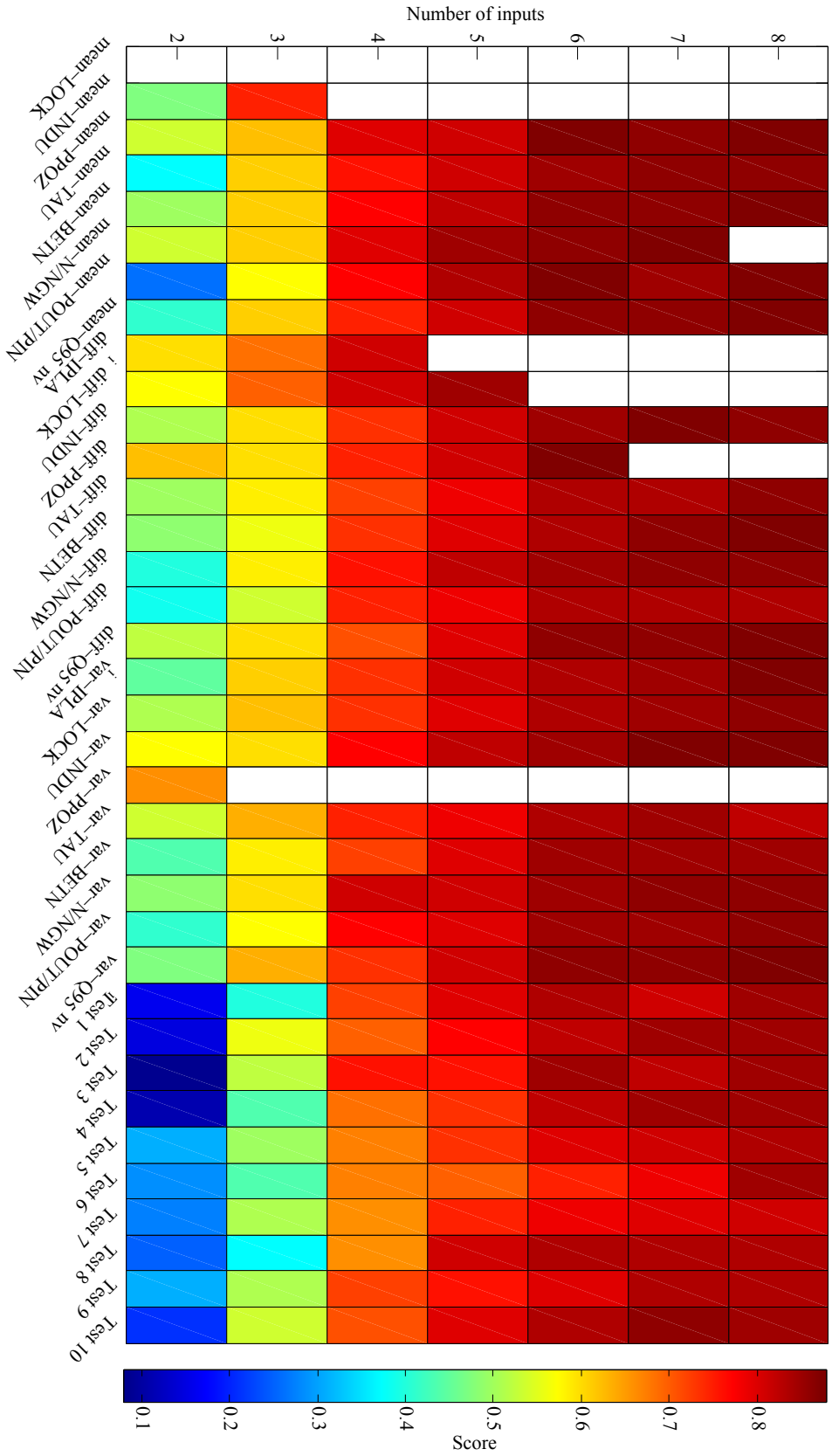


Figure 6.13: The output of the Forward Feature Selection. The selected variables have white background, the color of each cell corresponds to the score for the given combination of the parameters, each combination of the parameters was solved 10× and the final score in this figure is the mean score.

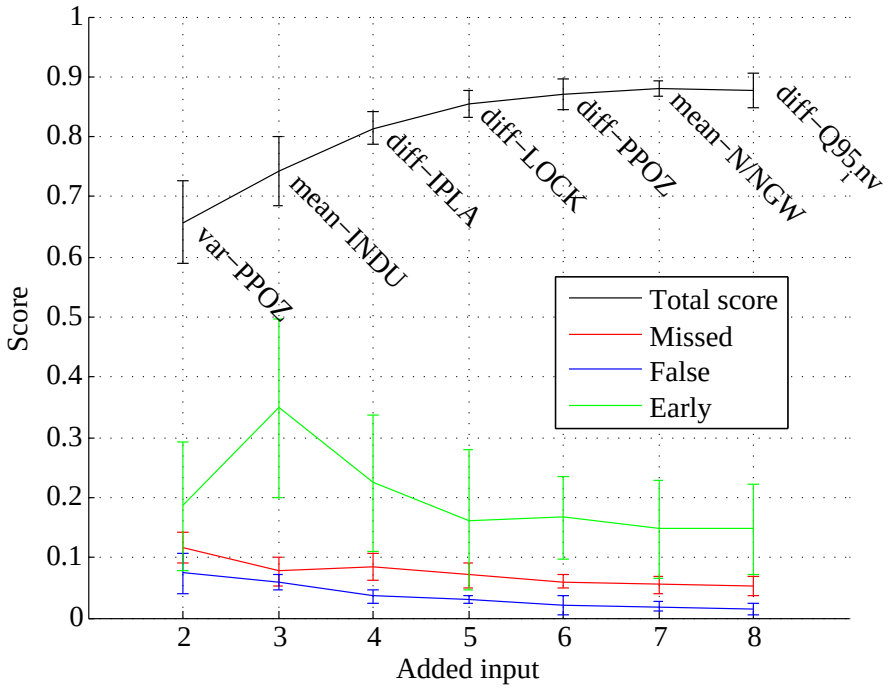


Figure 6.14: Dependence of the score on the number of the added dimensions using the Forward Feature Selection. Note that the dimension order for more than 4 inputs is unreliable due to high uncertainties.

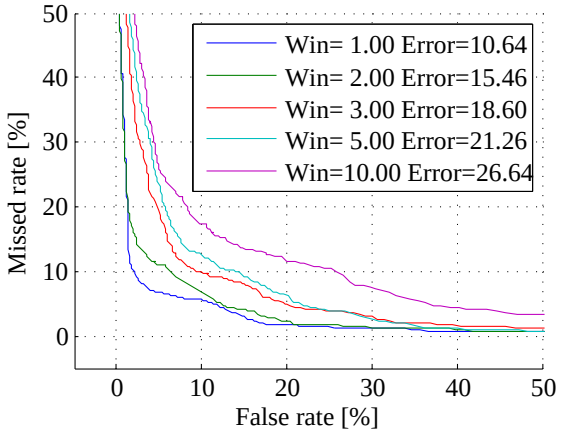


Figure 6.15: The Detection Error Tradeoff (DET) curves for the output smoothed by median filter over n time window of 30 ms time length

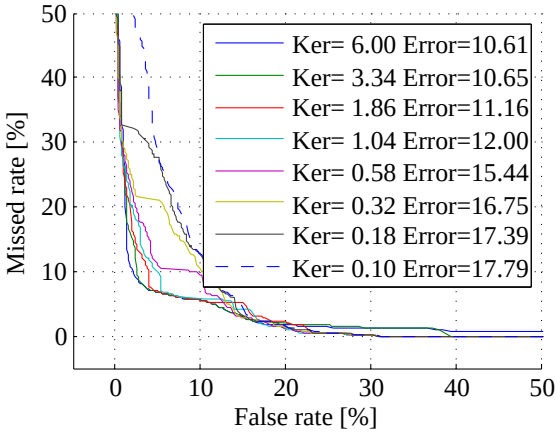


Figure 6.16: The Detection Error Tradeoff (DET) curves for the output smoothed by the weighted moving average with different damping constants $\exp(-\mathrm{ker})$

6.12 Results of Disruption Prediction System

6.12.1 Unintentional Disruptions

The unintentional disruptions are naturally arising disruptions caused either by technical issues or physical limits. Therefore, this type of the disruptions is the most important for the prediction. The training and validations sets are described in detail in the Section 6.3. 178 disruptive and 1246 nondisruptive shots were used for the training and internal validation. The number of the training points is >6000000, however only approximately 30000 points were extracted from this set as was explained in the Section 6.9.2.

Totally, 20 different models were trained, each with a different combination of the training/validation shots that were selected from the previously mentioned training set. The trained models had from 500 to 1500 support vectors. Therefore, the number of SV is equal to 3-5% of the used training points. The computer time needed for one time-slice prediction was $\approx 10\,\mu$s.

The optimal threshold was selected to maximize the score on the training dataset with the boundary condition FA$\leq$5% and the optimal threshold was selected individually for each of the models.

The results were analyzed over 1252 non-disruptive shots and 75 disruptive shots from 3 campaigns that were similar to the training set - C24, C25, C27b. The final score[6] was 88±2% and the detailed results are in the following table.

27%	JPS missed
6±2%	Missed alarms
2.0±0.8%	False alarms
11±3%	Early alarms

Table 6.8: The overall results for the unintentional disruptions. Training campaigns: C19–C22, Testing campaigns: C24,C25,C27b

The SVM predictor score (92±2) was significantly more successful than the JPS (JET Protection System) score (<73%). The number of the missed alarms and false alarms are connected: their sum is often near around 10%.

It should be noted that the premature alarms occurs relatively more often than the false alarms. It can be explained either by long precursor phase, especially locked modes can appear long time before the disruption, or the shot evolution gets near to the disruptive limits but the disruptions itself appears later.

The confusion matrix in the following table shows how many points were misfitted in each of the testing classes. It is only an example for the first model. Non-disruptive data are the class 1 and the class 2 is disruptive. The matrix rows are normalized on number of the member in each of the classes:

[6] Score = (FA+PA)/N_shots*100+ %MA

	1	2
1	99.97%	0.03%
2	36.37%	63.63%

Note that from the previous table it follows that the MA score of 6% (Tab 6.8) was achieved although 36% of the disruptive points were not recognized and the FA 2.0% corresponds to 0.03% of the incorrectly recognized non-disruptive data. It was expected that the final score is not proportional to the "zero-one" error.

Important information is visible from the Detection error tradeoff (DET) dependence in Fig 6.17. The right choice of the threshold can improve the results but some disruptions are almost impossible to identify and the false rate must be higher than 95% to recognize all the disruptions (for current input data set).

In Fig 6.18 the evolution of the recognized disruptions by the SVM (red) and by the JET protection system (black) is shown. The SVM classifier outperforms the JPS from time -0.7 s before the disruption and also the number of premature alarm is very low. It should be noted that some patterns in the curve are similar and therefore the classifiers observe similar phenomena in the plasma.

And finally, the most important graph is the evolution of the score over all campaigns that is shown in Fig 6.19. The score is usually above 85% and missed are below 5%. The campaigns C19–C22 were used for the training. The enormous number of the false alarms in the campaign C23 could be caused by the error field coils that were used in this campaign more often.

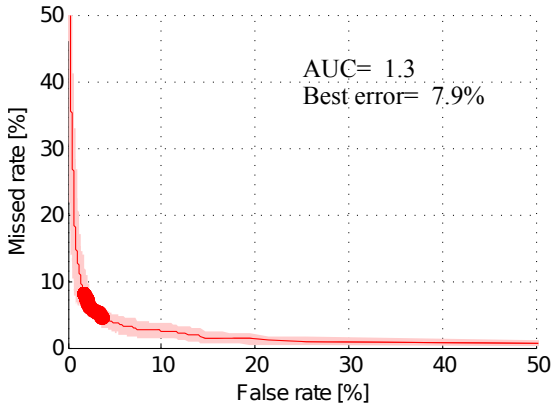


Figure 6.17: Detection error tradeoff (DET) for the unintentional disruptions. Red dots denoted the optimal score under the condition FA≤5%

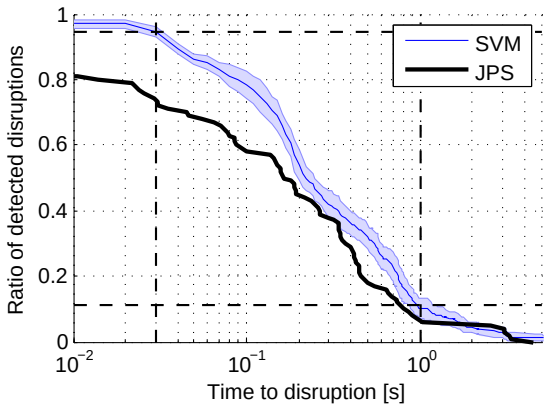


Figure 6.18: The time evolution of the detected unintentional disruptions for SVM and JPS (JET prediction system) from the campaigns C24,C25,C27b

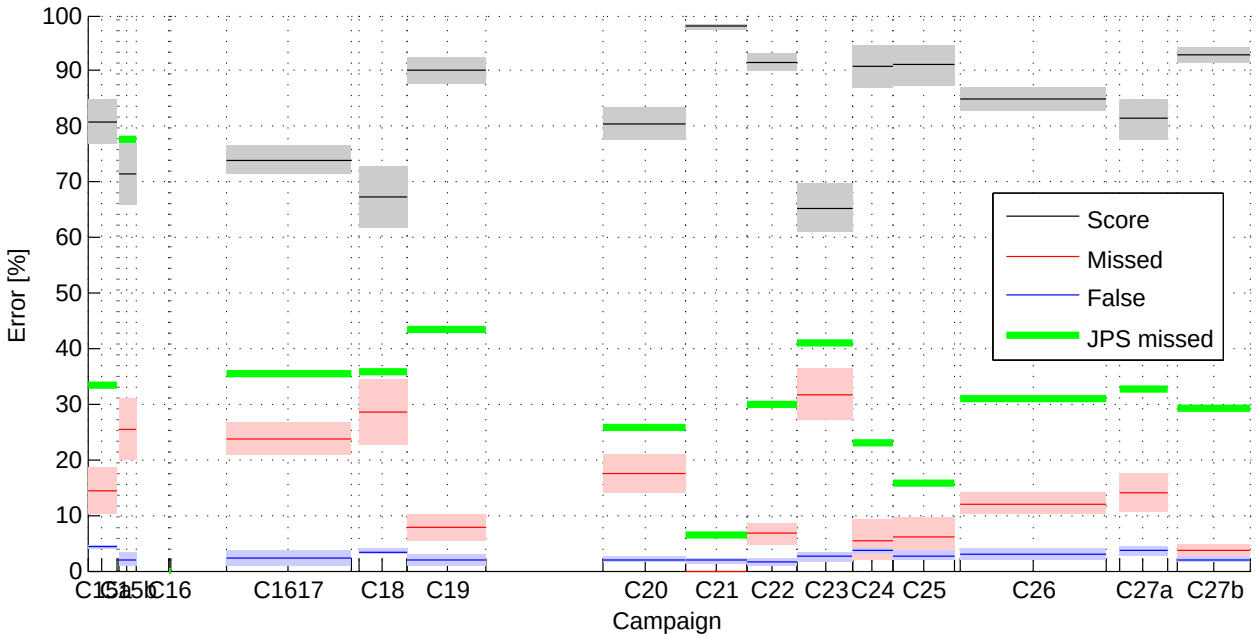


Figure 6.19: The total score over all campaigns only for the unintentional disruptions. Campaigns C19–C22 were used for training. Length of line corresponds to the number of shots in each campaign.

6.12.2 Intentional Disruption

Intentional disruptions are artificially caused by operators at the end of some shots. The plasma is destabilized by a gas puff and it causes disruption. The possibility of the training based on the intentional disruptions only is in particular important because it is expected that the ITER should withstand several hundreds of this disruptions [35].

The intentional disruptions are always caused under conditions that are artificially selected. Therefore no high current intentional disruptions are in JET database in campaigns C15-C27. Problem of the intentional disruptions is that the disruption precursors appear very shortly before the disruptions. It is the reason why the disruptive points set is very limited in spite of a high number of the intentional disruptions that were produced. Another issue can be similarity of the intentional disruptions that causes over-fitting if the intentional disruptions are used for the training. Finally, the disruptive region of the intentional and unintentional disruptions are not the same as was shown in the Fig 6.3.

We have tested two options, how to train the learning machines with the unintentional and intentional learning set.

1. Training with unintentional, testing with intentional
2. Training with intentional, testing with unintentional

In the first step the learning machine was trained with the unintentional disruptions and tested with intentional disruptions. Therefore, the trained models from the previous section were used for prediction of the intentional disruptions. The intentional disruptions for testing were used from campaigns C19–C22, C24, C25, C27b. The results are in the following table:

94%	JPS missed
45±3%	Missed alarms
2±1%	False alarms
0.1±0.1%	Early alarms

Table 6.9: The final score for the intentional disruptions from the campaigns C19–C22, C24, C25, C27b. Note that the percentage of the missed alarms is significantly higher than for the MA for the unintentional disruptions (Tab. 6.8)

The SVM results are good compared to the JPS that missed 94% disruptions. It implies that the intentional disruptions cause less locked modes than the unintentional and the precursor phase is very short as can be seen in the Fig. 6.20 that shows the detection time evolution. The MA of the SVM rate is also very high but it is mainly caused by the short precursor time. This claim is confirmed in the Section 6.16.

It is interesting that some intentional disruptions were detected more than 100 ms before disruption although the precursor phase is shorter. This could be caused because of the detection of changes in the plasma current and Q95 leading to disruption, i.e the Q95 was decreased very near to 2 just before some of the intentional disruptions.

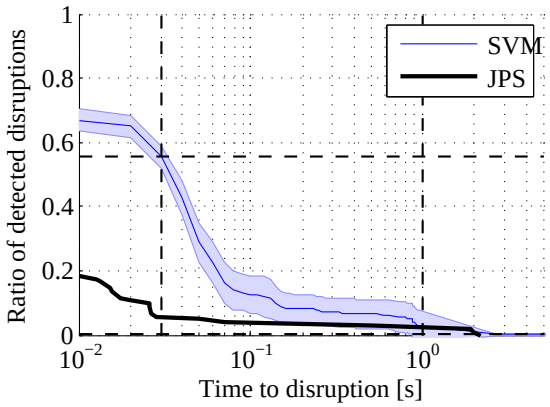


Figure 6.20: The time evolution of the triggered disruptions for the intentional testing set. Majority of the disruptions were detected around 50 ms before the disruption.

In the second step, all unintentional disruptions were removed from the training and validation set and only the intentional disruptions were used for learning from campaign C19–C22, C24-C25, C27b. Nondisruptive shots were selected only from the campaigns C19–C22. The main problem is that the intentional disruptions are usually quite similar to each other and therefore the SVM can reach severe over-fitting although the training and validation set belongs to different campaigns. The score for the validation set can be more than 90% although prediction capability for the unintentional disruptions is very low.

Unintentional disruptions from campaigns C19–C22,C24,C25,C27b were used for testing. The results are in Tab 6.10.

27%	JPS missed
26±16%	Missed alarms
4.6±0.5%	False alarms
10±5%	Early alarms

Table 6.10: The table of the results for the unintentional disruptions if the algorithm is trained on the intentional disruptions from all similar campaigns (C19–C22, C24,C25,C27b)

The number of the missed alarms is very high, hence the final score is very low (70±20%). The final score is on average comparable with the results of the cross-tokamak learning [9] where they reached score 68%. The Fig. 6.21 shows that the mean prediction time is much worse compared to the previous section, in particular the growth rate is slower compared to the Fig. 6.18.

The Fig 6.21 shows that some model could reach good results but the models are not reliable and without unintentional disruptions it is not possible to determine the best model. The final score is on average similar to the JPS.

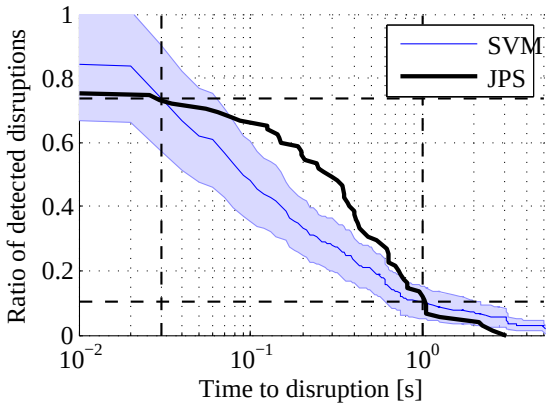


Figure 6.21: Time evolution of triggered disruptions for unintentional testing set and intentional training set. The prediction time and score are worse compared to the unintentional training set (see Fig. 6.18). Moreover, the reliability of the prediction is very low.

6.12.3 High-Low Current

The next topic in this work is prediction of the high current disruptions (above 2 MA) based on the low current shots. This task is especially important for ITER where the high current disruptions could cause severe damage of plasma facing components or even of the tokamak vessel. Moreover, the ITER disruption database will be significantly limited at the beginning and it will contain only the low current shots and disruptions. The threshold for low/high current disruptions on JET was selected at 2 MA at the time of the disruption. The training set contained only the low current shots (disruptive and nondisruptive) and the testing dataset contained only the high current disruptions and all non-disruptive shots.

	Training data	**Testing data**
	Low current disr.	High current disr.
Unknown	23	24
ASD	9	5
GWL	4	1
IMC	21	1
ITB	5	2
NC	24	7
NTM	7	7
Total	93	47

Table 6.11: Types of the high current disruptions used for the training

The final score is quite low (80±4%) mainly because of the high number of the missed alarms that reached 16%. The JPS missed alarms score was only 14%. However, the early alarms of JPS are 27% and therefore it rather triggers all the high current shots.

On the contrary, the score for the HC shots that were trained with all disruptions (Tab. 6.13 and the score for all types of shots and disruptions (Tab 6.8 are within errors similar.

14%	JPS missed
29%	JPS early
16±4%	Missed alarms
4.1±0.4%	False alarms
9±2%	Early alarms

Table 6.12: Final results for low-current (LC) disruptive / nondisruptive shots used for training, and high current (HC) disruptive and all nondisruptive shots used for testing

8±2%	Missed alarms
3.7±0.7%	False alarms
6±2%	Early alarms

Table 6.13: Score for all disruptive shots used for train, HC disruptions for test

The evolution of the detected ratio for SVM trained with all disruptions and only the unintentional disruptions are in the Fig 6.22.

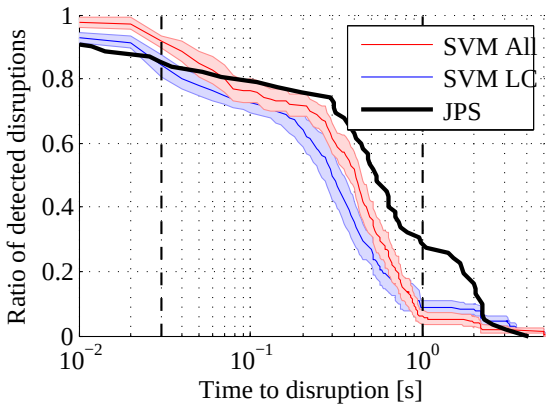


Figure 6.22: The time evolution of the triggered alarms for the SVM trained with all type of the shots (SVM All) and for the SVM trained only with the low current shots (SVM LC). The testing set contained only the HC disruptions and all types of the nondisruptive shots.

The SVM detection is only slightly better than the JPS (JET prediction system). However the JPS has still much more early alarms and possibly also false alarms.

6.13 One Class SVM

In the previous sections possible ways how to predict disruptions on limited datasets were investigated. Another interesting possibility is to remove all the disruptive data from the training set and use only the non-disruptive and One Class SVM [22] for learning. An introduction to the One Class SVM is in the Section 3.2.3. Basically, it is a modified version of the ν-SVM algorithm and it can be used to detect outliers and therefore the

26.0%	JPS system missed
7.0%	JPS system early
12±4%	Early alarms
6±2%	Missed alarms
4.3±0.5%	False alarms

Table 6.14: Total score of One Class SVM applied on testing campaigns C24,C25,C27b

disruptive data. However, there is still need to identify three learning constants: kernel size, ratio of outliers (ν) and threshold. The kernel size can be a priory selected σ=1 because of the right normalization as it is shown in Fig 6.23. and the dependence on ν is also weak. The optimal value is $\nu = 0.1$ but value near to the optimal is from 10^{-3} to 0.5.

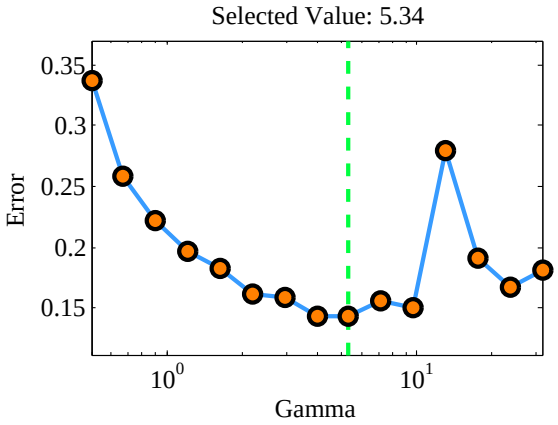


Figure 6.23: An example of the cross-validation error for the One Class SVM, the error is equal to MA+FA+AUC

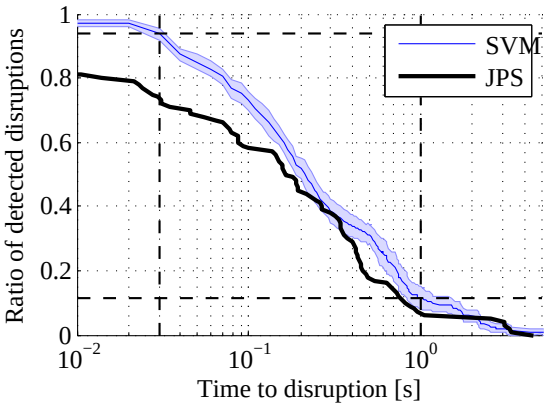


Figure 6.24: The time evolution of the score for the One Class SVM algorithm tested on campaigns C24,C25,C27b.

The only problem is the selection of the best threshold. There are two possibilities: use some disruptive data for testing or define the maximal acceptable False Rate and select the threshold to satisfy this condition. DET (Detection Error Tradeoff) curve for the One Class SVM is shown in the Fig 6.24 and if the maximal acceptable false rate would be chosen 5% then the missed rate would be 6%.

The results are very good. The final score is 89±2%. This is even slightly better than the results of the two class SVM 12±2% presented in the Section 6.12.1. On the other hand, the main disadvantage was a huge number of the support vectors that was from 10000 to 20000 whereas the two class SVM usually has around 1000. Therefore, the prediction time for the one time slice was approximately 0.1 ms.

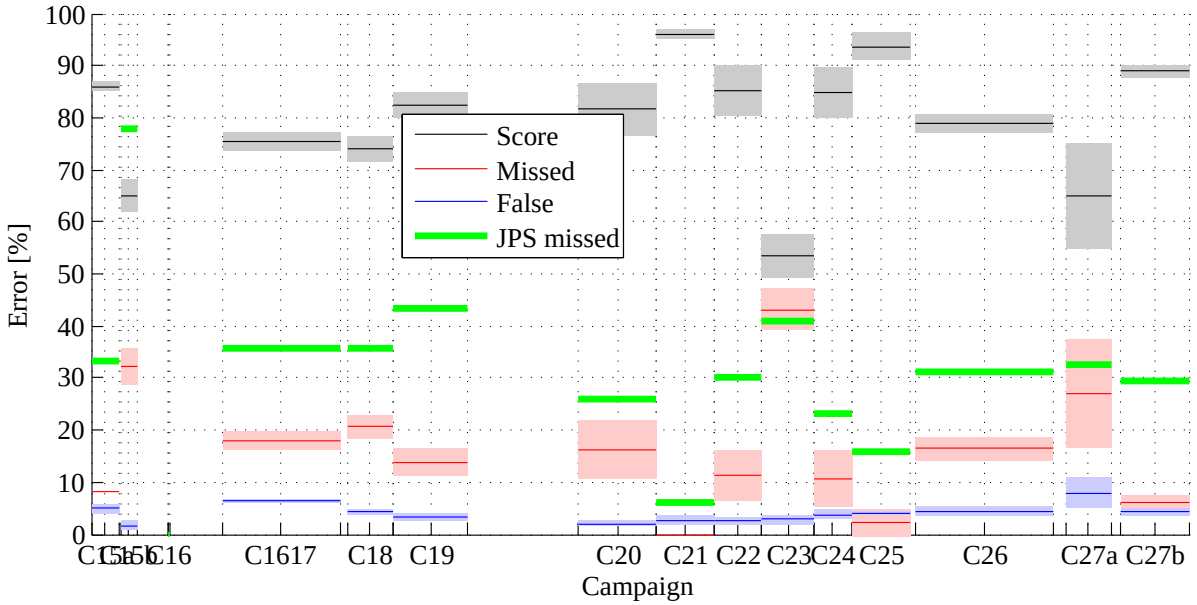


Figure 6.25: Total score over all used campaigns of One Class SVM. Campaigns C19–C22 were used for training. Length of line corresponds to the number of shots in each campaign.

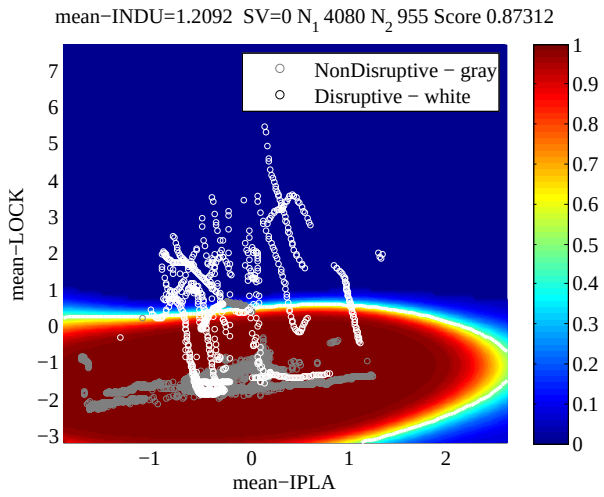


Figure 6.26: An example of the cut through the plasma parameters feature space with the two class SVM prediction on background. The background color corresponds to the probability estiamtion.

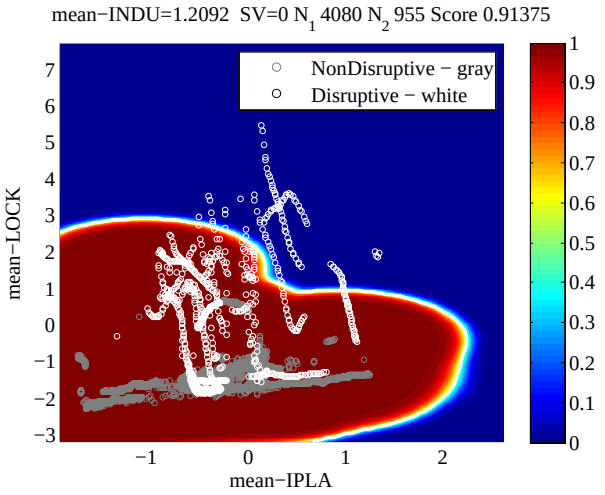


Figure 6.27: An example of the cut through the plasma parameters feature space with one-class SVM prediction on the background. LibSVM is not capable of finding probability estimation, therefore the output was only rescaled similarly to the two class example.

The total final score over the testing and training dimensions is reported in Fig 6.25. However, the score over all the campaigns is very similar to the two class SVM, see Fig. 6.19. It can be said that the score do not significantly dependents on the learning machine type and the main dependence is on the testing data and disruptions types that are in each campaign.

Moreover, the One class SVM can be used for novelty detection. If the model from the

26.0%	JPS system missed
7.0%	JPS system early
4±1%	Early alarms
7±1%	Missed alarms
3±1%	False alarms

Table 6.15: Total score of RVM applied on testing campaigns C24,C25,C27b

training campaigns is used for prediction the number of missed alarms should correspond to the similarity of campaigns. The second possibility is estimation similarity between campaigns from the values of threshold.

6.14 Relevance Vector Machine (RVM)

Furthermore, we have tested a Bayesian classification algorithm relevance vector machine introduced in Section 3.3. The main advantage of the RVM model compared to the SVM is lower number of support (relevance) vectors and ARD (automatic relevance determination). The RVM models are much less complex, therefore the models should avoid overfitting. Moreover, the prediction is significantly faster compared to the SVM because number of vectors used for prediction is 100–1000 times lower.

The success rate reached 90±2%. This is a very good results, although the number of relevance vectors was less than 50 compared to the SVM models with more than 1000 support vectors. The next reason, why it is an unexpected results, is that the relevance vector machine prefers to fit a majority of points and ignores the points on the border on contrary to SVM that uses mainly the information from the points around boundary. On the other hand, the training of the RVM is much more demanding on computational resources mainly the memory use and without significant improvements of the SparseBayes [16] algorithm it would be impossible to generate these results.

6.15 Learning on Small Dataset

In particular, a relevant problem for ITER is a small training database of the disruptive shots at the beginning. Therefore, it necessary to study the behavior of the prediction on small datasets to estimate the usability of the learning machines. For this study, the algorithm was learned on datasets containing from 10 to 100 disruptive shots and 20× more nondisruptive shots because the requested disruptivity should be under 5% for ITER. The shots were randomly selected from the training campaigns C19–C22 and tested on similar campaigns C24,C25,C27b. The results are shown in Fig 6.28.

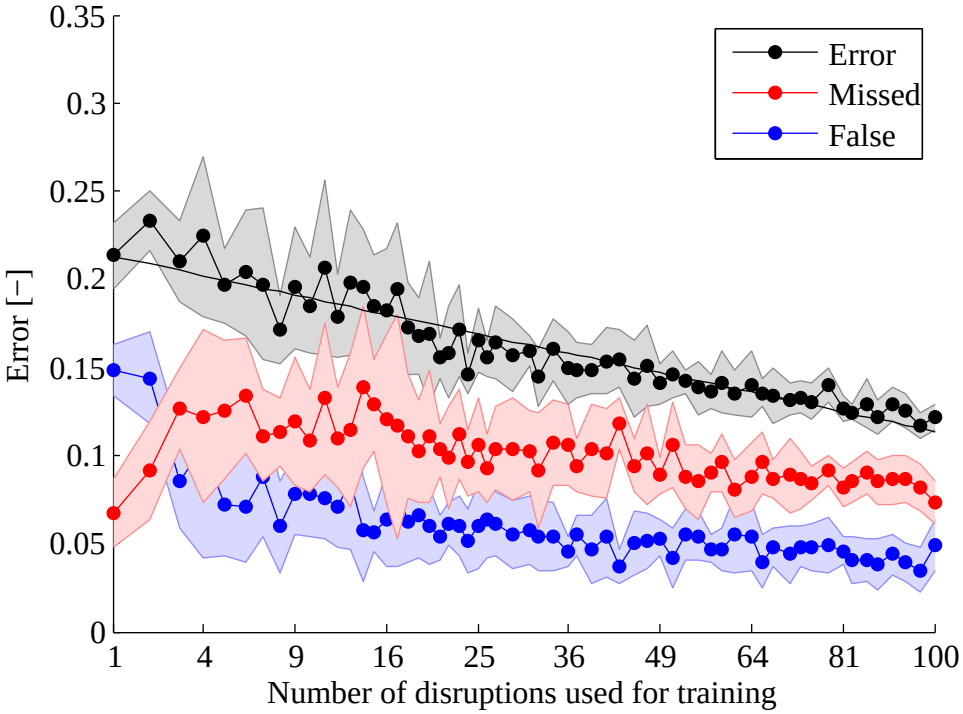


Figure 6.28: The overall dependence of the total error, FA and MA on the number of the disruptive and nondisruptive shots in the training database. The x-scale is quadratic to linearize the decline. The deviations corresponds to 67% confidence and models were tested on all shots from campaigns C24,C25,C27b.

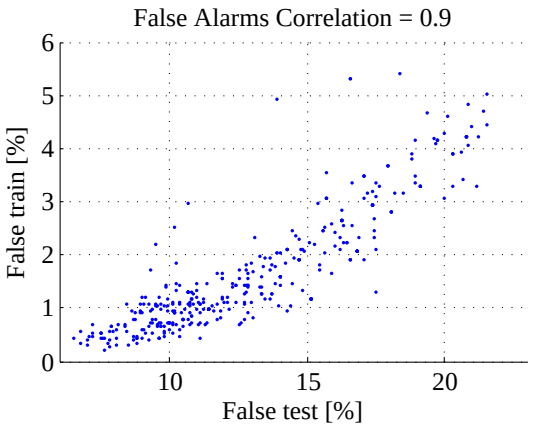


Figure 6.29: False alarms from all the tested models on the training and testing datasets. The correlation is very high Corr=0.9

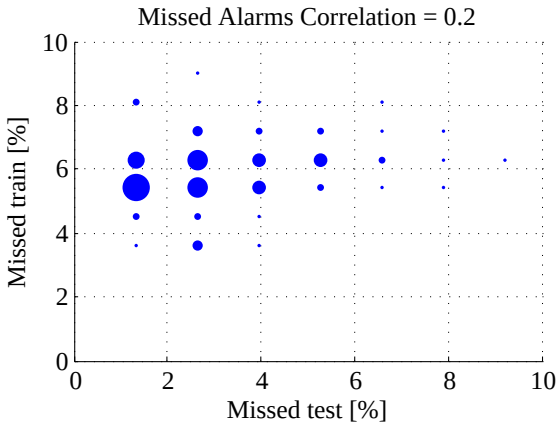


Figure 6.30: Missed alarms from all the tested models on the training and testing datasets. Size of the points correspond to number of models with the score. The correlation is very low Corr=0.2

There are several important things to note. Firstly, the total error is decreasing very slowly and the lower limit of the score stays almost constant, moreover the false alarms rate is declining also very slowly. Finally, the main effect of the increase number of points

is decreased deviation.

From the results follows that it is possible to find a good model even with a low number of shots but only under assumption that the shots were randomly selected from the full database. If the shots and disruptions in the training database would be very similar (data would be more correlated) the prediction capability of the models should be expected to be lower.

Nevertheless, some ways how to slightly improve the score exist:

- add artificial data as was proposed in section 6.9.4.
- use iterative solution to remove outliers

An associated problem to the learning on small datasets is the ability to predict to estimate the score on a new campaigns from the training dataset. It is interesting to study the correlation between results on the training campaigns and on the testing campaigns. This study was performed using 400 models generated on testing campaigns C19–C22 and tested on campaigns C24,C25,C27b. Each model was generated as the most optimal for a random subset of shots from the training campaigns that contained approximately 500 shots nondisruptive and all disruptive shots. Consequently, the model was tested on all shots from the training and all shots from the testing campaigns. The results are shown in Fig. 6.29, 6.30.

The correlation between the false alarms is 0.9 (Fig. 6.29), therefore the results on the training and testing campaigns correspond very well if the campaigns are similar. On the contrary, correlation between the missed alarms (Fig. 6.30) is low only 0.2. It implies that the extrapolation of the MA score from the testing campaigns to new campaigns is not reliable. Nevertheless, this was not unexpected result because the missed alarms score depends mainly on the types of the disruptions (see Section 6.16) in each campaigns and the false alarms depends on the precise and complete description of the nondisruptive feature space.

6.16 Properties of Disruptions

The shots in the used disruption database were manually classified into several groups [45]. This allowed as to study the properties of each disruption type separately. All the classified types of the disruptions and their representation are in the Tab. 6.16.

Firstly, we have focused on the detection time evolution and the missed/falsed alarm score for each of the disruption type. All unintentional disruptions from campaigns C15 to C27b were used in order to achieve sufficient statistical reliability, however some types are still very rare. The number of the used disruptions are in the Tab 6.16. 32% percent of the unintentional disruptions were not classified. These non-classified disruptions are on average detected later and missed more often.

The evolution of the disruptions that were recognized by our SVM algorithm trained on the unintentional dataset are in Fig 6.31 and the same for the JPS is plotted in

Type		#
Too little auxiliary power	ASD	51
Greenwald limit	GWL	10
Impurity control problem	IMC	77
Too fast a current ramp-up	IP	15
Too strong internal transport barrier	ITB	13
Too low density (and low q)	LON	21
Density control problem	NC	74
Neo-classical tearing mode	NTM	35
Nonclassified	???	142
Intentional	INT	234

Table 6.16: Types of disruptions in used database, campaigns C15-C27

the Fig 6.32. The simplest detectable types are LON (Too low density and low q) and NC (Density control problem). These disruption were often detected long time before the disruption and often these disruptions were triggered as an early alarm. On the other hand, the worst predictable types are intentional disruptions and ITB (Too strong internal transport barrier) although this type is rare in the used campaigns C15-C27. It should be noted that that the time evolution of both types is very similar for both systems SVM and JPS.

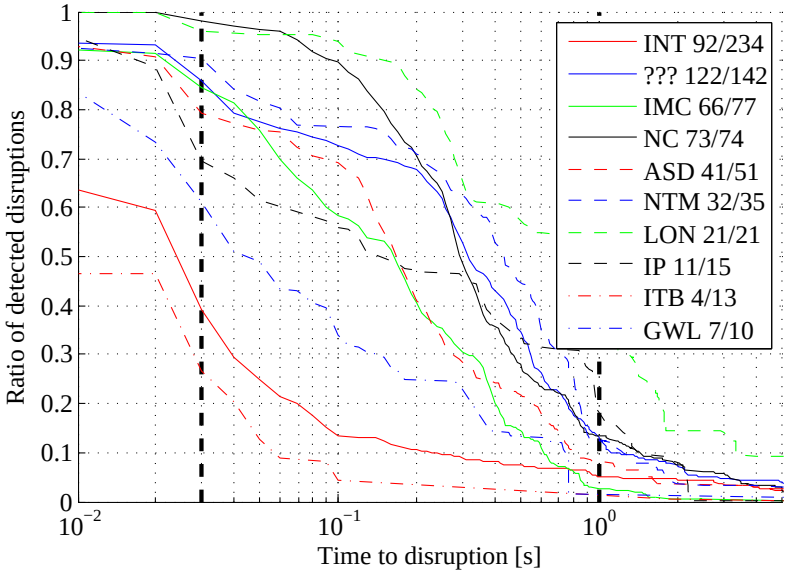


Figure 6.31: The time evolution of the detected disruptions for each of the classified disruptive type using SVM. The legend also shows number of correctly predicted disruption from all disruptions with given type.

The score of the SVM is almost in all cases better than the JPS (JET prediction system). It is even more visible in Fig 6.33, where the curves from Fig 6.31, 6.32 are subtracted. The only type that was detected by JPS significantly better are NTMs. However, it not clear advantage because it probably caused many false alarms. On the other hand, SVM was significanly better in prediction of IMC (Impurity control problem) that was detected

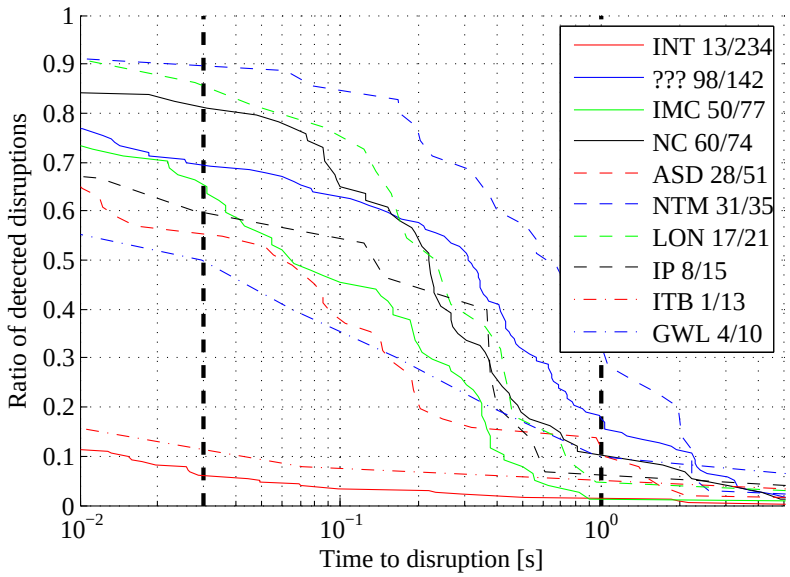


Figure 6.32: Time evolution of the detected disruptions for each of the classified disruptive type using JPS. The legend also shows number of correctly predicted disruption from all disruptions with given type.

long time before disruption.

Further, the SVM detection ratio grow faster, compared to the JPS from the time 50 ms before the disruption. It seems that the JPS rate exceeds the SVM in last 20 ms, however it probably caused by missing signals from some of the used inputs short time before the disruption. This prevented the SVM algorithm performing the prediction, Moreover, note that the time resolution of majority inputs saved in the JET database is usually lower than 10 ms (see 6.6).

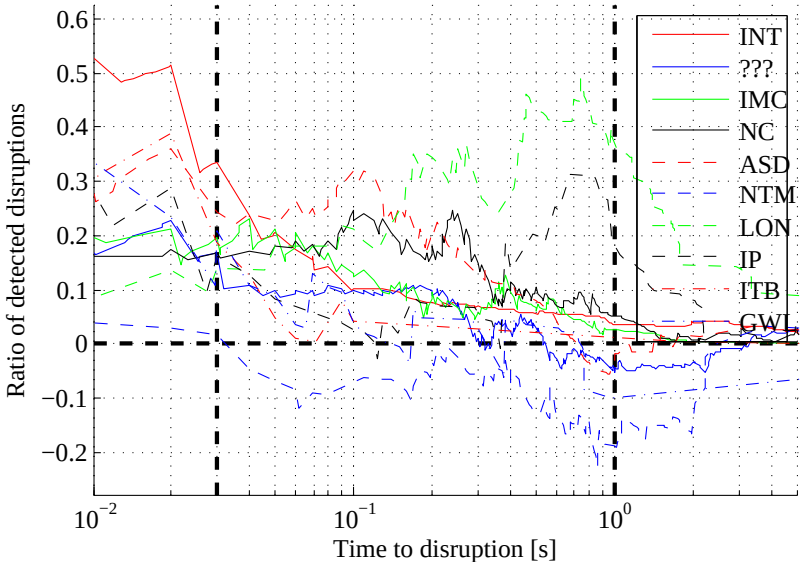


Figure 6.33: JPS subtracted from SVM detection rate evolution. Positive result mean that the SVM model is better.

The next investigated property was time dependence of each disruptive type on the di-

agnostic inputs. We have trained the models with all inputs and than one dimension was replaced by its median to remove the influence. It would be also possible to retrain the models with one dimension missing, however it can introduce unexpected effects to the models.

Examples of the resulting probability evolution after the subtraction of the most important dimensions are shown in the Figs 6.34, 6.35. It is visible that the locked modes are very important as an early precursor almost for all disruption types (Fig 6.34), the only exception are the intentional disruptions that were not affected. On the other hand, the density control problem (NC) disruptions significantly depends on the locked modes.

The second example (Fig 6.35) are variations in vertical position. Compared to the locked modes, the intentional disruptions are very sensitive and also the unclassified disruptions (???). Other examples and also the less frequent disruptions type are not shown because the results were less credible. Moreover, the locked modes and plasma position are "trigger" inputs. Other dimension serves mainly as a variables determining sensitivity (threshold) of these "triggers" but without the triggers are the dimensions useless.

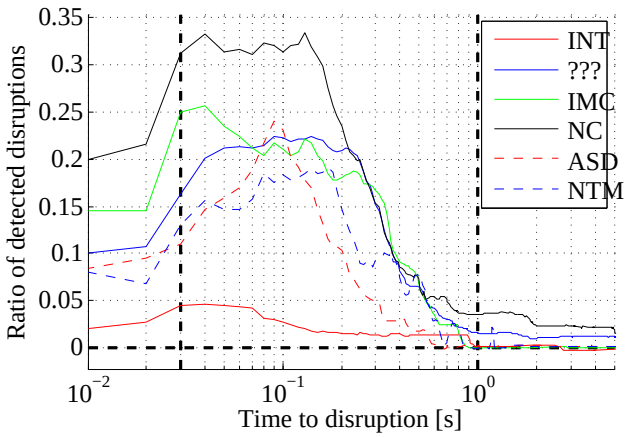


Figure 6.34: The prediction ratio evolution when the influence of locked modes was subtracted.

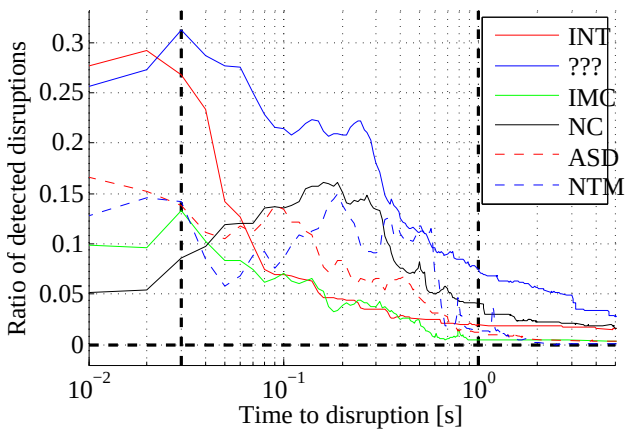


Figure 6.35: The prediction ratio evolution when the when the influence of variation of the plasma position was subtracted.

Summary

The main aim of this work was selection of a suitable model for plasma disruption prediction in order to prevent possible damage of plasma facing components in tokamaks. This will be mainly an issue for the future tokamak where a disruption can cause sever damage. However, if the disruption is predicted early enough, it is possible to use different methods to avoid the fail or at least mitigate the damage. The ideal goal is prediction of 95% disruptions mitigate the heat load on 1/10 of the original value.

This work was focused on disruption prediction on the currently largest tokamak JET. Discharges from this tokamak over three years of operation were used for training and testing of the models. Although, the models were not directly developed for the real-time prediction, only the learning machines algorithms with a possibility of the real-time prediction and also the real-time accessible plasma parameters were used. The following kernel based classification algorithms were tested: Support vector machine (one/two class), Relevance vector machine. Other widely used kernel based classification algorithms such as the Logistic regression are not suitable for the real-time prediction because of slow prediction. The real-time accessible plasma parameters were preferred. Furthermore, some dimensionless plasma parameters (see Section 6.1) were added in order to investigate a possibility of the cross-tokamak learning, although the cross-tokamak learning itself was out of this work.

Many possible problems caused by the data preprocessing and results post-processing were investigated and possible solutions were proposed.

Firstly, in Section 6.10 we have proved that a careful feature selection reduces the complexity of the models and can improve the results. The most important variables are locked modes, plasma position derivative, internal inductance, plasma current derivative. Moreover, in the Section 6.10 it is also shown that the parameters connected with the disruptive limits (Section 6.2) are not the essential for the disruption prediction. This feature selection used the largest training/testing datasets ever used and several different preprocessings. However, in the next step it is neccessary to test even more different inputs as a possible precursors.

Furthermore, different choices of the input dataset were tested in order to simulate complications that can appear during building a sufficient training database for the ITER tokamak. Firstly, a database of unintended disruptions (not caused by human intention) were used for training and testing. The total success rate is 92±2% where the total error is defined as 100 - missed alarm [%] + false alarms [%] + early alarm [%] ×disruptivity (see Section 6.9.1). The score was obtained for testing campaigns C24,C25,C27b.

In the next step, models trained only with the intentional disruptions were tested. The intentional disruptions are caused by an intended human action at the end of some discharges. The possibility of the training on the intentional disruptions is in particular important because it is expected that the ITER should withstand several hundreds of this disruptions compared to only several tens of unintentional. Unfortunately, our results indicate that the intentional disruptions are very different from the unintentional disruptions (Section 6.12.2). Therefore, results of the models trained on the intentional disruptions and tested in the unintentional disruptions are quite poor. The resulting success rate is only 70±20% and the stability of the prediction is very low.

An extrapolation from the low current shots and disruptions to the high plasma current was investigated. Again, the ITER discharges will be only operations with low-current at the beginning of operation. This option is more optimistic compared to the previous one. The score in case of the low current shots used to train and the high current to test is 80±4%. However, this result is still far from the optimal results using the full dataset of the unintentional disruptions.

Besides testing different types of shots/disruptions, learning on a database with a variable number of discharges was investigated (see Section 6.15). The conclusion is that the drawback of the small database is the creation of a good model but the main issue is the selection between many similar and indistinguishable (with a small testing database) models. Moreover, we have shown that the correlation between missed alarm on training and testing dataset is very low and the reliability of the success rate prediction for new campaigns from the score of the previous ones. This is a serious issue that has never been investigated before.

The results indicate that the effect of the the learning machine selection is only minor compared to the influence of the different disruption types in the new campaigns. In other words, it is not possible to compare success rates of different learning algorithms tested on different campaigns or even different tokamaks. In this work, we have shown that the input data are the main factor determining the score and not the used learning algorithm. A better selection of variables and model can improve the success rate, however limits on the minimal number of the missed alarms exist because some disruptions has no or too short precursor phase. This limits cannot be exceeded without a significant increase of the false alarms or without new input diagnostics.

Finally, it is important not to use the learning machine as a black box but carefully investigate every aspect of the training process in order to try to understand plasma properties that allow the disruption prediction. A score obtained without a careful evaluation of data, estimation of variation of the results and other possible issues can be unreliable as shown in Sections 6.6, 6.12.2 and 6.15. Moreover, the variation (precision) of the estimated score of disruption prediction is always limited by a low number of the tested disruptions as it is shown in Section 6.9.3, although the number of tested points is huge. Furthermore, our estimation of the uncertainty of the score can include only variation caused by the training set and not the bias caused by the testing set. This is the reason why score for all shots over the three years of JET operation is shown for the important cases (see Figs 6.19, 6.25). On the other hand, many campaigns were significantly different from the training ones and major changes in diagnostics were made over the period [41].

In my opinion, any significant improvement of the tested results cannot be reached without introduction of some new important diagnostics that are not currently real-time accessible such as plasma pressure profile, ion temperature (profile), plasma rotation profile, plasma emissivity profile, impurity content, ... Unfortunately, it was not possible to test these "non-real-time" inputs, because these inputs are not accessible for many shots/disruptions at the JET tokamak. Therefore, inclusion of these parameters to the used database would significantly limit the number of the accessible shots/disruptions.

On the other hand, many real-time algorithms for plasma equilibrium reconstruction are being developed [50, 51] and plasma emissivity profile [52] that should allow to obtain the pressure profile and a more precise plasma current profile with <1 ms delay. Furthermore, an improved NBI (neutral beam injection) was recently installed at ASDEX tokamak that allows to measure ion temperature profile, safety factor profile[7] and plasma rotation in real-time. It will be necessary to wait a few years until a sufficient database of shots with the new diagnostics will be created, however the first results can be expected in 2012-2013. Finally, the newest results from the JET tokamak shows that installation of a new Beryllium wall slowed the evolution of disruptions although no comprehensive study that would show effect on the "difficult disruptions" has been published yet and the tested database was rather small.

In future, the development should focus on the new diagnostics and mainly on study of the "difficult disruptions". Hopefully, in near future the JET database will be able to save the important variables in more than 10 ms time resolution for all shots/disruptions, not only on request. This could allow to use more advanced preprocessing such as wavelets optimized for disruptions prediction. Furthermore, it is necessary to study the cross-tokamak prediction in order to estimate the best models for the future tokamaks (ITER, DEMO) and perform computer simulations of the conditions in tokamak to allow the necessary adjustments of the prediction models before the tokamak ITER is finished. Virtual diagnostics simulators are already being developed [53, 54] however the disruptions are "outliers" compared to the ordinary shots and it will be a great challenge to simulate their signals.

In summary, nearly 94% disruptions in tokamak JET can be now predicted early enough in order to mitigate the damage. It is very near to the value 95% that was requested for the ITER tokamak [37]. However, the expected results for the ITER are worse, therefore disruptions need further research to achieve the prediction rate close to 95% with a very small training database.

[7] measurement of magnetic field helicity

Bibliography

[1] John Wesson. *Tokamaks (Oxford Engineering Science Series)*. Oxford University Press, USA, 1987. ISBN: 0198563280.

[2] JW Farthing et al. "Data Management at JET with a look forward to ITER". In: *International Conference on Accelerator and Large Experimental Physics Control Systems*. 2006.

[3] E. Ronchi et al. "Neural networks based neutron emissivity tomography at JET with real-time capabilities". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 613.2 (2010), 295–303. DOI: 10.1016/j.nima.2009.12.023.

[4] J. Vega et al. "Automated estimation of L/H transition times at JET by combining Bayesian statistics and support vector machines". In: *Nuclear Fusion* 49 (2009), p. 085023. DOI: 10.1088/0029-5515/49/8/085023.

[5] S. González et al. "Support vector machine-based feature extractor for L/H transitions in JET". In: *Review of Scientific Instruments* 81 (2010), 10E123. DOI: 10.1063/1.3502327.

[6] B. Cannas et al. "Disruption forecasting at JET using neural networks". In: *Nuclear fusion* 44 (2004), p. 68. DOI: 10.1088/0029-5515/44/1/008.

[7] F. Ryter et al. "H-mode power threshold and transition in ASDEX Upgrade". In: *Plasma physics and controlled fusion* 40 (1998), p. 725. DOI: 10.1088/0741-3335/40/5/032.

[8] AR Polevoi, M. Shimada, and VS Mukhovatov. "ITER plasma performance assessment on the basis of newly-proposed scalings". In: *Plasma physics and controlled fusion* 48 (2006), p. 449. DOI: 10.1088/0741-3335/48/5A/S46.

[9] CG Windsor et al. "A cross-tokamak neural network disruption predictor for the JET and ASDEX Upgrade tokamaks". In: *Nuclear fusion* 45 (2005), p. 337. DOI: 10.1088/0029-5515/45/5/004.

[10] K.P. Murphy. "Dynamic bayesian networks: representation, inference and learning". PhD thesis. Citeseer, 2002. DOI: 10.1.1.129.7714.

[11] K. Pelckmans et al. "Handling missing values in support vector machine classifiers". In: *Neural Networks* 18.5-6 (2005), 684–692. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2005.06.025.

[12] M.E. Tipping. “Sparse Bayesian learning and the relevance vector machine”. In: *The Journal of Machine Learning Research* 1 (2001), 211–244. ISSN: 1532-4435. DOI: 10.1.1.25.1089.

[13] C.M. Bishop and M.E. Tipping. “Variational relevance vector machines”. In: *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. Citeseer. 2000, 46–53.

[14] C.M. Bishop. *Pattern recognition and machine learning*. Vol. 4. Springer New York, 2006. ISBN: 0387310732.

[15] M.E. Tipping. “Bayesian inference: An introduction to principles and practice in machine learning”. In: *Advanced lectures on machine Learning* (2004), 41–62. DOI: 10.1.1.85.3851.

[16] M.E. Tipping. “An Efficient Matlab Implementation of the Sparse Bayesian Modelling Algorithm”. In: (2009).

[17] K.P. Murphy. “Machine Learning: a Probabilistic Perspective”. In: (2012).

[18] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000. ISBN: 0387987800.

[19] J. Platt. “Probabilistic outputs for support vector machines”. In: *Bartlett P. Schoelkopf B. Schurmans D. Smola* (1999), 61–74. DOI: 10.1.1.41.1639.

[20] H.T. Lin, C.J. Lin, and R.C. Weng. “A note on Platt probabilistic outputs for support vector machines”. In: *Machine learning* 68.3 (2007), 267–276. DOI: 10.1.1.15.5292.

[21] C.C. Chang and C.J. Lin. “LIBSVM: a library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27. DOI: 10.1.1.20.9020.

[22] B. Schölkopf et al. “Estimating the support of a high-dimensional distribution”. In: *Neural computation* 13.7 (2001), 1443–1471. DOI: 10.1.1.39.912.

[23] B. Schölkopf et al. “New support vector algorithms”. In: *Neural computation* 12.5 (2000), 1207–1245. DOI: 10.1.1.2.6040.

[24] C.J.C. Burges and B. Schölkopf. “Improving the accuracy and speed of support vector machines”. In: *Advances in neural information processing systems* 9 (1997), 375–381. DOI: 10.1.1.54.1171.

[25] A. Aizerman, E.M. Braverman, and LI Rozoner. “Theoretical foundations of the potential function method in pattern recognition learning”. In: *Automation and remote control* 25 (1964), 821–837.

[26] W. Martinez. “Computational statistics handbook with MATLAB”. In: (2001).

[27] T. Glasmachers and C. Igel. “Maximum likelihood model selection for 1-norm soft margin SVMs with multiple parameters”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.8 (2010), 1522–1528. DOI: 10.1.1.163.5086.

[28] I. Guyon and A. Elisseeff. “An introduction to variable and feature selection”. In: *The Journal of Machine Learning Research* 3 (2003), 1157–1182. ISSN: 1532-4435. DOI: 10.1.1.3.8934.

[29] Y.W. Chang and C.J. Lin. "Feature ranking using linear SVM". In: *JMLR Workshop and Conference Proceedings: Causation and Prediction Challenge*. Vol. 3. Citeseer. 2008, 53–64.

[30] S. González et al. "SVM-Based Feature Extractor for L/H Transitions in JET". In: (2010). DOI: 10.1063/1.3502327.

[31] J. Weston et al. "Use of the zero norm with linear models and kernel methods". In: *The Journal of Machine Learning Research* 3 (2003), 1439–1461. DOI: 10.1.1.18.8133.

[32] H. Kim, J. Mullen, and J. Kepner. "Introduction to Parallel Programming and pMatlab v2. 0". In: (2009). DOI: 10.1.1.211.6279.

[33] E. Hourdakis, B.J. Simonds, and N.M. Zimmerman. "Submicron gap capacitor for measurement of breakdown voltage in air". In: *Review of scientific instruments* 77 (2006), p. 034702. DOI: 10.1063/1.2185149.

[34] FC Schuller. "Disruptions in tokamaks". In: *Plasma Physics and Controlled Fusion* 37 (1995), A135. DOI: 10.1088/0741-3335/37/11A/009.

[35] TC Hender et al. "MHD stability, operational limits and disruptions". In: *Nuclear fusion* 47 (2007), S128. DOI: 10.1088/0029-5515/39/12/303.

[36] PC de Vries et al. "Survey of disruption causes at JET". In: *Nuclear Fusion* 51 (2011). DOI: 10.1088/0029-5515/51/5/053018.

[37] V. Mukhovatov et al. "Overview of physics basis for ITER". In: *Plasma physics and controlled fusion* 45 (2003), A235. DOI: 10.1088/0741-3335/45/12A/016.

[38] G. Pautasso et al. "On-line prediction and mitigation of disruptions in ASDEX Upgrade". In: *Nuclear fusion* 42 (2002), p. 100. DOI: 10.1088/0029-5515/42/1/314.

[39] JV Hernandez et al. "Neural network prediction of some classes of tokamak disruptions". In: *Nuclear fusion* 36 (1996), p. 1009. DOI: 10.1088/0029-5515/36/8/I05.

[40] R. Yoshino. "Neural-net disruption predictor in JT-60U". In: *Nuclear fusion* 43 (2003), p. 1771. DOI: 10.1088/0029-5515/43/12/021.

[41] GA Rattá et al. "An advanced disruption predictor for JET tested in a simulated real-time environment". In: *Nuclear Fusion* 50 (2010). DOI: 10.1088/0029-5515/50/2/025005.

[42] WM Stacey. "A survey of thermal instabilities in tokamak plasmas: Theory, comparison with experiment, and predictions for future devices". In: *Fusion science and technology* 52.1 (2007), 29–67.

[43] M. Murakami, JD Callen, and LA Berry. "Some observations on maximum densities in tokamak experiments". In: *Nuclear Fusion* 16 (1976), p. 347. DOI: 10.1088/0029-5515/16/2/020.

[44] J. Hugill. "Transport in tokamaks–a review of experiment". In: *Nuclear Fusion* 23 (1983), p. 331. DOI: 10.1088/0029-5515/23/3/006.

[45] P.C. de Vries, MF Johnson, and I. Segui. "Statistical analysis of disruptions in JET". In: *Nuclear Fusion* 49 (2009). DOI: 10.1088/0029-5515/49/5/055011.

[46] CZ Cheng, HP Furth, and AH Boozer. "MHD stable regime of the Tokamak". In: *Plasma Physics and Controlled Fusion* 29 (1987), p. 351. DOI: 10.1088/0741-3335/29/3/006.

[47] D. Asimov. "The grand tour". In: *SIAM Journal of Scientific and Statistical Computing* 6.1 (1985), 128–143. DOI: 10.1137/0906011.

[48] GA Rattá, J. Vega, and A. Murari. "Improved Feature Selection Based on Genetic Algorithms for Real Time Disruption Prediction at JET". In: (2011).

[49] A. Murari et al. "Prototype of an adaptive disruption predictor for JET based on fuzzy logic and regression trees". In: *Nuclear Fusion* 48 (2008). DOI: 10.1088/0029-5515/48/3/035010.

[50] D. Mazon et al. "Equinox: A real-time equilibrium code and its validation at JET". In: *From physics to control through an emergent view* 15 (2010), pp. 327–333.

[51] D. Moreau et al. "Real-time control of the q-profile in JET for steady state advanced tokamak operation". In: *Nuclear fusion* 43 (2003), p. 870. DOI: 10.1088/0029-5515/43/9/311.

[52] PJ Carvalho et al. "Real-time plasma control based on the ISTTOK tomography diagnostic". In: *Review of Scientific Instruments* 79 (2008), 10F329. DOI: 10.1063/1.2917011.

[53] S. Tugarinov et al. "Conceptual design of the charge exchange recombination spectroscopy diagnostic for ITER". In: *Review of scientific instruments* 74 (2003), p. 2075. DOI: 10.1088/0741-3335/45/7/304.

[54] TA Casper et al. "ITER shape controller and transport simulations". In: *Fusion Engineering and Design* 83.2-3 (2008), pp. 552–556. DOI: 10.1016/j.fusengdes.2007.09.009.