Czech Technical University
in Prague

Faculty of Electrical Engineering

Department of Computer Science

Diploma thesis

Large-scale feature selection

Author: Bc. Jan Štercl

Supervisor: Ing. Danila Khikhlukha, Ph.D.

May 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štercl** Jméno: **Jan** Osobní číslo: **434986**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Kybernetická bezpečnost**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Velkoškálový výběr atributů

Název diplomové práce anglicky:

Large-scale feature selection

Pokyny pro vypracování:

Analyze current feature selection methods. Implement those methods, which are suitable for use on large datasets, according to the analysis. Compare implemented methods to standard benchmark datasets, measure the performance on large datasets and select the best feature selection approach.

Seznam doporučené literatury:

[1] SOMOL, Petr, Jiri GRIM a Pavel PUDIL. Fast dependency-aware feature selection in very-high-dimensional pattern recognition. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics [online]. IEEE, 2011, 2011, s. 502-509 [cit. 2018-10-05]. DOI: 10.1109/ICSMC.2011.6083733. ISBN 978-1-4577-0653-0. Available at: http://ieeexplore.ieee.org/document/6083733/
[2] SIKORSKI, Michael. a Andrew. HONIG. Practical malware analysis: the hands-on guide to dissecting malicious software. San Francisco: No Starch Press, c2012. ISBN 978-1-59327-290-6.
[3] HASTIE, Trevor, Robert TIBSHIRANI a J. H FRIEDMAN. The elements of statistical learning: data mining, inference, and prediction. 2nd ed. New York, NY: Springer, c2009. ISBN 978-0-387-84857-0.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Danila Khikhlukha, Ph.D., katedra počítačů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **29.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Ing. Danila Khikhlukha, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Prohlášení autora

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne . . . .

podpis autora

Acknowledgements

First of all, I would very much like to thank Ing. Danila Khikhlukha, Ph.D. for guiding me through the great field of feature selection, for his time and effort to push me forward.

Secondly, big thanks belong to my family and friends, for tolerating that I was very busy when writing this thesis.

Finally, special thanks need to be given to Cisco Systems, the company, which provided me with data and other resources to perform the experimental part of my thesis.

Title: Large-scale feature selection

Author: Jan Štercl

Department: Department of Computer Science

Supervisor: Ing. Danila Khikhlukha, Ph.D.

Abstract: Although the computational power is still growing, the amount of data, that needs to be processed in the field of machine learning quickly exceeds this capacity. The thesis focuses on feature selection aspect of a general classifier. There is a summary of important feature selection methods used since about seventy years ago. According to the research, seven methods were selected and implemented. Finally, it was shown, that all of the implemented methods helps to obtain better results in classification, which is true not only for public trial datasets, but also for real data from cybersecurity domain.

Index terms: machine learning, feature selection, big data, computer security, static analysis

Název práce: Velkoškálový výběr atributů

Autor: Jan Štercl

Katedra: Katedra počítačů

Vedoucí práce: Ing. Danila Khikhlukha, Ph.D.

Abstrakt: Možnosti výpočetních prostředků se neustále zvyšují, ovšem neúměrně k velikosti dat zpracovávaných v oblasti strojového učení, která narůstá ještě rychleji. Tato práce se zabývá výběrem atributů při klasifikaci dat. Teze shrnuje metody výběru atriubutů od padesátých let minulého století. Na základě rešerše bylo vybráno a implementováno sedm metod. Provedené experimenty ukazují, že implementované metody pomáhají k dosažení lepších výsledků nejen na veřejných zkušebních datových sadách, ale i na reálných datech z domény kybernetické bezpečnosti.

Klíčová slova: strojové učení, výběr atributů, velká data, počítačová bezpečnost, statická analýza

Contents

List of Figures

List of Tables

Abbreviations

AMP Advanced Malware Protection

ASFFS Adaptive Sequential Floating Forward Selection

AWS Amazon Web Services

CART Classification And Regression Tree

CTU Czech Technical University

DAF Dependency Aware Feature selection

FS Feature Selection

FSV Feature Selection concaVe

GA Genetic Algorithm

LASSO Least Absolute Shrinkage and Selection Operator

LGBM Light Gradient Boosting Machine

LSVM Linear Support Vector Machines

MI Mutual Information

ML Machine Learning

mRMR Max-Relevancy Min-Redundancy

NIPS Neural Information Processing Systems

OBD Optimal Brain Damage

PCA Principal Component Analysis

PE Portable Executable

PoC Proof of Concept

PSO Particle Swarm Optimization

RBA Relief-Based Algorighm

RF Random Forest

RFE Recursive Feature Elimination

S3 Simple Storage Service

SBS Sequential Backward Selection

SFFS Sequential Floating Forward Selection

SFS Sequential Forward Selection

SVM Support Vector Machine

Chapter 1

Introduction

> *„Data is the new oil. It's valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc. to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value.“*
>
> Clive Humby, 2006

Almost everyone interested in business or the industry in general probably once heard the quote above, at least the first sentence. Naturally there are people who try to detract it, saying it is just a marketing tool or proposing the minor differences, such as the tangibility, claiming that the same data can be collected by multiple companies and used more than one time, instead of one barrel of oil [1].

However, the arrival of „knowledge economy“ is undeniable. Looking into the global ranking of the top 10 biggest companies in 2008, one can find there is PetroChina, followed by Exxon, Russian Gazprom, Royal Dutch Shell and Sinopec (China). 10 years later, in 2018, guess how many oil companies are in this ranking? None. The top spots are occupied by technological companies and most of them are very deep in the data related business, for example Google, Microsoft, Amazon or Facebook [2].

Obviously, not just large corporations collect data. Everyone does. Currently, it is a necessity for every chain store, even the smallest ones, to have some kind of loyalty programs and collect data based on a customer IDs, what they buy, what they search online. This data might be used for targeted advertisements or sold later on to some third parties.

Going back to the quote from the beginning of this chapter, we would like to point out the main idea of Clive Humby's claim, that the oil has to be refined to be useful, in case of data, it has to be processed to obtain something of a value.

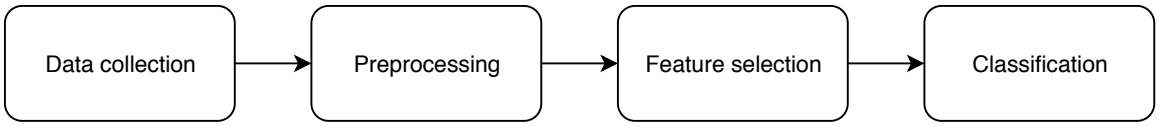


Figure 1.1: Common ML pipeline consists of data collection, extraction of numerical features in the preprocessing step, feature selection and classification.

Currently, the best known methods for dealing with data fall under the machine learning framework, whose name suggests, that the amount of data has far overreached the human comprehension. Some typical machine learning pipeline (see Figure 1.1) may have the following components. At first the data has to be collected, then the data is processed to introduce some numerical features, that represent the model of a problem. Those steps are followed by feature selection, the process intended to pick up the most expressive features and consequently reduce the dimensionality of the problem. The last step is the classification. Depending on the goal and circumstances classification may be substituted with regression or clustering.

The purpose of this thesis is to examine the wide field of one of these machine learning steps - feature selection - with the focus on large sets of data and problems connected with that. FS (feature selection) is an important stage in the whole procedure of data processing. It helps to reduce the complexity of datasets rich with features, select the most expressive features, reducing the dimensionality and make the classification process less computationally demanding. This is of a great importance as the supply of data is growing faster than computational resources, therefore there is a demand for more effective and faster algorithms.

To demonstrate the power of feature selection we would like to present here the story about the co-operation of Andrew Pole and the retailer Target from the beginning of this millennium [3].

The marketers of this chain selling daily consumption and household items once got a brilliant idea how to increase the enterprise's revenue. They spotted that future parents tend to buy goods all in one place without judging the price just to save energy and time. All they needed was to make the customers buy diapers at Target, because then the client will buy everything else there. They wanted to give the discount coupons for diapers to pregnant customers with the right timing before their baby was born (and before any competing retailer does that). The key was to determine the date of birth of the baby. That was Andrew Pole's time to shine. After running data through his algorithms, from thousands of products (features), he selected 25 items that should be tracked. Pole discovered some interesting patterns

such as buying a lot of unscented lotion somewhere in the fourth month of pregnancy, later followed by cotton balls, vitamin supplements and other products. Based on that he was able to estimate the birth date admirably precisely.

The presented research is aimed to study the effects of feature selection on classification performance. Introduction to the FS and overview of available methods is described in Chapter 2. The practical dimension starts with the description of the project owned by Cisco, in which these FS methods should be utilized, in Chapter 3. Chapter 4 further characterizes the methods selected to be implemented and explains why these were selected. The implementation details are covered in Chapter 5, followed by the experiments in Chapter 6. Then, in Chapter 7, there are the most important results highlighted and also, there is a proposal for future work. In the final part, Chapter 8, there is a conclusion of this thesis.

Chapter 2

State of the Art

Feature selection is a process of selecting a subset of features in order to improve a following predictor, reduce resources for storage and computation, reduce the training time, better understand the data or lower the effects of the curse of dimensionality[1]. The methods may even be tuned to give preferential treatment to one of these FS benefits. [4, 5].

Occasionally, just to filter out the features that are redundant can improve the generalization performance in the classification process, model interpretability and the training speed [6]. It is important to note here, that feature selection should not be mixed with methods like Principal Component Analysis (PCA) which is just reducing the number of dimensions by applying a liner transformation to a data space. Meanwhile, FS uses the data in their original form and does not transform them to another space [5].

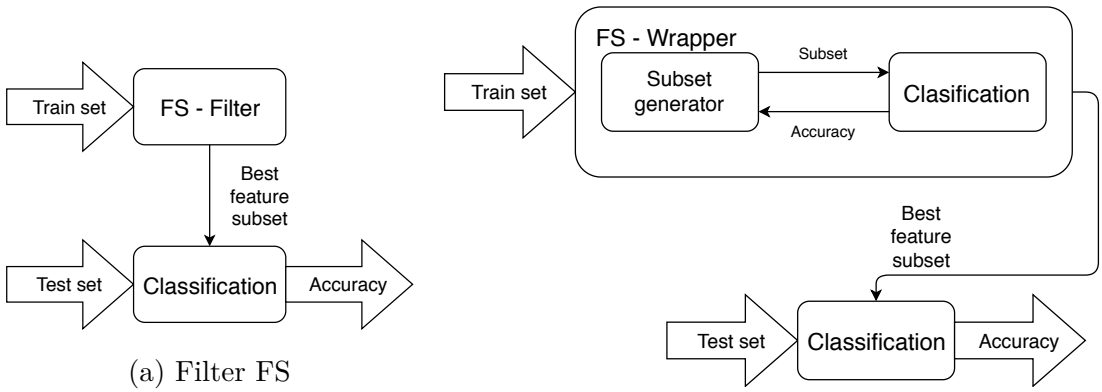
There is a straightforward way to find the best subset of features which is the exhaustive search. Unfortunately, with increasing n (number of features), it is computationally impossible to evaluate all subsets, as it is NP-hard because of the 2^n possible subsets [7]. That is a reason to adopt a heuristic approach for feature selection[2].

These heuristic algorithms are usually sorted to three classes (filters, wrappers and embedded methods) [4, 5]. Each of them we are going to review in the following sections. The chapter is going to be concluded with a comparison of the method classes from the theoretical point of view.

We take for granted the precomputed matrix of features $x_{i,j}$ with $i = 1, .., N$

[1] This expression was first used by Richard E. Bellman. When the dimension increases, the data becomes very sparse. That is a problem for traditional algorithms, starting on so simple things such as Euclidean distance. This phenomena is usually illustrated with a ratio of volumes of a hyper-sphere and a hyper-cube, which goes to zero as the dimension goes to infinity.

[2] Actually, in 1991, an algorithm called FOCUS was founded [8], which does the exhaustive search in quasi-polynomial time.



(a) Filter FS

(b) Wrapper FS

Figure 2.1: The difference between the filter FS (2.1a) and wrapper FS (2.1b) architecture is in the dependency on a learning algorithm (here mentioned as classifier). Accuracy is used as the performance evaluation criterion in this case. Inspired by [9].

samples in rows, $j = 1, .., D$ features in columns and a separate vector y of class labels, where the label c is usually an integer. This shall be the default notation for this thesis, unless locally stated otherwise.

2.1 Filter methods

Filter FS method's aim is to calculate a score for each feature, according to which the features are sorted. Then one of the two scenarios happen. Either a group of features is filtered out based on some predefined threshold, or a subset of k best features with the highest score is selected.

Typically, these scores are computed without employing the learning algorithm, which makes filter methods very undemanding for resources. The straightforwardness of filter methods is visualized in Figure 2.1a They are also statistically robust against overfitting [4, 10].

2.1.1 Correlation criteria

The Pearson correlation coefficient is an example of the simplest criteria [4, 11]. It is defined as follows:

$$R(j) = \frac{\text{cov}(x_j, y)}{\sqrt{\text{var}(x_j)\text{var}(y)}}, \tag{2.1}$$

where cov and var means covariance and variance respectively, x_j represents the j-th feature, y designates the response variable. We are able to approximate coefficients

(2.1) from data using the following estimator:

$$\hat{R}(j) = \frac{\sum_{i=1}^{N}(x_{i,j} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_{i,j} - \bar{x}_j)^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}}, \tag{2.2}$$

where the hat notation symbolize an estimation and the bar notation is a mean (in this case over index i).

Usually, $\hat{R}^2$ is used in practical applications, because it is easy to apply quadratic programming. When the task is formulated as a quadratic programming problem, the optimization is straightforward because of the convex objective function. However, using $\hat{R}^2$ over $\hat{R}$ brings a loss of information, since it is no longer distinguishable, if the value of $\hat{R}$ was initially positive of negative [4].

2.1.2 Mutual information

Another classifier independent metric is mutual information (MI). It is based on Shannon's information theory [12]. MI is defined using *entropy*:

$$\mathrm{H}(y) = -\sum_{c} \mathrm{P}(c) \log \mathrm{P}(c), \tag{2.3}$$

which gives the uncertainty in the class output. In combination with *conditional entropy*:

$$\mathrm{H}(y|x) = -\sum_{i} \sum_{c} p(i, c) \log p(c|i), \tag{2.4}$$

which expresses the uncertainty after knowing the i-th feature vector. Then, *mutual information* is defined as:

$$\mathrm{I}(y, x) = \mathrm{H}(y) - \mathrm{H}(y|x). \tag{2.5}$$

MI is a measure of dependence between two variables, in this case it is the dependency of the feature vector and the class labels. The higher the dependency is, the bigger the MI is. Absolutely independent x and y will result in zero MI.

In the end, the mutual information computed for each feature is taken as the score for a filter feature selection method.

2.1.3 Relief

This filter method was proposed by [13]. Relief benefits from randomized procedure, which is highlighted as very important part in [14]. In each round a random sample of data X is selected. Then, two samples from its neighborhood are found, one of the same class, called *Near-hit* (NH) by the authors and one of the other class (original

Relief was designed for two-class problems), called *Near-miss* (NM). These three feature vectors are then used to update weights of the features as follows:

$$W_j = W_j - \text{diff}(x_j, \text{NH})^2 + \text{diff}(x_j, \text{NM})^2, \tag{2.6}$$

where

$$\text{diff}(a, b) = (a - b)/\text{nu}, \tag{2.7}$$

nu being so called *normalization unit* to keep diff values between 0 and 1 included, when the feature is numerical. For categorical features:

$$\text{diff}(a, b) = \begin{cases} 0, & a \text{ and } b \text{ are the same,} \\ 1, & a \text{ and } b \text{ are different.} \end{cases} \tag{2.8}$$

Relief is efficient (polynomial time complexity, which is based just on number of features and iterations), but that can be said about all filter methods. According to the authors, advantages of Relief are its robustness against noise and that it is unaffected by feature interaction [13].

Over the time, Relief was evolved by many individuals and groups[3]. Problems of the original version were tackled. To mention few of many, [16] introduces a multi-class solution, [17] proposes a Relief mutation robust to outliers.

The original Relief has more drawbacks. One of them is a need of manual selection of the threshold to divide the selected subset of features from the rest of them. Secondly, Relief can select redundant features. And lastly, as the authors admit, it has problems with datasets that are sparse or not rich in samples.

2.2 Wrapper methods

Feature selection methods known as wrappers generate subsets of features and evaluate them using a classifier. That means, there are no scores for each individual feature.

The main difference between filters and wrappers is, that filters does not use a prediction algorithm in its process, meanwhile wrappers do and so they are classifier dependent, see Figure 2.1. On one hand, the presence of classifier in feature selection can lead to better results, on the other hand there is a high risk of over-fitting, also computational complexity might be a problem, especially with increasing dimension [14, 5].

Wrapper methods can be further divided to Sequential Selection Algorithms and Heuristic Search Algorithms.

[3] Today, it is even called a *family* of RBAs - Relief-based algorithms [15].

2.2.1 Sequential Selection Algorithms

Here, Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) can be distinguished. SFS starts with an empty subset. In each iteration, one individual feature which, together with previously selected features, has the best score in a learning algorithm, is added. When the required size of the subset is reached, SFS is stopped. SBS is the opposite. It starts with a full subset and the features are eliminated from the most dispensable. This simplistic greedy approach should not be so prone to overfit [18].

One can combine both of these methods into so called Sequential Floating Forward Selection (SFFS) [19]. After each iteration of SFS, there is one step of SBS applied on the currently selected subset, to conditionally exclude one feature. Then, according to a classifier, this feature is returned back in the set if the results are impaired by the exclusion, otherwise the feature is removed permanently. Similarly to SFS, SFFS is terminated when reaching the desired number of features.

The auhtors of [5] claim, that these methods produce nested subsets, meaning that they might allow some highly correlated features to be selected. In order to prevent this behavior, the authors of [20], suggest to conditionally exclude the feature and replace it with the second best, and so on. ASFFS (Adaptive SFFS) allows to add and remove adaptively computed number of features in one step of SFFS.

2.2.2 Heuristic Search Algorithm

Methods from this category usually deal with the problem of scanning the feature space effectively, which is not necessarily sequential.

Typical example of a heuristic search algorithm is Genetic Algorithm (GA) [21, 22], which is applied to binary vectors representing a bit mask over the feature vectors [5]. Starting from an initial generation of randomly sampled masks (chromosomes), in each round the best parents are chosen as the basis of the next generation, according to a performance predictor. In traditional GA there are two operations to compute the next generation with, firstly *crossover* (two parents swap parts of itself) and secondly *mutation* (bit - gene - is inverted; or multiple bits).

Development of GAs brought many conventional and unconventional improvements, which propose some additional methods of evolution. First one of them is *elitism* which lets the best individuals advance without any change. Second is a supervised form of crossover, so the children are significantly different from their parents. Crossover of very similar genes is prevented with a mechanism called *incest prevention*. Finally, *re-initialization* is forced, when the population is not evolving,

with next initial genes generated from the best individuals [23, 24].

Another frequently mentioned method is Particle Swarm Optimization (PSO) [25], which was originally developed as a model of animal behavior in a group, like a flock of birds. It works similarly as a genetic algorithm. PSO, compared to GA, does not employ any operations like mutation. It just updates the generated population based on the swarm knowledge and the individual knowledge of each agent (alternative of a GA's chromosome in PSO) [26, 27].

2.3 Embedded methods

The main characteristic of embedded methods is selecting features during the classification training process [28]. Embedded methods tries to balance the disadvantages of filter and wrapper approaches [5]. They are not as simple as filter methods and therefore not so fast, but also are nowhere near to wrappers in complexity.

Embedded methods usually optimize some objective function J. There are two types of methods. The first one usually combines greedy search and sequential selection to create nested subsets, because it is easy to compute J or at least to approximate. But, instead of using classifier, simpler metrics, like those used in filter methods are used. The second type directly optimizes an objective function by maximizing the goodness of fit (or minimizing the empirical error) while minimizing the number of used features [4].

2.3.1 Nested Subsets

Mutual information is good metric to use as the objective function. It is used for example by the authors of [11]. The difference between this approach and the one presented in 2.1.2 is that here a potential feature to add to a subset of selected ones is compared with MI not to the class, but to the subset of already sequentially selected features. An upgrade of this procedure which estimates the mutual information using Parzen windows is introduced in [29].

Mutual information is good metric to use as the objective function. It is used for example by the authors of [11], who minimize the MI between a potential feature to add and a subset of already sequentially selected features, meanwhile the approach presented in 2.1.2 uses the MI between the feature to add and the vector of class labels y.

Another method which uses MI as its scoring parameter is mRMR [30]. The relevancy of features is measured with MI between a feature and a class label. From

this, the relevancy part, computed basically the same way as the objective function of the method from the previous paragraph, is subtracted [5].

Before leaving mutual information in peace, it is necessary to mention that MI is the basis of feature selection embedded in tree algorithms, for example CART (Classification And Regression Tree) [31, 32].

In literature, one particular criterion appears often, just in slightly modified form. This feature score is used for two class problem as follows:

$$w_j = \frac{\mu_j(+) - \mu_j(-)}{\sigma_j(+) + \sigma_j(-)}, \tag{2.9}$$

where $(+)$ and $(-)$ denotes the two classes, j is the j-th feature, μ stands for mean and σ for standard deviation (σ^2 is variance then), was defined by [33]. In similar fashion, there is:

$$w_j = \frac{|\mu_j(+) - \mu_j(-)|}{\sigma_j(+) + \sigma_j(-)} \tag{2.10}$$

in [34],

$$w_j = \frac{(\mu_j(+) - \mu_j(-))^2}{\sigma_j^2(+) + \sigma_j^2(-)} \tag{2.11}$$

in [35] and

$$w_j = \frac{\sum_{k=1}^{C} n_k(\mu_{j,k} - \mu_j)^2}{\sum_{k=1}^{C} n_k \sigma_{j,k}^2} \tag{2.12}$$

which is called the Fisher's score, k stands for a class here (there are C classes), n_k means number of instances of the k-th class [36].

A representative of a method which approximates the objective function is for example Optimal Brain Damage (OBD), the pruning algorithm first used in neural networks [37] (from here the symptomatic name). This method uses second order Taylor expansion in the optimum of the cost function to approximate it.

2.3.2 Direct Objective Optimization

The second type of embedded methods is characteristic by direct optimization of the objective function. This category is well represented for example by Support Vector Machines (SVM) and similar methods. Those, which uses a linear predictor $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ are the most popular by far (and also in this thesis the linear methods would be covered, but non-linear would be not).

The theory of linear models is summarized in [32]. The author explains that this classification with embedded selection is an optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{w} \cdot \mathbf{x}_i + b, y_i) + C\Omega(\mathbf{w}), \tag{2.13}$$

where $\ell(f(x_i), y_i)$ is the value of loss function, in which $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, represents the empirical error, and $\Omega(\mathbf{w})$ is the penalty term, punishing the amount of used features. C is the coefficient controlling the influence of the penalty, $(\mathbf{x}_i, y_i)$ is a training point, $y_i \in \{-1, 1\}$.

Furthermore, [32] claims, majority of the methods that are trying to optimize the fit and penalize overfitting at the same time, are combinations of just a few loss functions and penalty (sparsity/density) terms. The usual empirical error terms are the ℓ_1 *loss*, also called *hinge loss*:

$$\ell_1(\mathbf{w} \cdot \mathbf{x} + b, y) := \max(1 - y(\mathbf{w} \cdot \mathbf{x} + b), 0), \tag{2.14}$$

the ℓ_2 *loss*:

$$\ell_2(\mathbf{w} \cdot \mathbf{x} + b, y) := (\mathbf{w} \cdot \mathbf{x} + b - y)^2 \tag{2.15}$$

and the *logistic loss*, which is used in logistic regression:

$$\ell_{logistic}(\mathbf{w} \cdot \mathbf{x} + b, y) := \log(1 + e^{-y(\mathbf{w} \cdot \mathbf{x} + b)}). \tag{2.16}$$

The two mentioned penalty terms are the ℓ_0 *norm*, which is the number of non-zero coordinates of $\mathbf{w}$, and the ℓ_1 *norm*:

$$\Omega(\mathbf{w}) = \sum_{j=1}^{D} |w_j|. \tag{2.17}$$

The original SVM [38] was just an algorithm maximizing the margin between data points. Back then in 1992 there were no controllable penalty term yet[4], that was added later.

By combining the *hinge loss* and ℓ_1 or ℓ_0 penalty a method known as ℓ_1-SVM and respectively FSV (Feature Selection concaVe) is obtained, both first proposed by [39]. In reality, the ℓ_0 penalty is approximated. Optimization with true ℓ_0 norm would not generalize well, because it would have many solutions and therefore the regularization would be insufficient as [40] explained. Also, using the real ℓ_0 penalty would result in an NP-Hard problem [7].

According to [28], the authors of [39] considered the ℓ_0 penalty better than the ℓ_1. The authors of [40] agrees and present their own re-scaling iterative SVM method, how to estimate ℓ_0 penalty.

Some research groups creates a mix of ℓ_0, ℓ_1 and ℓ_2 together, for example [41] use ℓ_0 as a penalization function and other two to support the goodness of fit. Other combination are examined by [28].

[4] The algorithm from [38] was not even called SVM, but the basis of the SVM algorithm is already distinguishable from here.

There are also authors, who think that the density penalty ℓ_1-SVM provide is good enough. [42] uses that simply without any iteration, claiming, that it will result in zero weight for enough features and after that, the selection might continue with backward elimination. This supports the idea of [43] who proposed similar algorithm a year before, under the name SVM-RFE (SVM-Recursive Feature Elimination).

To finalize the enumeration of loss-penalty combinations according to [32], LASSO method by [44] and Generalized LASSO derived of it by [45] have to be mentioned. The first one is a consolidation of ℓ_2 *loss* and ℓ_1 penalty, the second one uses *logistic loss* as its function to measure the empirical error.

2.4 State of the Art Conclusion

Before we start comparing the methods together, few more things have to be explained. Because of the project for which some FS methods should be implemented (see next chapter - 3) let us limit to supervised methods of feature selection, which keeps features in the original form (no feature extraction). Obviously, unsupervised methods could be taken into consideration and according to [4] or [5], also clustering or methods like PCA and similar can reduce the dimension, but that is not the aim of this thesis. Another simplification was, to present many of the methods as two-class problems, but for majority of them, there already are multi-class solutions, they are just not mentioned here.

At this point it should be evaluated which methods are worth considering to implement for a big-data project. In terms of the computational complexity, filters are clearly the winner with some embedded methods using filter-like metrics tightly behind. Wrappers, on the other hand should have advantage in selecting the best appropriate subset of features, but for the cost of speed, which is shown in Figure 2.2.

On average, embedded method seems to be good compromise, but also, it depends on individual use case. If someone needs to retrain the model daily, filter selection might be his best option. Otherwise, if a new model is needed every three months, spending a day instead of an hour with the training process is negligible and wrappers are the way to go.

The initial intention to decide which methods to implement based on the information gained form literature was left aside. Even though, the obtained knowledge is in favor of filter methods and maybe embedded ones, we decided to select multiple methods across all types.

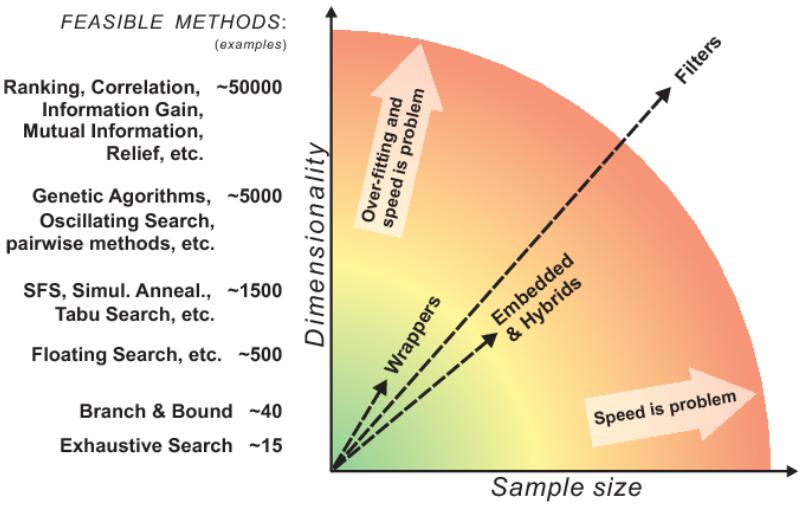


Figure 2.2: Approximated feasibility of the classes of methods. Although, the absolute numbers displayed are outdated, proportionally it corresponds to the reality. Taken over from [14].

Chapter 3

Cisco Advanced Malware Protection and Static Analysis

Information in this chapter are not so tightly connected to feature selection techniques, but explains the context of the practical application of this thesis.

This chapter is an intermezzo between a pure research of FS algorithms and a description of methods purposefully selected and implemented into the project introduced in this chapter.

In the first part, there is an introduction of the product, that is directly impacted by the results of this thesis. The second section briefly covers the pipeline of the particular project that applies machine learning (ML) to the static classification of executable files.

3.1 Cisco AMP

Historically, Cisco is recognized for its manufacturing of high-end network devices. However, due to the general trend in the IT business it has also focused on expanding of its software division. Its computer security branch has a wide portfolio of products and the AMP (Advanced Malware Protection) package is one of them [46]. It is a security tool that works both as an intrusion prevention system and as an intrusion detection system.

AMP is divided into multiple parts, for example AMP for Networks, AMP for Email Security, AMP for Web Security and AMP for Endpoints. The name of each system speaks for itself and additional information are on the product web page [46], but we definitely have to devote a paragraph at least to the last one mentioned, because the feature selection methods from this thesis should help to this particular product.

AMP for Endpoints focuses to find and mitigate the threats that are not detected by traditional anti-virus systems. To do this, it uses the information shared across all of the AMP products as well as its own cloud-based engines. One of these engines is the one we try to improve, and it is based on the Static Analysis project.

3.2 Static Analysis

Static analysis is a computer security discipline, which usually employs a malware analyst, who tries to uncover the behavior of an executable file, without actually running it, to decide, if a piece of software is malicious or benign. For this the analyst may employ tools, such as parsers, string matchers, disassemblers or even decompilers, which are able to extract a close approximation of the actual source code that the file was compiled from.

In spite of dynamic analysis, the binary file is not executed in this type of analysis. The analyst has to make a decision based on information available in PE (Portable Executable) file headers. These are data structures containing links for dynamic libraries, lists of imported and exported functions and so on [47].

The bottleneck in this procedure is, that the analyst is human. Even all analysts from all over the Earth, perfectly organized, could not analyze all the samples. Therefore the solution is to make a completely automated process of that, as a supervised learning task. Analysts are still used to provide labels for training data, but their work is significantly reduced.

There is nothing special about the pipeline of the Static Analysis project after all, because it is almost identical to the flow of any machine learning task, similar to the graphical interpretation in Figure 1.1.

The first step is to ingest as many data as possible. The examined object are obviously computer programs, more precisely the stored PE headers. This project has multiple sources. The data with labels are obtained for example the shared AMP database, Threat Grid cloud [48] and other internal Cisco databases.

The second phase is to normalize the data, because it comes in slightly different form from each of the sources. Also the labels has to be assigned correctly to known samples. Outputs of this step are training and testing (validation) datasets.

The third step is what this thesis is about. There are huge amounts of samples, the dimension of the data is also quite large. Classification results would be impaired if all dimensions were used, so the feature selection is applied. Moreover, there is a strict limit for number of used features set by the project documentation. This will be extended in Chapter 6, where the data is also described more precisely.

The last action done is the classification, which indicates, how good the feature selection was.

That is where the research part ends. In reality, data without labels from the endpoints, which needs to be classified are collected after the research FS and classification on the validation data. Since getting the full sample would be too burdensome for the endpoint, only a small percentage of features is downloaded to the AMP engine to predict the correct label.

Chapter 4

Selected feature selection methods

This chapter presents the FS methods implemented into Cisco AMP project.

As a way to handle a big data and fit the dataset in memory during computation, a map reduce procedure was created to handle data in batches. Individual methods use the map reduce slightly differently, some just loads the data in an efficient sparse matrix form, to do one big computation at the end, the other methods stores just a result of computation on this small chunk of data.

4.1 Current method

This method was already implemented in the Static Analysis project and it was the only method of feature selection. One of the goals of this thesis is to search for a better FS method. That makes an obvious reason why this baseline method should be at the first place in this list.

No large description is needed here. This method just adds up the data together in parallelized manner and then applies the scoring formula from the equation 2.10 on each feature to then rank them according to the score. This behavior makes the method easily classifiable as a filter method.

Also the properties of the method corresponds to filter methods in general. The advantage of this approach is its low computational time from which the vast majority is spent on loading the data. The disadvantage is that it does not inspect any relationships between features. It is just a straight, proof of concept (PoC) method.

4.2 Linear SVM method

The Linear SVM method (LSVM) makes use of the embedded feature selection in the SVM algorithm, in the same way, it was already presented in Chapter 2. A

LSVM classifier is provided with a training data, after the algorithms is fitted to the data, the coefficients from the primal SVM problem are taken as feature weights and features are sorted accordingly.

Relatively simple approach is selected. Following footsteps of [42], SVM with ℓ_2 loss and ℓ_1 penalty was used. On top of that, an iterative reweighing is done, inspired by [40].

Advantages of Linear SVM method should be its speed and effectiveness in high dimensions. Also, there are not many parameters to tune in SVM, just the constant C controlling influence of the penalty term. The C is also the biggest weakness of LSVM, because there is no deterministic procedure how to derive it. C has to be guessed from experience or multiple values for C have to be tried.

4.3 LightGBM method

Also in this method it was taken an advantage of an embedded FS process from a classifier. This time, it was a forest method developed in Microsoft called LightGBM (LGBM - Light Gradient Boosting Machine). The specialty of LGBM is that its trees are grown leaf-wise, not level-wise [49].

Similarly to the previous method, LGBM is executed on training data and during this process. Then a vector of so called feature importance is derived from the trained model. It is the sum of total information gains from nodes, in which the particular feature was used to split the samples [50].

Although there are multiple forest methods available, we were prone to select this one, because it is the method used as the classification engine in the Static Analysis project.

The positives of this method are basically the benefits shared across all embedded methods. It is relatively fast and so on. The only thing that could be viewed as a negative, apart of the need to select the parameters of the learning algorithm, is that trees in general are known to easily overfit the data [51]. But, in FS this is not so big issue, as it could be in the classification afterwards.

4.4 DAF methods

DAF methods are based on Dependency-Aware Feature selection algorithm proposed in [14].

The algorithm generates so called *probe* feature subsets, limited in size with an upper-bound (random number from 1 up to τ, which is an optional parameter, or D,

the number of features), which are then evaluated by a criterion function $J(\cdot)$ This is repeated until a desired number of probe subsets is generated, after a predefined time limit, or until there are such subsets generated, so that for every feature there are at least ω (another optional parameter) subsets that do contain the feature and at least ω that do not.

After the probe subsets with scores are prepared (let $\mathbb{S}$ denote the set of generated subsets, $\mathbb{S}_f$ and $\bar{\mathbb{S}}_f$ the set of probe subsets containing feature f and do not respectively), the summarizing part starts. From the generated subsets and their scores, for each feature there are means and variances calculated for cases where the feature is present and when not:

$$\mu_f = \frac{1}{|\mathbb{S}_f|} \sum_{S \in \mathbb{S}_f} J(S), \tag{4.1}$$

$$\bar{\mu}_f = \frac{1}{|\bar{\mathbb{S}}_f|} \sum_{S \in \bar{\mathbb{S}}_f} J(S), \tag{4.2}$$

$$\sigma_f^2 = \frac{1}{|\mathbb{S}_f|} \sum_{S \in \mathbb{S}_f} [J(S) - \mu_f]^2, \tag{4.3}$$

$$\bar{\sigma}_f^2 = \frac{1}{|\bar{\mathbb{S}}_f|} \sum_{S \in \bar{\mathbb{S}}_f} [J(S) - \bar{\mu}_f]^2. \tag{4.4}$$

Out of this, two dependency-aware scores are derived:

$$\mathrm{DAF}_0(f) = \mu_f - \bar{\mu}_f \tag{4.5}$$

$$\mathrm{DAF}_1(f) = \frac{(\mu_f - \bar{\mu}_f)|\mathbb{S}|}{|\mathbb{S}_f|\sigma_f + |\bar{\mathbb{S}}_f|\bar{\sigma}_f} \tag{4.6}$$

According to the authors of [14], DAF_0 may be seen as the average benefit of including the feature f. The normalized version, DAF_1, should prevent „possibly misleading emphasis put on features that appear important but behave unstably“.

Finally, as in every of the implemented methods, features are sorted by the DAF score and k best are selected.

Inspired by the article, the third score is proposed, which just replaces the means μ_f and $\bar{\mu}_f$ with medians m_f and $\bar{m}_f$, as a solution to the same issue, because of which DAF_0 was normalized to DAF_1, it is called DAF_2:

$$\mathrm{DAF}_2(f) = m_f - \bar{m}_f. \tag{4.7}$$

As a typical representative of wrapper FS algorithms, DAF methods heavily employs the learning algorithm, which takes majority of its running time (even though,

the generating phase and evaluating phase are embarrassingly parallelizable). Another disadvantage is the need to select a classifier and tune it. For the investigation in Cisco project a small RF (Random Forest) was selected as the guts of the $J(.)$ function, even though [14] originally uses k-Nearest Neighbor or SVM. The good thing is, that DAF evaluates the feature quality in context of other features. Also, this method can exclude correlated features in process, which is positive. DAF should be better than other context-aware FS methods, because of its speed and robustness against over-fitting [14].

There are some old context-aware methods, that are pretty much not feasible. To give an example, one comes from cooperative game theory and it is called *Shapley values* after Lloyd Shapley, who invented it in 1953 [52]. Shapley values are very similar to DAF, but does some reweighing with factorials and combination numbers, which makes the method useful only as a theoretical concept.

4.5 Parez method

Along with aforementioned methods yet another filter class algorithm was implemented. This one is custom made and it is based on information gain.

The data are split according to all features individually and the information gain is computed. That means that the features, according to which the data are divided in the most precise way, are selected.

A big plus for this method is, that is it as fast as the Current method, probably because Parez is a filter too. Another benefit is, that Parez does not have any demands for the data, like to have a normal distribution (which was true for the Current method, but it was silently ignored by the authors and not cared about later when the method was tested and worked). The first, general, disadvantage is the same as for a few other methods here, the relationships between features are omitted so, redundant features can go through. The second one is, that for the data split, a threshold has to be selected. We selected our threshold as the average of a minimal and a maximal value in a feature vector.

4.6 Meta method

The last method is basically a compilation of the other methods. Its name was developed from the word „metaphysics“, which in the original Greek version of the word - „metaphysika“ - meant „after physics“ [53]. In the same way, we have our „after method“ or „meta method“.

The method takes the ranks of features from many methods, in this case, it was fed with results of all previous seven methods (DAF methods are counted as 3, since all 3 types were used). Then a median of ranks is taken as the score of Meta method. Then the top k demanded features are selected based on this median of ranks.

The advantages of this approach are, that it should be robust and it should find an appropriate feature representation. It is very fast to do the computation of the Meta method itself. The only disadvantage is that it takes a lot of time to go through the other methods first. So, in cases, where the other methods would be used anyway, as in our case, it is a good idea to use this method afterwards. On the contrary, it makes no sense to do many FS procedures, just for the purpose of using Meta method or any similar collage concept.

Chapter 5

Implementation

In this chapter we would like to mention the tools used to implement our FS methods. In many cases the choice of a particular third party library or it's version was dictated by already available set of tools in Static Analysis project.

5.1 Python

The Static Analysis project introduced in Chapter 3 is implemented in Python, version 3.7. Python is an object-oriented, high-level programming language [54]. Python is de-facto a standard language, in the field of machine learning and artificial intelligence it is the number one language according to many user rankings, see for example [55, 56, 57].

Python is also an interpreted language, which could indicate lower performance than other standard languages, but a big advantage of Python is that there exist huge amounts of libraries, usually written in languages like C, which run very fast. Python works here like an universal glue to put the libraries together, benefiting from the very easy syntax.

Few of the libraries, that made our work much more convenient, will be mentioned next.

5.2 Python libraries

SciPy [58] is an organization, that provides many Python libraries as open source software. Majority of the important libraries that are used in the project were created by them. The development of these open-source tools are mostly sponsored by non-traditional charity organization NumFOCUS [59].

First library is NumPy. It is the basic package for fast computing with vectors and n-dimensional array objects in general.

The library Sparse does almost the same as NumPy, but it is dedicated to matrices with a sparse structure. The matrices can be stored in multiple formats, by row, by column, coordinate format and so on. The formats are quickly interchangeable and therefore can be tailored to the specific algorithm.

Scikit-learn is a huge quality of life improvement for ML experts from all over the World. The authors [60] made a collection of various machine learning algorithms, from Decision Tree to, for example, SVM, which is used in one of our methods (4.2). The whole library is based on Python, NumPy and other tools from SciPy project.

One last important library was LightGBM from Microsoft, this one was already mentioned as the basis of one of the methods proposed in this thesis (4.3) and as the classifier in the Static Analysis project.

5.3 AWS

Amazon Web Services (AWS) is a cloud service, that offers computing power and storage. AWS has many products for analytic purposes, machine learning, blockchain, internet of things, etc. [61]. We had the opportunity to use these resources thanks to Cisco.

From the variety of AWS tools we used Amazon Simple Storage Service (S3) for the storage purposes and AWS Batch from the category of computing resources to do experiments with big datasets.

5.4 Docker

To use AWS Batch conveniently, Docker Containers was used [62]. Container is a standardized unit of software which packs the code and its dependencies into one package - image, which is executable on any operating system as long it has some implementation of the Docker Engine installed.

Docker as a company currently occupies 83 percent of the market [63]. A year or two ago, when it was 99 percent and when the Static Analysis project for AMP was started, the choice of Docker as a container vendor was unambiguous.

5.5 Testing

Although this chapter so far was more or less about giving credits to the tools that was used in the Static Analysis project, this place is the best one for a few paragraphs about testing. The practical part of this thesis is not a basic school programming task, one finishes working on when it passes few test examples. This was written with intention to contribute to a project that would help to secure millions of devices globally. Therefore it has to be properly tested.

For the testing, we used a library called Pytest. It is a tool that allows user to test programs with the easy Python syntax and that is well implementable to continuous integration tools like Jenkins.

We had to create unit tests for a main FS class, and for the methods classes. LGBM and LSVM methods were focused less, because they use already mentioned external libraries. That is a great advantage. Firstly, open-source code is used by many users and when there is a bug, the community realizes that very fast. Secondly, Cisco has a strict policy about 3^{rd} party software and devotes a specialized testing department for this purposes.

Chapter 6

Experiments

In this chapter, there are results of the measurements done in order to research the qualities of the implemented FS methods.

In section 6.1 the datasets used in this research are described. There are definitions of the experiments in part 6.2, followed by the results in passage 6.3.

6.1 Datasets

Apparently, we need an object to test the methods on. The title of this thesis indicates, that some big data are to be involved in this chapter. Moreover, some smaller datasets were selected, with purpose to confirm a general functionality of the methods.

6.1.1 NIPS FS challenge datasets

As a workshop at the Neural Information Processing Systems (NIPS) in 2003, there was a FS challenge organized by Isabelle Guyon and her colleagues [64]. For this competition, a set of five datasets was provided which we are also going to use.

The set consists of five small datasets called Arcene, Dexter, Dorothea, Gisette and Madelon. They were chosen, because together they make a complete collection of datasets which are complementing each other in multiple characteristics, e.g. density or attribute type, and so on. Therefore it makes a great sense to use them all in our research as a good baseline.

In Table 6.1 there is further specification of the data, including the original domain from which the data are taken, the type of features, which might be binary, integer or real. In the density aspect, the datasets are distinguished as either dense or sparse. The authors of [64] originally provided each of the dataset in three parts (train set, validation set and test set) with intention to use the third one for the

Name	Domain	Type	Density	#Feat	#Trn	#Tst	%Pr
Arcene	Mass spectrometry	Real	Dense	10000	100	100	30
Dexter	Text classification	Integer	Sparse	20000	300	300	50
Dorothea	Drug discovery	Binary	Sparse	100000	800	350	50
Gisette	Digit recognition	Integer	Dense	5000	6000	1000	30
Madelon	Artificial	Real	Dense	500	2000	600	96
Static	Malware binaries	Integer	Sparse	400000	10^7	10^6	-

Table 6.1: Parameters of the NIPS FS challenge datasets and the Static dataset - name, domain, type of features, density, the count of features, the number of train samples and test samples, the percentage of probes used. For Static dataset, the number of probes is unknown. Inspired by [64].

evaluation of the competition, so there are known labels only for the first two. Since our task requires labels, only train set and validation set is used from these NIPS datasets, but from now the validation data and labels would be referenced as test data and test labels. Original test data would be omitted.

There are few more aspects about these FS Challenge datasets that makes them such a balanced portfolio and very appropriate to use for our task. All sets of data are provided with a specific percentage of features, by the authors of [64] called probes which are features made up to fit into the feature distribution but have no informational value. This probe percentage is displayed in the last column of Table 6.1. Moreover, Dorothea makes the challenge more difficult by having imbalanced classes. The Madelon dataset has its own specialty, that no single feature is informative on its own and Madelon was artificially created in this way.

These datasets are now available at the Irvine Machine Learning Repository of the University of California [65].

6.1.2 Static Analysis dataset

We also test the proposed methods with a real dataset used in Static Analysis project. Unfortunately this data set is property if Cisco and can not be publicly disclosed. Cisco has many of them and the datasets usually differ just in the dates, when the samples were collected, while features stay generally the same.

For this thesis we selected a dataset containing approximately ten millions samples in training part and a test set of about one million samples. Positive and negative class are almost in balance and the features are very sparse (about 99 percent of data are zeros).

We use PE headers as the source of features (see Chapter 3 for details). The individual data are integers, usually representing a binary feature with zeros and ones or specifies a count of how many times some property appeared, for example so called „strings“, which has a specific meaning in malware analysis theory, see [47]. These strings very much helps to boost the number of features in this dataset up to about 400000. Usually, in one sample there are just few of them involved and many features indicating a count of the particular „string“ stays zero, from here the great amount of features and the sparsity of the dataset. Extra features which a human analyst would not probably use are for example sizes of the program sections.

6.2 Experimental design

This experiment measures the performances of the individual FS methods. At first, FS is done on the first part of data, training data. Then, using just the selected features of the training data a LGBM classifier[1] is trained. Finally, test data are fed into the trained classifier. The accuracy of this classification on test data was selected as the criterion for comparison. Accuracy is a standard metric used by many authors ([9, 14, 27] are the examples) for this kind of experiments. All of the methods are compared on each dataset. The number of involved features is used only as a secondary metric.

The feature selection process was done for each method on each particular dataset. It was measured, how the accuracy changes when just the top k features selected. The k was increased with a growing step up to the point where all features were used with one exception, which is the Static dataset.

This limitation was already mentioned in section 3.2. Due to the hardware limitation only a maximum of 10000 features can be obtained from the endpoint to not overload it and its resources. For the research purposes we went a bit further, up to 15000 features, but not the full 400000.

Even though the comparison with all features is not possible on Static dataset, we found at least one way, how to compare quality of the FS methods. There would be an experiment with ten thousand (the upper bound for selected features in Static Analysis project) randomly selected features.

[1] That is the one used in Static Analysis project.

Method	Accuracy	#Feat
Current	0.68	2
LSVM	0.83	900
LGBM	0.68	4
DAF_0	0.82	91
DAF_1	0.81	9
DAF_2	0.84	35
Parez	0.82	350
Meta	0.71	250

Table 6.2: The best accuracy reached by individual methods in Arcene dataset and number of features used for that.

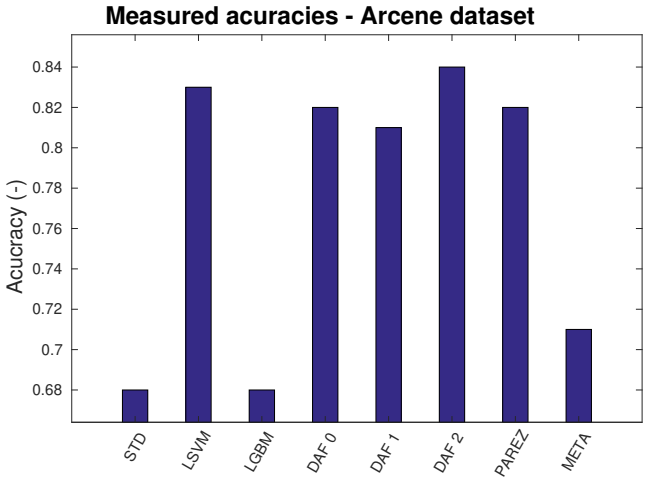


Figure 6.1: The best accuracy reached by individual methods in Arcene dataset.

6.3 Results

This section is going to be divided in two parts. In the first one, the general results on each dataset will be announced. In the second, the key facts and results will be highlighted.

6.3.1 General results

We are aware of the fact, that some of the graphs proposed in this section seems to be unclear, but they are necessary for the general comparison of the methods. With thorough reading, the graphs have a value. At the end of this section, there is a summary of the best methods for each dataset.

Arcene

The aggregate graph for the smallest of the datasets is in Figure 6.7. Table 6.2 and Figure 6.1 show the best results of individual methods and number of features needed to achieve it. The best accuracy is obtained with DAF_2 method: 84 percent, while using just 35 out of 10000 features. Also DAF_1 worked well, with only 9 features used it gained 81 percent accuracy.

Also, DAF_0, LSVM and Parez scored satisfyingly, but with much more used features than the best methods. Otherwise, Current and LGBM, in spite of having maximal accuracy with the least features, had very poor performance, which heavily influenced the Meta method, too.

Method	Accuracy	#Feat
Current	$0.85\bar{6}$	25
LSVM	$0.89\bar{3}$	36
LGBM	0.88	15
DAF_0	$0.89\bar{3}$	32
DAF_1	0.88	120
DAF_2	$0.88\bar{6}$	40
Parez	0.83	7000
Meta	$0.88\bar{3}$	10

Table 6.3: The best accuracy reached by individual methods in Dexter dataset and number of features used for that.

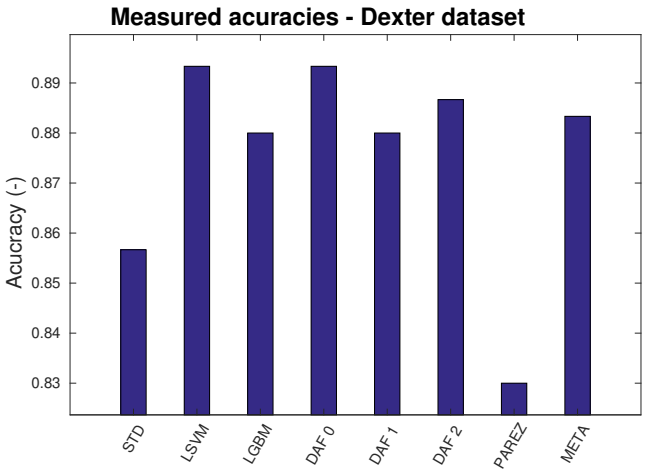


Figure 6.2: The best accuracy reached by individual methods in Dexter dataset.

Dexter

In Figure 6.2 and Table 6.3, there is a summary of the measurements on Dexter dataset. The best methods here are DAF_0 and LSVM, with accuracy of $0.89\bar{3}$ and 32, respectively 36 used features. Figure 6.8 captures the whole experiment.

The disappointment here is the Parez method, which just ties the accuracy that is reached without feature selection. The good thing is, that it reduces the number of features from 20000 to 7000, but 7000 is a huge number compared to the performance of other methods. Otherwise, Meta method was a pleasant surprise here. With only 10 features used, its accuracy was relatively close to the highest one reached on this dataset.

Other methods scored relatively well, with an improvement in accuracy in comparison to classification without FS. Current method is the worst from this group, but we do not consider it a failure, because of the adequate number of selected features.

Dorothea

Dorothea dataset was dominated by DAF methods. Surprisingly, all DAF types reached the same accuracy, but with different amount of features used. DAF_1 was the best of them, meanwhile DAF_0 was the worst of the three. Even though DAF_0 used just 5 percent of features it is too much compared to other methods.

In this series of experiments with Dorothea dataset, no clear loser is going to be nominated. Usage of all methods resulted into significant growth in accuracy and

Method	Accuracy	#Feat
Current	0.94286	37
LSVM	0.94571	67
LGBM	0.94286	220
DAF_0	0.95429	5000
DAF_1	0.95429	200
DAF_2	0.95429	490
Parez	0.94571	460
Meta	0.94857	430

Table 6.4: The best accuracy reached by individual methods in Dorothea dataset and number of features used for that.

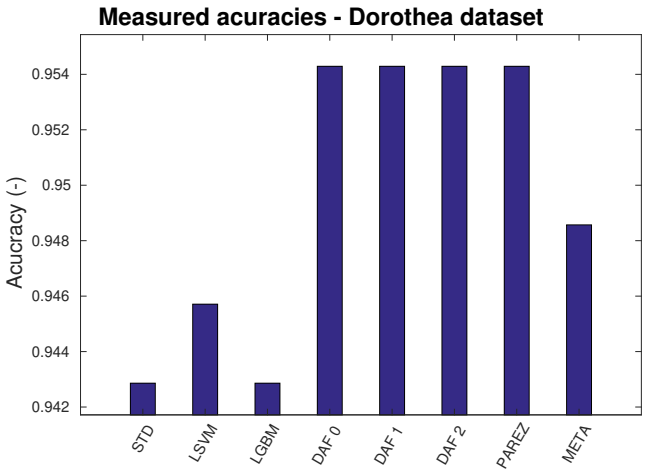


Figure 6.3: The best accuracy reached by individual methods in Dorothea dataset.

all methods (excluding one, already mentioned exception) less than 0.5 percent of features.

The cumulative graph for Dorothea dataset is in Figure 6.9 and maximal values reached are in Table 6.4 and Figure 6.3. One might be afraid, that the results were influenced by the fact, that the dataset is not balanced. However, this issue was addressed by setting proper class weights in classification and also in feature selection, where it was possible.

Gisette

The measurements on Gisette dataset came out in similar manner as for Dorothea dataset. The DAF methods helped to gain the highest accuracy score of 96 percent, but this time DAF_1 left the most features for classification.

Also, all of the methods were even more close together, both in accuracy and count of used features, than in the Dorothea experiments.

The details of the experiment with Gisette dataset can be found in Figure 6.10, Table 6.5 and Figure 6.4 displays the maximum values.

Madelon

With Madelon dataset, things starts to get more interesting again. Madelon, as mentioned, is an artificial dataset, with just 4 % features (20), that are useful. The rest of the features is there just to confuse the observer.

Method	Accuracy	#Feat
Current	0.957	340
LSVM	0.959	500
LGBM	0.958	130
DAF_0	0.96	340
DAF_1	0.96	1000
DAF_2	0.96	400
Parez	0.956	280
Meta	0.959	600

Table 6.5: The best accuracy reached by individual methods in Gisette dataset and number of features used for that.

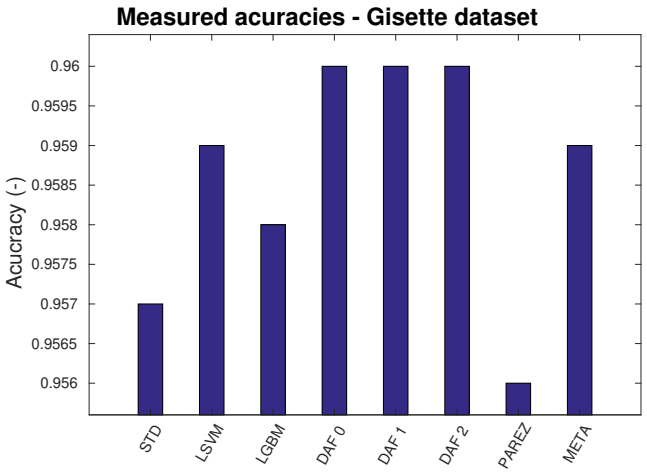


Figure 6.4: The best accuracy reached by individual methods in Gisette dataset.

Here, the clear winner is LGBM method, which can be seen in Figure 6.5 or Table 6.6. Also the least amount of features selected by LGBM, with just 2 extra unnecessary features. The graph of all measurements on Madelon dataset is in Figure 6.5.

The worse method can be also clearly marked, even though the difference from the rest of the methods is not as significant as it was in case of the best one. It was the Current method scoring the lowest accuracy and selecting the most features, too. The rest of the methods was anomalous neither positively, nor negatively.

Static

For Static dataset, there is a difference, that there is no experiment with all features done, because of memory limits. For all other datasets it could be said, that classification after FS was better then without FS.

From Figure 6.6 and Table 6.7 it can be seen, that accuracy gained with five out of eight methods have growing tendency after the mentioned border of 10000 features. Unluckily, the limitation is present in the Static Analysis project and has to be considered. In Table 6.8, the accuracy gained by the methods using exactly 10000 features is displayed for additional comparison. No method got its highest score with less than 10000 features used. The overall graph can be seen in Figure 6.12.

The method with the highest accuracy is Meta, which won using maximum amount of features, that was measured with. Although all of the methods resulted

Method	Accuracy	#Feat
Current	0.835	400
LSVM	$0.851\bar{6}$	370
LGBM	$0.87\bar{3}$	22
DAF_0	$0.84\bar{6}$	70
DAF_1	$0.851\bar{6}$	45
DAF_2	$0.85\bar{3}$	86
Parez	0.845	160
Meta	$0.848\bar{3}$	33

Table 6.6: The best accuracy reached by individual methods in Madelon dataset and number of features used for that.

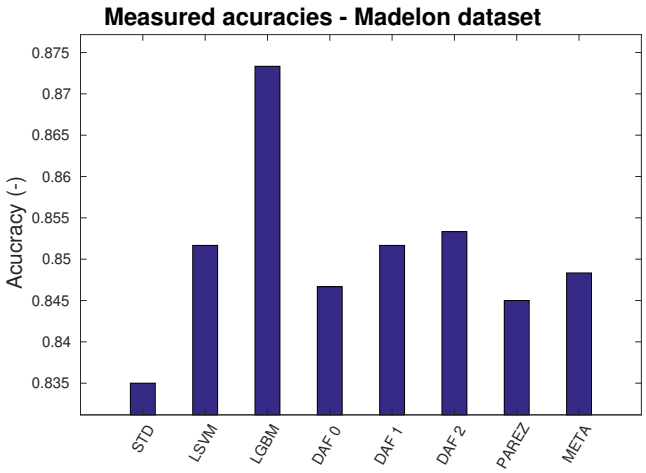


Figure 6.5: The best accuracy reached by individual methods in Madelon dataset.

in a very similar score in the end, much slower growth of DAF curves, cannot be overlooked. Interestingly enough, DAF methods did not negatively affect the Meta method. On the 10000 border it is the LSVM method, which is the best. The bonus result is that even the worst method scores almost 3 percent higher in accuracy, than if the features were selected by bare luck[2].

General results summary

Table 6.9 recapitalizes the winning and losing methods on each dataset. DAF methods might seem to be very superior, but in terms of accuracy, the first place was shared with another method 3 times (only once it was shared with non-DAF method), the one which selected less features was preferred as the winner in that case.

In the same table, the worst methods on each dataset are mentioned. In our case, the worst method is the one with the lowest maximum value of accuracy. Also, there is the difference between the maxima of the best and the worst method.

[2] Compared on 10000 features.

Method	Accuracy	#Feat
Current	0.99045	10000
LSVM	0.99124	10000
LGBM	0.99125	12500
DAF_0	0.98951	15000
DAF_1	0.98964	15000
DAF_2	0.98872	15000
Parez	0.99031	15000
Meta	0.99149	15000

Table 6.7: The best accuracy reached by individual methods in Static dataset and number of features used for that.

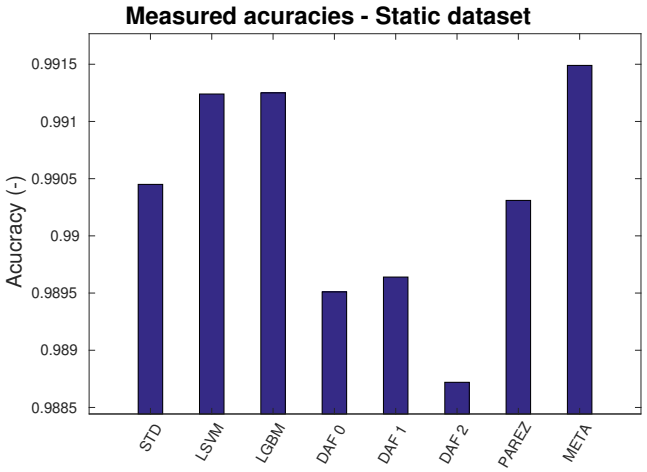


Figure 6.6: The best accuracy reached by individual methods in Static dataset.

Method	Accuracy
Current	0.99045
LSVM	0.99124
LGBM	0.99119
DAF_0	0.98813
DAF_1	0.98851
DAF_2	0.98728
Parez	0.98995
Meta	0.99108

Table 6.8: The accuracy reached by individual methods in Static dataset while using 10000 features.

Dataset	Arcene	Dexter	Dorothea	Gisette	Madelon	Static
Best method	DAF_2	DAF_0	DAF_1	DAF_0	LGBM	Meta
Worst method	LGBM	Parez	LGBM	Parez	Current	DAF_2
Difference	0.16	$0.06\bar{3}$	0.0114	0.004	$0.038\bar{3}$	0.004

Table 6.9: The methods which helped to gain maximal accuracy for each dataset.

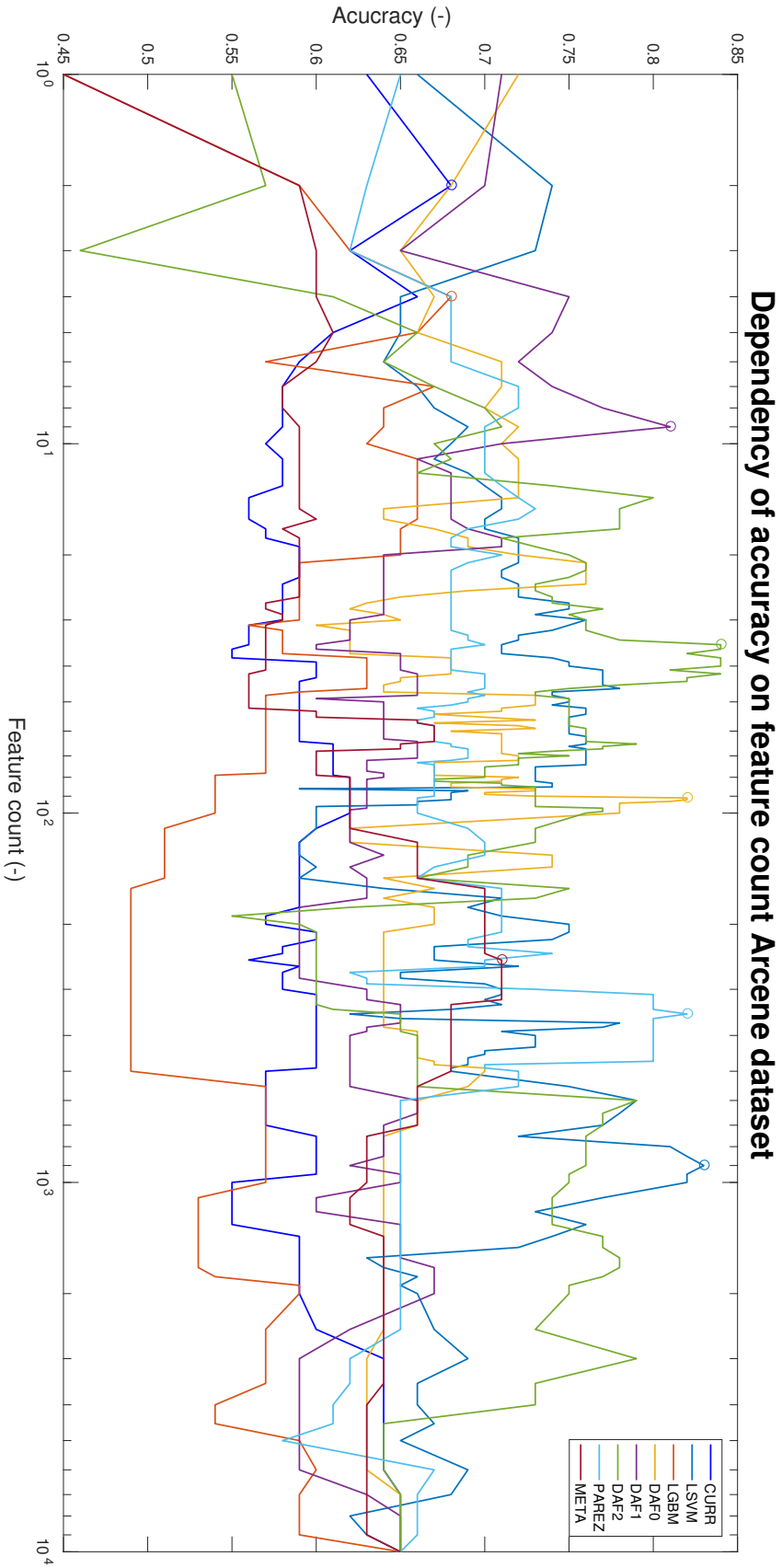


Figure 6.7: Dependency of accuracy on increasing number of features, measured on Arcene dataset. The best result is denoted for each curve with a circle.

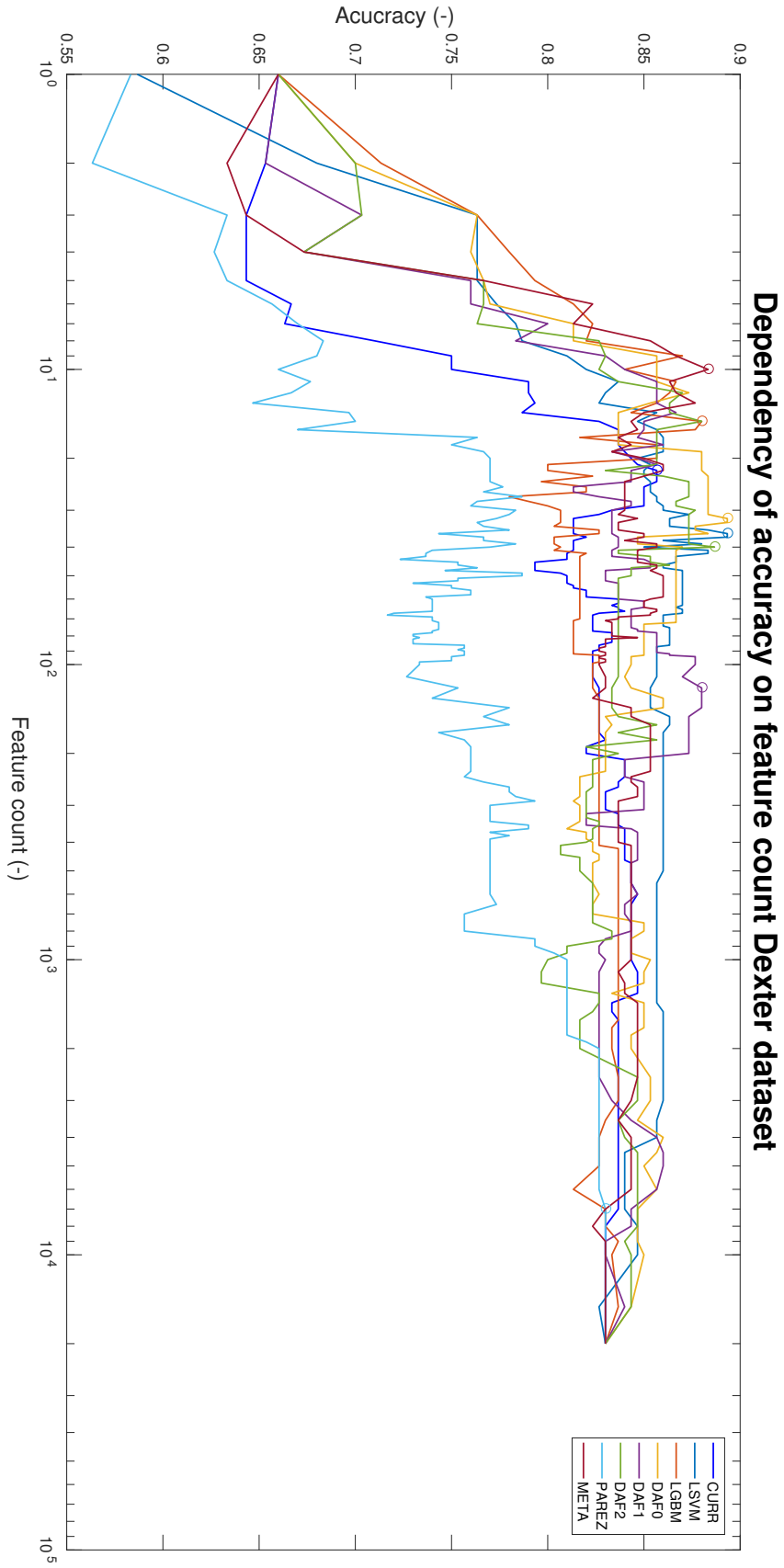


Figure 6.8: Dependency of accuracy on increasing number of features, measured on Dexter dataset. The best result is denoted for each curve with a circle.

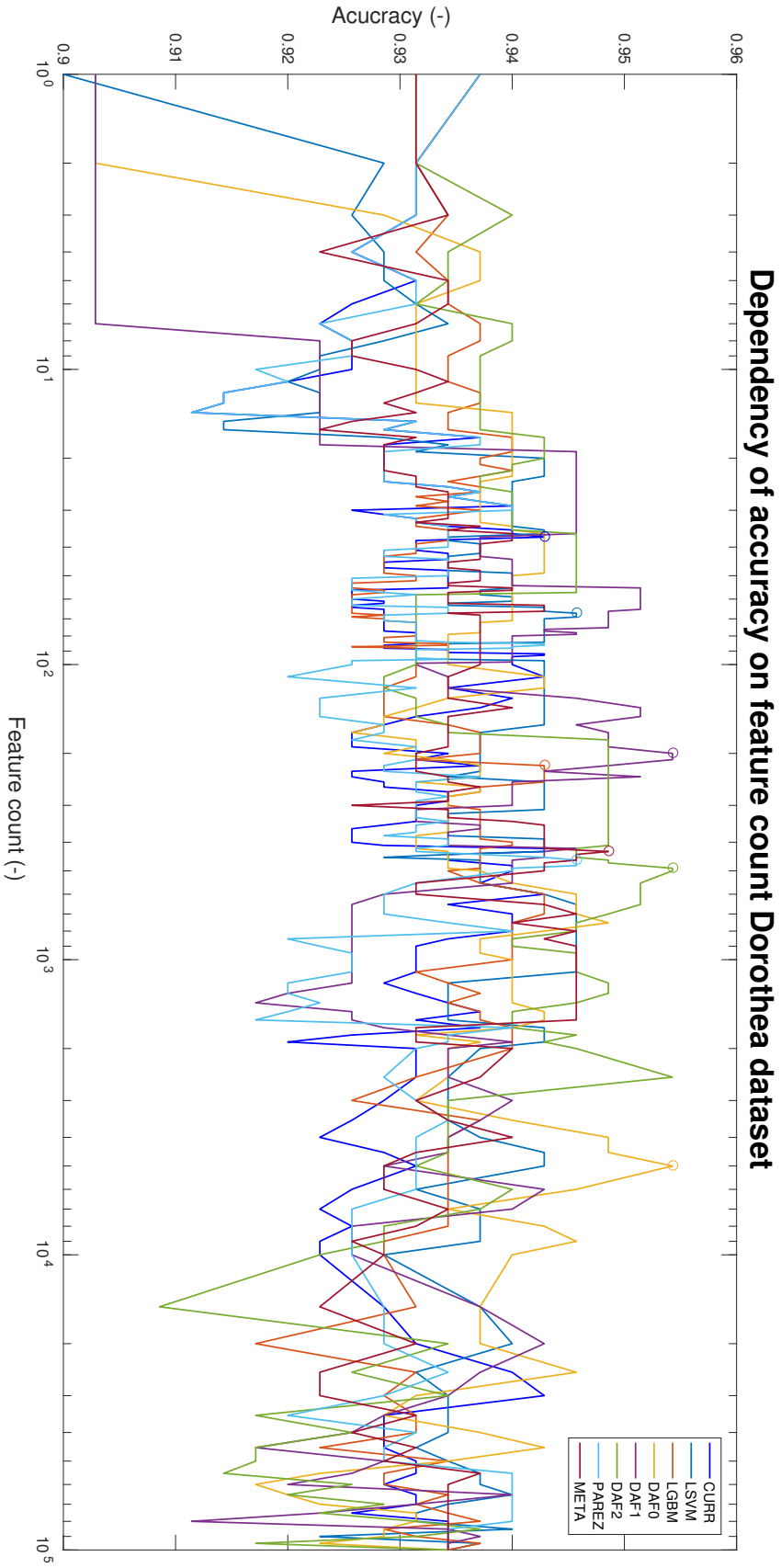


Figure 6.9: Dependency of accuracy on increasing number of features, measured on Dorothea dataset. The best result is denoted for each curve with a circle.

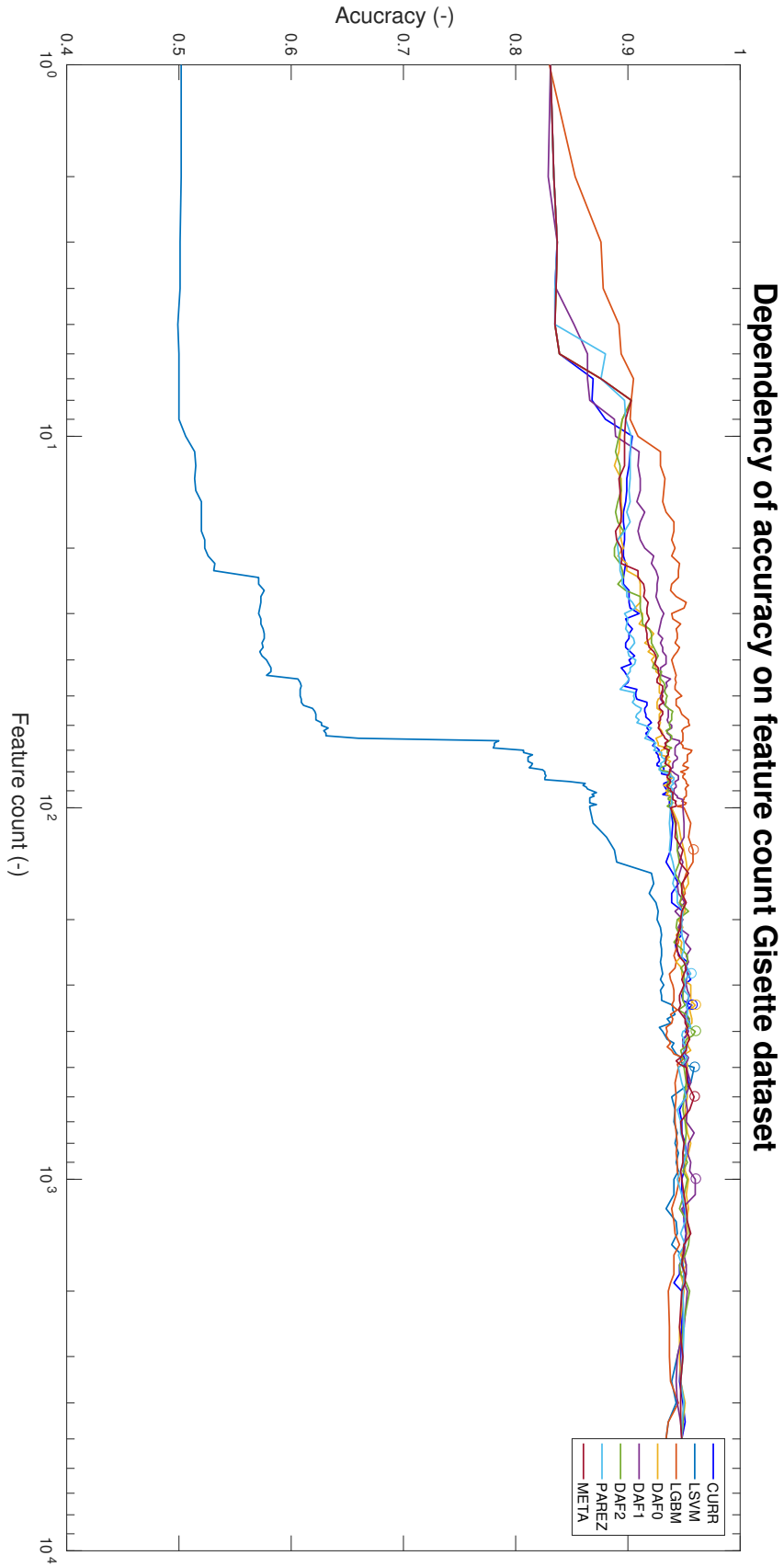


Figure 6.10: Dependency of accuracy on increasing number of features, measured on Gisette dataset. The best result is denoted for each curve with a circle.

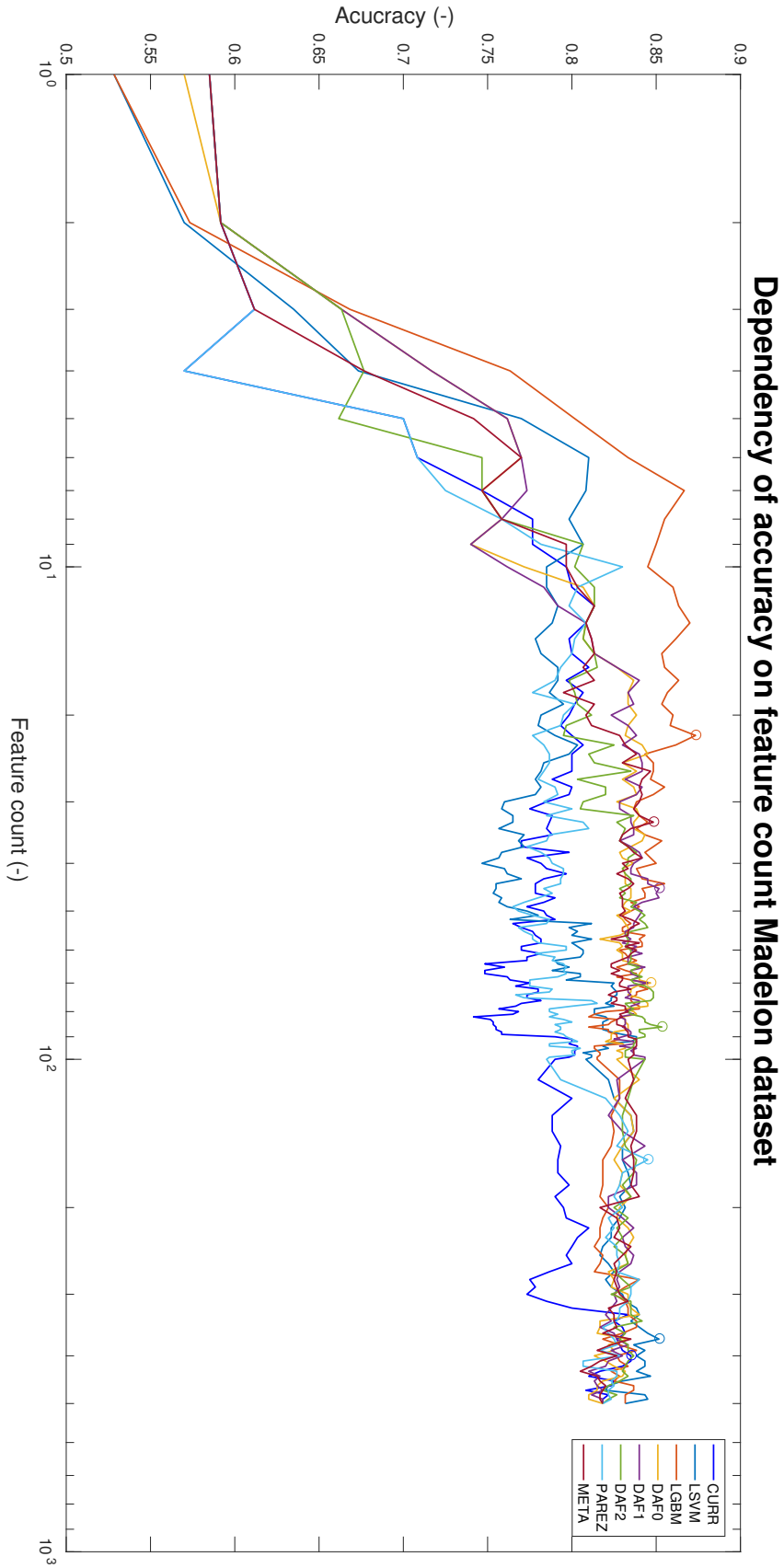


Figure 6.11: Dependency of accuracy on increasing number of features, measured on Madelon dataset. The best result is denoted for each curve with a circle.

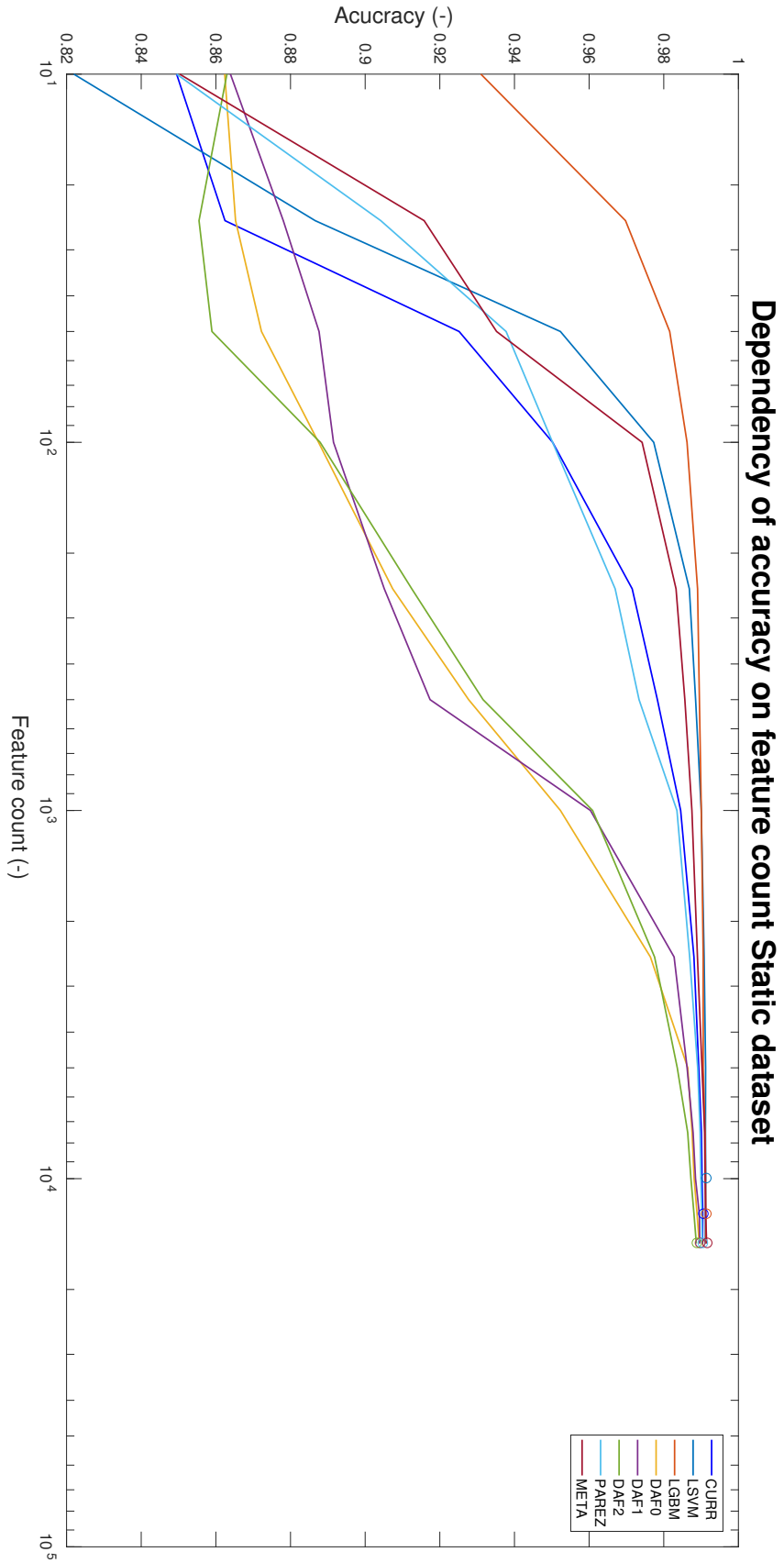


Figure 6.12: Dependency of accuracy on increasing number of features, measured on Static dataset. The best result is denoted for each curve with a circle.

Chapter 7

Discussion

In the first part of this chapter, we are going to highlight the results that are most important for us. The second proposes some ideas for future work.

7.1 Result's highlights

This part will present a compilation of the best and the most interesting results and facts from the experimental part.

It shows unexpected problems that occurred in the results and highlights, which anticipations came true and which did not.

7.1.1 The ultimate goal

The assignment of this thesis brought an extra task, which was not mentioned, but was the most expected. The actual goal, why Cisco assigned the task, was not to compare the FS methods, but to beat the Current one.

The thesis could end up in two ways. Either it would propose a FS procedure, that would beat the Current method at least on Static dataset, or there would be a bunch of theoretical concepts and proofs showing, why Current method is the best one for this kind of data.

On this place, we can happily announce, that the thesis evolved the first way and we proposed feature selection methods, which beat the Current one not only on Static dataset (see Figure 7.1), but on all trial datasets, too.

7.1.2 Usefulness of feature selection

With exception of Static dataset, where it cannot be confirmed due to insufficient working memory, FS improves the resulting accuracy in a comparison to classifi-

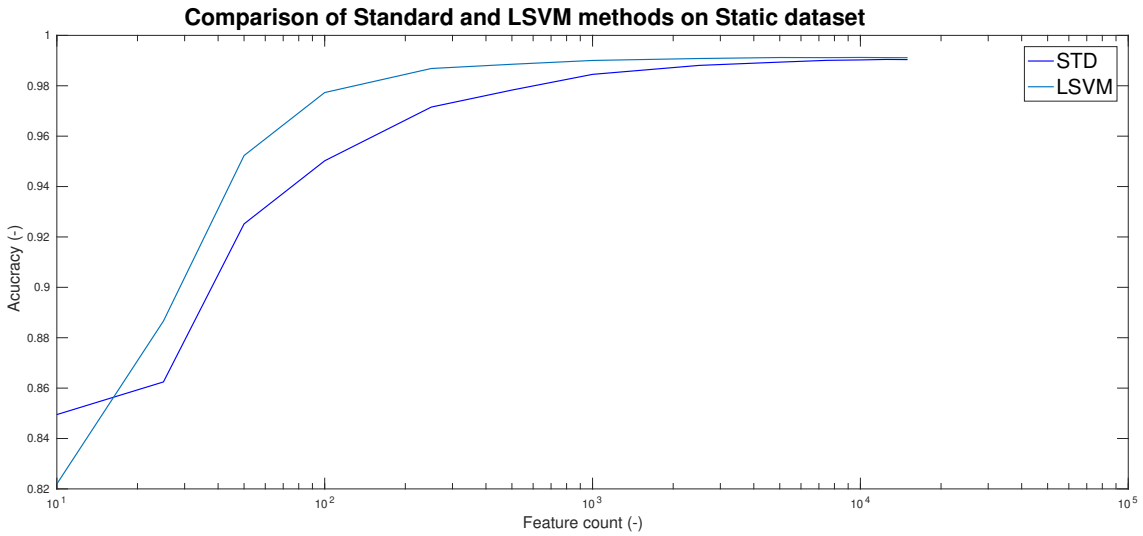


Figure 7.1: Comparison of Current and LSVM method on Static dataset. LSVM was selected for this, because it provided the best result, when considering the 10000 upper bound for used features.

cation with all features. The importance of this fact for the thesis is, that the developed methods generalize and no method damages the process.

7.1.3 Literature's favorites

At the end of conclusion, the authors of [32], presents an assumption, that with an increasing number of training samples, embedded methods will be better than filter methods.

This seems to be correct expectation, at least based on our experiments. On Arcene dataset, which has just 100 training data points, LGBM shares the worst accuracy and LSVM does good in this, but for the cost of selecting incomparably many features than winner does. With more and more samples, these embedded methods became better and better, up to a point, that they are the best two methods on Static dataset with ten millions samples.

7.1.4 Some information gains are more equal than others

Parez and LGBM methods are both based on information gain. Parez was implemented in the most primitive approach, computing just information gain in level one trees, meanwhile training a forest of trees and counting in all splitting nodes is very sophisticated algorithm.

We expected that considering accuracy, Parez would be beaten in all cases by LGBM. We were wrong. As already mentioned, on Arcene dataset LGBM was the

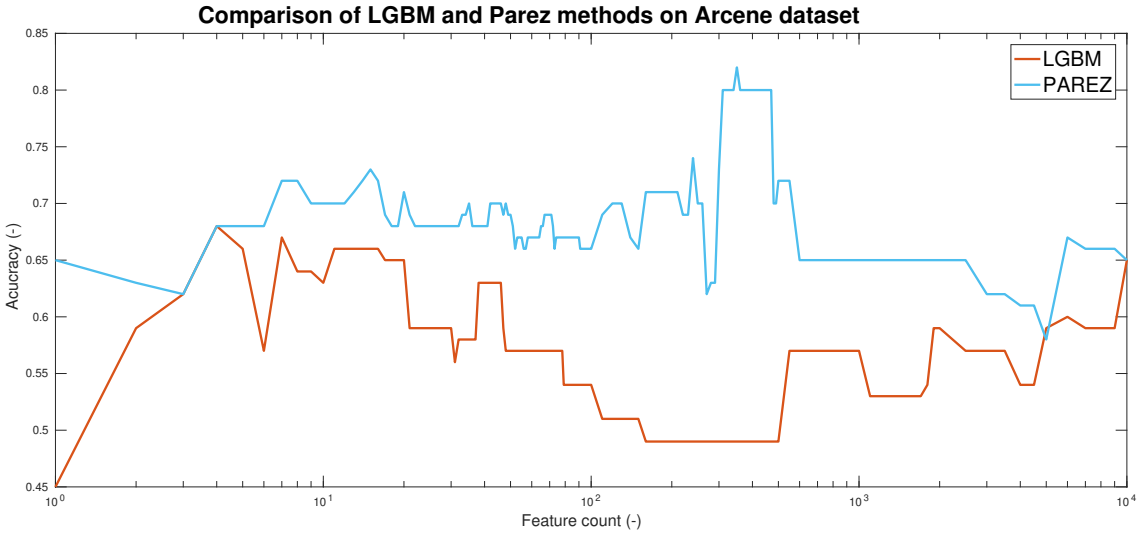


Figure 7.2: Comparison of Parez and LGBM method on Arcene dataset. With exception of one point where the Parez's curve has its minimum, LGBM scores worse than Parez.

worst one. On the other hand, as can be seen from Figure 7.2, Parez is not better than LGBM not only in maximal accuracy, but consistently across almost every feature count measured.

This might be caused by the fact, that Parez is well suited to classify Arcene data, or, the amount of samples is just too low, so the forest structure cannot learn from it properly.

7.1.5 DAF disappointment

DAF methods in its performance did exactly the opposite from the embedded methods. Meanwhile LSVM and LGBM worked poorly on smaller datasets and were the top methods on the big dataset, DAF produced good results at first but then it was lagged behind the other methods.

The reason for that could be the limitation on probe subset count[1]. To defend our choice of this parameter, we can say that a Mann-Whitney U-test was performed, according to which even ten times less probe subsets would be enough for DAF to work just alright. Another option is that there are just better methods better suitable for the type of data that are in the Static dataset. Also, there might be problem with correct labeling of samples, which could impair DAF more than other methods.

Although there is a word „disappointment“ in the headline for this part, we

[1] We selected the probe subset limit as the stopping criterion of the three options, more information about stopping criteria available in section 4.4.

would say, that DAF worked pretty well. We just had higher expectations and there just were better methods, easier to tune.

Speaking of tuning the methods, we performed no exhaustive search for the parameter selection. We could call it just a „very sparse grid search“. The parameters of the methods were selected more or less by experience. Of course the tuning could be done in exhaustive manner, but that would be costly, time consuming and unnecessary for this thesis.

7.2 Future work and development possibilities

In this part there are three topics, which are obvious next steps in FS part of the Static Analysis project. Some of them were already in progress, but are too practical or way out of the assignment, to cover them in this thesis.

7.2.1 Multi-class feature selection

Many of the methods mentioned in State of the Art chapter (2) and methods implemented this thesis (chapter 4), have already prepared theoretical or even practical solution for multi-class problems.

We decided to stick to two-class problems in this thesis, because of inability to find such a great set of datasets testing FS quality for multi-class problems, as the datasets from NIPS FS challenge, which are just two-class tasks. Also it made the prototyping of the methods much easier.

This would be useful to further dig in. We could imagine adding a new label, „suspicious“ to Static dataset. Samples marked like this could be then passed to a malware analyst to decide between its maliciousness or legitimacy.

7.2.2 Deeper data analysis

When we will be freed from the boundaries of the assignment, we would like to make a proper data analysis. For the thesis, it was sufficient to know, that the data are sparse integers, and a little bit about the domain.

There is a suspicion, that a lot of binary features are uncorrelated. If this was the case, Standard FS method, the already implemented one, would be the best possible method.

7.2.3 Method chaining

What did not happened much in this thesis was experimentation with combinations of the methods. Actually there was one.

One of them was Meta method, which was here strictly implemented as a combination of all previous methods. In section 4.6 it was said, that it is not worth the time to run all methods, just to use Meta method after. But, with current knowledge, we could imagine the Meta method based just on LGBM and LSVM as a very potent combination for Static Analysis data.

We see Meta method as a conjunction of a parallel combination of other methods. What we are still missing is a serial combination of methods, for example to first run ℓ_1-SVM and then pass just features with non-zero weights to the next step, maybe a wrapper like DAF, which would be benefited with already lowered dimension.

Unfortunately, even though it would be very interesting to try all of these and other combinations, it is more of an engineering task, based on experience instead of research, therefore it was also excluded from the thesis.

Chapter 8

Conclusion

The fact, that the assignment did not come form a university environment, but from a company, that operates on the market, has its benefits and disadvantages. It needs to be noticed, that the thesis was partially influenced by that.

Although such case might have it's pros and cons, the bottom line is that the positives heavily outweigh the negatives. The biggest advantage is availability of resources, both the data, which were that large, that it would be very hard to obtain elsewhere, and the computation resources, which the university does not have so far[1]. Another perk is also personal. Since the implemented piece is used in a real product, I see that as the first contribution of mine to the cyber security industry, importance of which grows rapidly.

The cost for all of this is, that the thesis was censored by Cisco with intention, not to leak the valuable pieces, like the code or the data. But to relieve the pressure form this point, there is a guarantee, that the work was very well checked and it was taken care of the quality.

Despite this one stain on the transparency, the assignment might be considered fulfilled in all points. The research of FS methods was thorough. The key elements of the field were observed from as far back as the half of the previous century.

Based on the research, favorable methods were selected. In total, seven new FS methods were created for Static Analysis project. The size of the contribution can be expressed by comparison of Figure 1.1 as the „before“ state and Figure 8.1 as the „after“ state.

The methods had to undergo a series of experiments on public datasets, created especially to test feature selection quality and then on the large data from the Static Analysis project.

[1] Actually from very recent, there is a computing cluster on CTU, which would be sufficient [66], but it came too late and there was not any chance, even theoretical to use it.

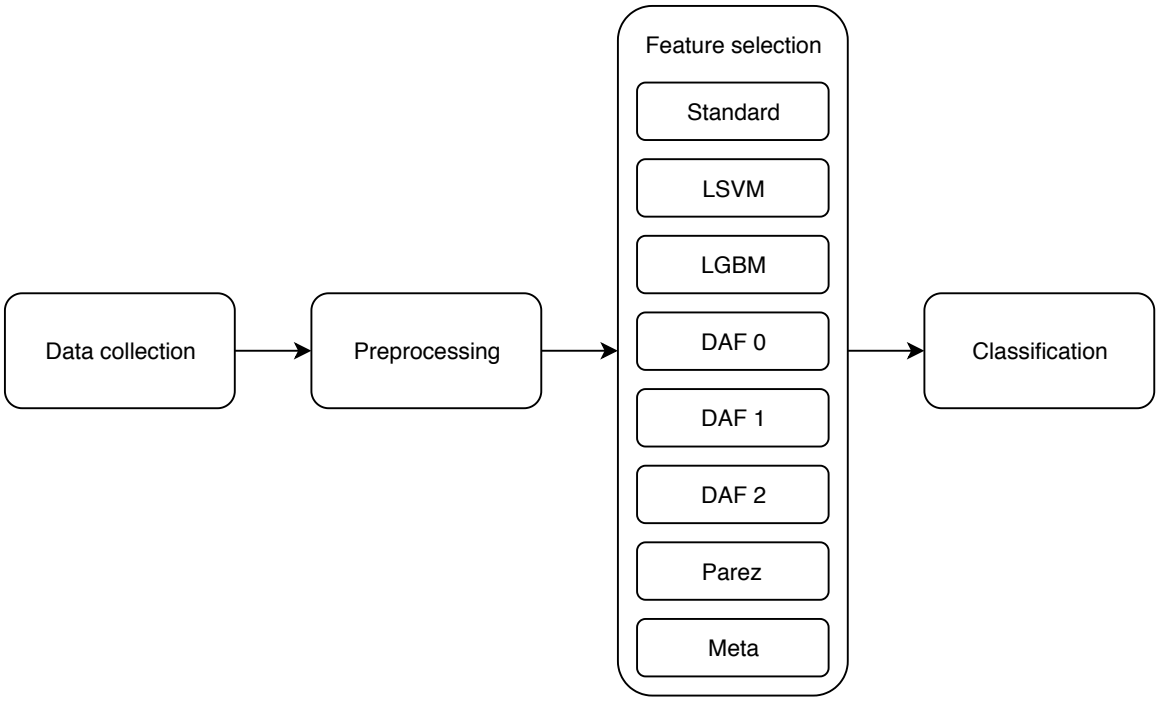


Figure 8.1: Static Analysis project pipeline after adding the products of this thesis.

The task, to „select the best feature selection approach“ could be a bit obscure, but basically it was meant to find a FS method, which would improve the results of the classifier in the AMP for Endpoints tool and also in this last point, the goal was achieved.

Bibliography

[1] New, Joshua. 2018. “Why Do People Still Think Data Is The New Oil?”. Online. In Center For Data Innovation. https://www.datainnovation.org/2018/01/why-do-people-still-think-data-is-the-new-oil/.

[2] Johnston, Stephen. 2019. “Largest Companies 2008 Vs. 2018, A Lot Has Changed”. Online. In Millford Asset Management Limited. https://milfordasset.com/insights/largest-companies-2008-vs-2018-lot-changed.

[3] Duhigg, Charles. 2014. The Power Of Habit: Why We Do What We Do In Life And Business. New York: Random House Trade Paperbacks.

[4] Guyon, Isabelle, and André Elisseeff. 2003. “An Introduction Of Variable And Feature Selection”. Journal of Machine Learning Research: Special Issue On Variable And Feature Selection 3: 1157 - 1182.

[5] Chandrashekar, Girish, and Ferat Sahin. 2014. “A Survey On Feature Selection Methods”. Online. Computers & Electrical Engineering 40 (1): 16-28. doi:10.1016/j.compeleceng.2013.11.024.

[6] Koehrsen, Will. 2018. “A Feature Selection Tool For Machine Learning In Python”. Online. Towards Data Science. https://towardsdatascience.com/a-feature-selection-tool-for-machine-learning-in-python-b64dd23710f0.

[7] Amaldi, Edoardo, and Viggo Kann. 1998. “On The Approximability Of Minimizing Nonzero Variables Or Unsatisfied Relations In Linear Systems”. Online. Theoretical Computer Science 209 (1-2): 237-260. doi:10.1016/S0304-3975(97)00115-1.

[8] Almuallim, Hussein, and Thomas. G. Dietterich. 1991. “Learning With Many Irrelevant Features”. In Proceedings of the Ninth National Conference on Artificial Intelligence, 547-552. Anaheim, California: AAAI Press.

[9] Kohavi, Ron, and George H. John. 1997. “Wrappers For Feature Subset Selection”. Online. Artificial Intelligence 97 (1-2): 273-324. doi:10.1016/S0004-3702(97)00043-X.

[10] Hastie, Trevor, Robert Tibshirani, and J. H Friedman. c2009. The Elements Of Statistical Learning: Data Mining, Inference, And Prediction. 2nd ed. New York, NY: Springer.

[11] Battiti, Roberto. 1994. “Using Mutual Information For Selecting Features In Supervised Neural Net Learning”. Online. IEEE Transactions On Neural Networks 5 (4): 537-550. doi:10.1109/72.298224.

[12] Shannon, Claude Elwood, and Warren Weaver. 1949. The Mathematical Theory Of Communication. 1st ed.. Urbana: University of Illinois Press.

[13] Kira, Kenji, and Larry A. Rendell. 1992. “The Feature Selection Problem: Traditional Methods And A New Algorithm”. In Proceedings of the Tenth National Conference on Artificial Intelligence, 129-124. San Jose, California: AAAI Press.

[14] Somol, Petr, Jiri Grim, and Pavel Pudil. 2011. “Fast Dependency-Aware Feature Selection In Very-High-Dimensional Pattern Recognition”. Online. In 2011 IEEE International Conference On Systems, Man, And Cybernetics, 502-509. IEEE. doi:10.1109/ICSMC.2011.6083733.

[15] Urbanowicz, Ryan J., Melissa Meeker, William La Cava, and Jason H. Moore. 2017. “Relief-Based Feature Selection: Introduction And Review”. Online. In Computer Research Repository. http://arxiv.org/abs/1711.08421.

[16] Kononenko, Igor. 1994. “Estimating Attributes: Analysis And Extensions Of Relief”. Online. In Ecml-94: Proceedings Of European Conference Of Machine Learning, 171-182. Lecture Notes In Computer Science. Secaucus, New Jersey: Springer-Verlag New York. doi:10.1007/3-540-57868-4_57.

[17] Sun, Yijun. 2007. “Iterative Relief For Feature Weighting: Algorithms, Theories, And Applications”. Online. IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6): 1035-1051. doi:10.1109/TPAMI.2007.1093.

[18] Reunanen, Juha. 2003. “Overfitting In Making Comparisons Between Variable Selection Methods”. Journal of Machine Learning Research 3: 1371-1382.

[19] Pudil, Pavel, Jana Novovičová, and Josef Kittler. 1994. “Floating Search Methods In Feature Selection”. Online. Pattern Recognition Letters 15 (11): 1119-1125. doi:10.1016/0167-8655(94)90127-9.

[20] Nakariyakul, Songyot, and David P. Casasent. 2009. “An Improvement On Floating Search Algorithms For Feature Subset Selection”. Online. Pattern Recognition 42 (9): 1932-1940. doi:10.1016/j.patcog.2008.11.018.

[21] Goldberg, David E. 1989. Genetic Algorithms In Search, Optimization, And Machine Learning. 1st ed.. Reading, Mass.: Addison-Wesley Pub. Co.

[22] Hussein, F., N. Kharma, and R. Ward. 2001. “Genetic Algorithms For Feature Selection And Weighting, A Review And Study”. Online. In Proceedings Of Sixth International Conference On Document Analysis And Recognition, 1240-1244. IEEE Comput. Soc. doi:10.1109/ICDAR.2001.953980.

[23] Eshelman, Larry J. 1991. “The CHC Adaptive Search Algorithm: How To Have Safe Search When Engaging In Nontraditional Genetic Recombination”. Online. In Foundations Of Genetic Algorithms, 265-283. Elsevier. doi:10.1016/B978-0-08-050684-5.50020-3.

[24] “JCLEC Evolutionary Algorithms”. 2019. Online. SourceForge. Slashdot Media. http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=4&Itemid=5.

[25] Kennedy, James, and Russell Eberhart. 1995. “Particle Swarm Optimization”. Online. In Proceedings of ICNN’95 - International Conference On Neural Networks, 1942-1948. IEEE. doi:10.1109/ICNN.1995.488968.

[26] Tu, Chung-Jui, Li-Yeh Chuang, Jun-Yang Chang, and Cheng-Hong Yang. 2007. “Feature Selection Using PSO-SVM”. Online. IAENG International Journal Of Computer Science 33 (1).

[27] Alba, Enrique, Jose Garcia-Nieto, Laetitia Jourdan, and El-Ghazali Talbi. 2007. “Gene Selection In Cancer Classification Using PSO/SVM And GA/SVM Hybrid Algorithms”. Online. In 2007 IEEE Congress On Evolutionary Computation, 284-290. IEEE. doi:10.1109/CEC.2007.4424483.

[28] Neumann, Julia, Christoph Schnörr, and Gabriele Steidl. 2005. “Combined SVM-Based Feature Selection And Classification”. Online. Machine Learning 61 (1-3): 129-150. doi:10.1007/s10994-005-1505-9.

[29] Kwak, N., and Chong-Ho Choi. 2002. “Input Feature Selection For Classification Problems”. Online. IEEE Transactions On Neural Networks 13 (1). Piscataway, New Jersey: IEEE Press: 143-159. doi:10.1109/72.977291.

[30] Peng, Hanchuan, Fuhui Long, and Chris Ding. 2005. “Feature Selection Based On Mutual Information Criteria Of Max-Dependency, Max-Relevance, And Min-Redundancy”. Online. IEEE Transactions On Pattern Analysis And Machine Intelligence 27 (8): 1226-1238. doi:10.1109/TPAMI.2005.159.

[31] Breiman, Leo, Jerome Friedman, Charles J. Stone, and R. A. Ohlsen. 1984. Classification And Regression Trees. Wadsworth and Brooks.

[32] Lal, Thomas Navin, Olivier Chapelle, Jason Weston, and André Elisseeff. 2006. “Embedded Methods”. Online. In Feature Extraction, 137-165. Studies In Fuzziness And Soft Computing. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-35488-8_6.

[33] Golub, T. R., C. Huard, D. K. Slonim, P. Tamayo, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield. 1991. “Molecular Classification Of Cancer: Class Discovery And Class Prediction By Gene Expression Monitoring”. Science 286: 531-537.

[34] Furey, T. S., N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. 2000. “Support Vector Machine Classification And Validation Of Cancer Tissue Samples Using Microarray Expression Data”. Online. Bioinformatics 16 (10): 906-914. doi:10.1093/bioinformatics/16.10.906.

[35] Pavlidis, Paul, Jason Weston, Jinsong Cai, and William Noble Grundy. 2001. “Gene Functional Classification From Heterogeneous Data”. Online. In Proceedings Of The Fifth Annual International Conference On Computational Biology - Recomb ’01, 249-255. New York, New York, USA: ACM Press. doi:10.1145/369133.369228.

[36] Aggarwal, Charu C. 2014. Data Classification: Algorithms And Applications. Data Mining And Knowledge Discovery Series. Boca Raton, FL: CRC press.

[37] LeCun, Yann, John S. Denker, and Sara A. Solla. 1990. “Optimal Brain Damage”. In Advances In Neural Information Processing Systems 2, 598-605. San Francisco, California: Morgan Kaufmann Publishers.

[38] Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. “A Training Algorithm For Optimal Margin Classifiers”. Online. In Proceedings

Of The Fifth Annual Workshop On Computational Learning Theory - Colt '92, 144-152. New York, New York, USA: ACM Press. doi:10.1145/130385.130401.

[39] Bradley, Paul S., and O. L. Mangasarian. 1998. "Feature Selection Via Concave Minimization And Support Vector Machines". In Proceedings Of The Fifteenth International Conference On Machine Learning, 82-90. San Francisco, CA, USA: Morgan Kaufmann Publishers.

[40] Weston, Jason, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. 2003. "Use Of The Zero Norm With Linear Models And Kernel Methods". Journal Of Machine Learning Research 3 (3): 1439-1461.

[41] Perkins, Simon, Kevin Lacker, and James Theiler. 2003. "Grafting: Fast, Incremental Feature Selection By Gradient Descent In Function Space". Journal Of Machine Learning Research 3 (3): 1333-1356.

[42] Bi, Jimbo, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. 2003. "Dimensionality Reduction Via Sparse Support Vector Machines". Journal Of Machine Learning Research 3 (3): 1229-1243.

[43] Guyon, Isabelle, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. "Gene Selection For Cancer Classification Using Support Vector Machines". Online. Machine Learning 46 (1/3): 389-422. doi:10.1023/A:1012487302797.

[44] Tibshirani, Robert. 1996. "Regression Shrinkage Selection Via The Lasso". Journal Of The Royal Statistical Society Series B (Methodological) 58 (1): 267-288.

[45] Roth, V. 2004. "The Generalized Lasso". Online. Ieee Transactions On Neural Networks 15 (1): 16-28. doi:10.1109/TNN.2003.809398.

[46] "Advanced Malware Protection". Online. Malware Protection - Cisco. https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/index.html.

[47] Sikorski, Michael., and Andrew. Honig. 2012. Practical Malware Analysis: The Hands-On Guide To Dissecting Malicious Software. San Francisco: No Starch Press.

[48] "Cisco Threat Grid". Online. Threat Grid - Advanced Malware Protection - Cisco. https://www.cisco.com/c/en/us/products/security/threat-grid/index.html

[49] Mandot, Pushkar. 2017. “What Is LightGBM, How To Implement It? How To Fine Tune The Parameters?”. Online. In Medium. https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc.

[50] “LightGBM Documentation”. 2019. Online. In Read The Docs: Create, Host, And Browse Documentation. https://lightgbm.readthedocs.io/en/latest/index.html.

[51] Tang, Cheng, Damien Garreau, and Ulrike von Luxburg. 2018. “When Do Random Forests Fail?”. In 32Nd Conference On Neural Information Processing Systems. Montréal, Canada.

[52] Shapley, Lloyd S. 1953. “A Value For N-Person Games”. Annals Of Mathematics Studies 28: 307-317.

[53] “Metaphysics”. 2019. Online. The Basics Of Philosophy. https://www.philosophybasics.com/branch_metaphysics.html.

[54] “Python”. 2019. Online. Python Software Foundation. https://www.python.org/.

[55] Heath, Nick. 2019. “Github: The Top 10 Programming Languages For Machine Learning”. Online. In Tech Republic. CBS Interactive. https://www.techrepublic.com/article/github-the-top-10-programming-languages-for-machine-learning/.

[56] Nautiyal, Dewang. 2019. “Top 5 Best Programming Languages For Artificial Intelligence Field”. Online. In Geeks For Geeks. https://www.geeksforgeeks.org/top-5-best-programming-languages-for-artificial-intelligence-field/.

[57] Chand, Mahesh. 2019. “Best Programming Language For Machine Learning”. Online. In C# Corner. https://www.c-sharpcorner.com/article/best-programming-language-for-machine-learning/.

[58] “SciPy”. 2019. Online. SciPy Developers. https://scipy.org/.

[59] “NumFOCUS: Open Code = Better Science”. 2019. Online. https://numfocus.org/.

[60] Pedregosa, Fabian, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos and David Cournapeau. 2011. "Scikit-Learn: Machine Learning In Python". In Journal Of Machine Learning Research 12, 2825-2830.

[61] "Cloud Computing With Amazon Web Services". 2019. Online. What Is AWS? - Amazon Web Services. https://aws.amazon.com/what-is-aws/.

[62] "What Is A Container?". 2019. Online. Enterprise Application Container Platform | Docker. https://www.docker.com/resources/what-container.

[63] Doerrfeld, Bill. 2019. "5 Container Alternatives To Docker". Online. Container Journal. https://containerjournal.com/2019/01/22/5-container-alternatives-to-docker/.

[64] Guyon, Isabelle, Steve Gunn, Asa Ben-Hur, and Gideon Dror. 2005. "Result Analysis Of The Nips 2003 Feature Selection Challenge". Online. MIT Press, 545–552. http://papers.nips.cc/paper/2728-result-analysis-of-the-nips-2003-feature-selection-challenge.pdf.

[65] Dua, Dheeru, and Casey Graff. 2017. "UCI Machine Learning Repository". Online. University Of California, Irvine. Irvine: University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

[66] Wolf, Karel. 2019. "Pražské ČVUT si postavilo nejvýkonnější superpočítač pro výzkum AI v ČR". Online. In Lupa.cz. https://www.lupa.cz/aktuality/prazske-cvut-si-postavilo-nejvykonnejsi-superpocitac-pro-vyzkum-umele-inteligence-v-cr/.

Appendix A

Contents of the CD

- **JS_DP.pdf** - this document
- **MATLAB code**
  - **graph_creator.m** - m-file for overall graph generating
  - **graph_creator2.m** - m-file for summary graph generating
- **Measured data**
  - **arcene.txt**
  - **dexter.txt**
  - **dorothea.txt**
  - **gisette.txt**
  - **madelon.txt**
  - **static.txt**