On Performance Optimization Potentials Regarding Data Classification in Forensics

Veit Köppen, Mario Hildebrandt, Martin Schäler

Faculty of Computer Science
Otto-von-Guericke-University Magdeburg
Universitätsplatz 2
39106 Magdeburg
veit.koeppen@ovgu.de
mario.hildbrandt@ovgu.de
martin.schaeler@ovgu.de

Abstract: Classification of given data sets according to a training set is one of the essentials bread and butter tools in machine learning. There are several application scenarios, reaching from the detection of spam and non-spam mails to recognition of malicious behavior, or other forensic use cases. To this end, there are several approaches that can be used to train such classifiers. Often, scientists use machine learning suites, such as WEKA, ELKI, or RapidMiner in order to try different classifiers that deliver best results. The basic purpose of these suites is their easy application and extension with new approaches. This, however, results in the property that the implementation of the classifier is and cannot be optimized with respect to response time. This is due to the different focus of these suites. However, we argue that especially in basic research, systematic testing of different promising approaches is the default approach. Thus, optimization for response time should be taken into consideration as well, especially for large scale data sets as they are common for forensic use cases. To this end, we discuss in this paper, in how far well-known approaches from databases can be applied and in how far they affect the classification result of a real-world forensic use case. The results of our analyses are points and respective approaches where such performance optimizations are most promising. As a first step, we evaluate computation times and model quality in a case study on separating latent fingerprint patterns.

1 Motivation

Data are drastically increased in a given time period. This is not only true for the number of data sets (comparable to new data entries), but also with respect to dimensionality. To get into control of this information overload, data mining techniques are used to identify patterns within the data. Different application domains require for similar techniques and therefore, can be improved as the general method is enhanced.

In our application scenario, we are interested in the identification of patterns in data that are acquired from latent fingerprints. Within the acquired scanned data a two-class classification is of interest, to identify the fingerprint trace and the background noise. As point

of origin, experts classify different cases. This supervised approach is used to learn a classification and thus, to support experts in their daily work. With a small number of scanned data sets that the expert has to check and classify, a high number of further data sets can be automatically classified.

Currently, the system works in a semi-automatic process and several manual steps have to be performed. Within this paper, we investigate the influence on system response and model quality, in terms of accuracy and precision, in the context of integrating the data and corresponding processes in a holistic system. Although a complete integration is feasible, different tools are currently used, which do not fully cooperate. Therefore, the efficiency or optimization regarding computation or response time are not in the focus of this work. With this paper, we step forward to create a cooperating and integrated environment that performs efficient with respect to model quality.

This paper is structured as follows: In the next section, we briefly present some background regarding classification and database technologies for accessing multi-dimensional data. In Section 3, we describe the case study that is the motivation for our analysis. Within Section 4, we present our evaluation on the case study data regarding optimization due to feature and data space reduction. Finally, we conclude our work in Section 5.

2 Background

In this section, we give background on classification algorithms in general. Then, we explain one of these algorithms that we apply in the remainder of this paper in more details. Finally, we introduce promising optimization approaches known from databases. We use these approaches in the remainder to discuss their optimization potential with respect to classification.

2.1 Classification Algorithms

In the context of our case study in Section 3, several classification algorithms can be utilized, see, e.g., [MKH+13]. Each of those algorithms is used for supervised learning. Such type of learning consists of a model generation based on training data, which are labeled according to a ground-truth. The utilized classification algorithms in [MKH+13] partition the feature space to resemble the distribution of each instance (data point) in this space. Afterward, the quality of the model can be evaluated using an independent set of labeled test data by comparing the decision of the classifier with the assigned label.

The utilized classification schemes from the WEKA data mining software [HFH+09] in [MKH+13] include support vector machines, multilayer perceptrons, rule based classifiers, decision trees, and ensemble classifiers. The latter ones combine multiple models in their decision process.

C4.5 decision tree

In this paper, we use the classifier J48, WEKA's [HFH+09] implementation of the fast C4.5 decision tree [Qui93], which is an improvement of the ID 3 algorithm [Qui86] and one of the most widely known decision tree classifiers for such problems. The advantage of decision trees is their comprehensiveness: the classifier's decision is a leaf reached by a path of single feature thresholds. The size of the tree is reduced by a pruning algorithm which replaces subtrees. Furthermore, this particular implementation is able to deal with missing values. In order to do that, the distribution of the available values for this particular feature is taken into account.

```
1  build_tree ( Data (R:{r_1,..,,r_n},C)
2      R: non-categorical_attributes r_1 to r_n,
3      C: categorical attribute,
4      S: training set in same schema as Data)
5  returning decision_tree;
6
7  begin
8      -- begin exceptions
9      If is_empty(S)
10         return FAILURE;
11     If only_one_category(DATA)
12         return single_node_tree(value(C));
13     If is_empty(R)
14         return single_node_tree(most_frequent_value(C));
15     -- end excpetions
16     Attribute r_d (elem of R) := largest_Gain(R,S);
17     {d_i| i=1,2, .., m} := values_of_attribute(r_d);
18     {S_i| i=1,2, .., k} := subsets(S) where in each subset value(r_d) = d_i holds;
19     decision_tree := tree(r_d) with nodes { d_1, d_2, .., d_m} pointing to trees
20     call ID3((R-{r_d}, C), S1), ID3((R-{r_d}), C, S2), .., ID3((R-{r_d}, C), S_k);
21     return decision_tree;
22 end build_tree;
```

Figure 1: Algorithm to build a C4.5 decision tree, adapted from [Qui86]

In Figure 1, we depict the general algorithm to build a C4.5 decision tree. The argument for the algorithm is a training set consisting of: (1) n non-categorical attributes R reaching from r_1 to r_n, (2) the categorical attribute (e.g., spam or not spam), and (3) a training set with the same schema. In Lines 8 to 15, the exception handling is depicted, for instance if there are only spam mails (Line 11). The actual algorithm tries to find the best attribute r_d and distributes the remaining tuples in S according to their value in r_d. For each subtree that is created in that way the algorithm is called recursively.

2.2 Approaches for Efficient Data Access

Data within databases have to be organized in such a way that they are efficiently accessed. In the case of multi-dimensional data, an intuitive order does not exist. This is even more apparent for the identification of unknown patterns, where an ordering in a multi-dimensional space always dominates some dimensions. For these reasons, differ-

ent approaches have been proposed. They can be differentiated into storage and index structures.

Typical storage improvements within the domain of Data Warehousing [KSS14a] are column-oriented storage [AMH08], Iceberg-Cube [FSGM+98], and Data Dwarf [SDRK02]. Whereas the Iceberg-Cube reduces computational effort, column-oriented storage improves the I/O with respect to the application scenario, where operations are performed in a column-oriented way. The Data Dwarf heavily reduces the stored data volume without loss of information. It combines computational effort and I/O cost for improving efficiency.

Furthermore, there exist many different index structures for specialized purposes [GBS+12]. Very well-known index structures for multi-dimensional purposes are the kd-Tree [Ben75] and R-Tree [Gut84]. Both mentioned indexes are candidates, which suffer especially from the curse of dimensionality. The curse of dimensionality is a property of large and sparsely populated high-dimensional spaces, which results in the effect that for tree-based indexes often large parts have to be taken into consideration for a query (e.g., because of node overlaps). To this end, several index structures, as the Pyramid technique [BBK98] or improved sequential scans, such as the VA-File [WB97] are proposed. In the following, we briefly explain some well-known indexes that, according to prior evaluations [SGS+13], result in a significant performance increase. A broader overview on index structures can be found in [GG98] or [BBK01]. Challenges regarding parameterization of index structures as well as implementation issues are discussed in [AKZ08, KSS14b, SGS+13].

2.2.1 Column vs. Row Stores

Traditionally, database systems store their data row-wise. That means that each tuple with all its attributes is stored and then the next tuple follows. By contrast, columnar storage means that all values of a column are stored sequentially and then the next column follows. Dependent on the access pattern of a classification algorithm, the traditional row-based storage should be replaced if, for instance, one dimension (column) is analyzed to find an optimal split in this dimension. In this case, we expect a significant performance benefit.

2.2.2 Data Dwarf

The basic idea of the Data Dwarf storage structure is to use prefix and suffix redundancies for multi-dimensional points to compress the data. For instance, the three dimensional points $A(1, 2, 3)$ and $B(1, 2, 4)$ share the same pre-fix $(1, 2, \_)$. As a result, the Dwarf has two interesting effects that are able to speed-up classifications. Firstly, due to the compression, we achieve an increased caching performance. Secondly, the access path is stable, which means that we require exactly the number of dimension look-ups to find a point (e.g., three look-ups for three dimensional points).

2.2.3 kd-Tree

A kd-Tree index is a multi-dimensional adaption of the well-known B-Tree cycling through the available dimensions. Per tree level, this index distributes the remaining points in the current subtree into two groups. One group in the left subtree where the points have a value smaller or equal than the separator value in the current dimension, while the remaining points belong to the right sub tree. The basic goal is to achieve logarithmic effort for exact match queries. In summary, this index structure can be used to efficiently access and analyze single dimensions in order to separate two classes.

2.2.4 VA-File

Tree-based index structures suffer from the curse of dimensionality. This may result in the effect that they are slower than a sequential scan. To this end, improvements of the sequential scan are proposed. The basic idea of the Vector Approximation File is to use a compressed *approximation* of the existing data set that fits into the main memory (or caches). On this compressed data an initial filter step is performed in order to minimize actual point look-ups. In how far this technique can be applied to speed-up classifications is currently unknown.

3 Case Study

As described in [HKDV14], the classification of contact-less scans of latent fingerprints is performed using a block based approach. The following subsections summarize the application scenario, the data gathering process, and a description of the feature space. We depict this process in Fig. 2. We describe the steps in the following in more detail.

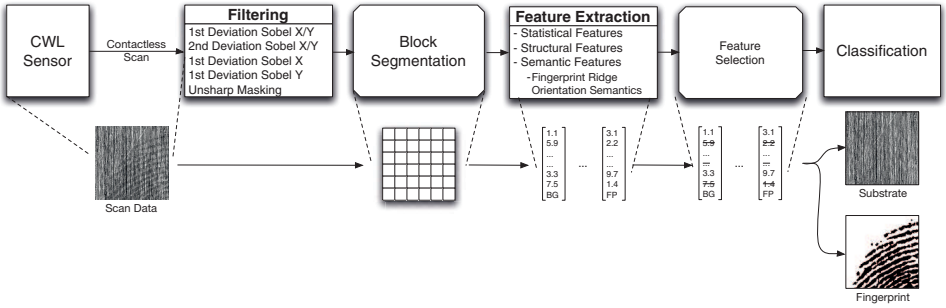

Figure 2: Data acquisition process, processing, and classification

3.1 Application scenario

The application scenario for this case study is the contact-less, non-invasive acquisition of latent fingerprints. The primary challenge of this technique is the inevitable acquisition of the substrate characteristics superimposing the fingerprint pattern. Depending on the substrate, the fingerprint can be rendered invisible. In order to allow for a forensic analysis of the fingerprint, it is necessary to differentiate between areas of the surface without fingerprint residue and others covered with fingerprint residue (fingerprint segmentation).

For this first evaluation, we solely rely on white furniture surface, because it provides a rather large difference between the substrate and the fingerprint. The achieved classification accuracy in a two-fold cross-validation based on 10 fingerprint samples is 93.1% for the J48 decision tree in [HKDV14]. The number of 10 fingerprints is sufficient for our evaluation, because we do not perform a biometric analysis. Due to the block-based classification, 1,003,000 feature vectors are extracted. For our extended 600 dimensional feature space (see Section 3.3), we achieve a classification accuracy of 90.008% based on 501,500 data sets for each of the two classes "*fingerprint*" and "*substrate*".

3.2 Data Gathering Process

The data gathering process utilizes a FRT CWL600 [Fri14] sensor mounted to a FRT MicroProf200 surface measurement device. This particular sensor exploits the effect of chromatic aberration of lenses to measure the distance and the intensity of the reflected light simultaneously. Due to this effect, the focal length of different wavelength is different. Thus, only one wavelength from the source of white light is focused at a time. This particular wavelength yields the highest intensity in the reflected light. So, it can be easily detected using a spectrometer by locating the maximum within the spectrum.

The intensity value is derived from the amplitude of this peak within the value range $[1; 4,095]$. The wavelength of the peak can be translated into a distance between the sensor and the measured object using a calibration table. The achieved resolution for this distance is 20 nm. The data itself are stored within a 16 bit integer array which can be afterward converted to a floating point distance value. The CWL600 is a point sensor which acquires the sample point-by-point while the sample is moved underneath. Thus, it is possible to select arbitrary lateral resolutions for the acquisition of the sample.

In our case study, we use a lateral dot distance of 10 μm which results in a resolution five times as high as the commonly used resolution of 500 ppi in biometric systems.

3.3 Data Description

The feature space in [HKDV14] contains statistical, structural, and fingerprint semantic features. The final feature space is extracted from the intensity and topography data (see

Section 3.2) and preprocessed versions of these data sets. Table 1 summarizes the 50 features which are extracted from each data set.

Feature Set	Features
Statistical Features	Minimum value; maximum value; span; mean value; median value; variance; skewness; kurtosis; mean squared error; entropy; globally and locally normalized values of absolute min, max, median; globally and locally normalized values of relative min, max, span, median; globally normalized absolute and relative mean value of B
Structural Features	Covariance of upper and lower half of a block B; covariance of left and right half of the block B; line variance of a block B; column variance of a block B; most significant digit frequency derived from Benford's Law [Ben38] (9 features); Hu moments [Hu62] (7 features)
Fingerprint Semantic Features	Maximum standard deviation in B_ϵ after Gabor filtering; mean value of the block B for the highest Gabor response

Table 1: Overview of the extracted features

All features are extracted from blocks with a size of 5×5 pixels with the exception of the fingerprint semantic feature of the maximum standard deviation in B_ϵ after Gabor filtering. The fingerprint semantic features are motivated by the fingerprint enhancement, e.g. [HWJ98], which utilize Gabor filters for emphasizing the fingerprint pattern after determining the local ridge orientation and frequency. Since this filtering relies on a ridge valley pattern, it requires larger blocks. In particular, we use a block size of 1.55 by 1.55 mm (155×155 pixels) as suggested in [HKDV14].

The features are extracted from the original and pre-processed data. In particular, the intensity and topography data are pre-processed using Sobel operators in first and second order in X and Y direction combined, Sobel operators in first order in X, as well as Y direction separately, and unsharp masking (subtraction of a blurred version of the data).

In result, we get a 600-dimensional feature space. However, some of the features cannot be determined, e.g., due to a division by zero in case of the relative statistical features. Thus, either the classifier must be able to deal with missing values, or those features need to be excluded. To this end, we apply the J48 classifier, because it handles missing data.

4 Evaluation

In this section, we present the evaluation of the classification according to the J48 algorithm. We restrict this study to performance measurements of computation time for

	Condition positive	Condition negative
Test outcome positive	True Positive (TP)	False Positive (FP)
Test outcome negative	False Negative (FN)	True Negatives (TN)

Table 2: Contingency table

building the model and for the evaluation of the model. As influences on the performance, we identify according to Section 2.2 cardinality of the dimensions and involved features. Therefore, we investigate the model quality with respect to precision and recall. First, we present our evaluation setup. This is followed by the result presentation. Finally, we discuss our findings.

4.1 Setup

Our data are preprocessed as described in Section 3. We use the implementation of C4.5 [Qui93] in WEKA, which is called J48. For the identification of relationships between included feature dimensions and feature cardinality and model build time and model evaluation time, we use different performance measurements regarding the model. We briefly describe the model performance measurements in the following.

In classification, the candidates can be classified correctly or incorrectly. Compared to the test population four cases are possible, as presented in Table 2.

In the following we define measures that can be derived from the contingency table. The recall (also called sensitivity or true positive rate) represents the correctly identified positive elements compared to all identified positive elements. This measure is defined as:

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{1}$$

Accuracy describes all correctly classified positive and negative elements compared to all elements. This measure assumes a non-skewed distribution of classes within the learning as well as training data. It is defined as:

$$\textit{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{2}$$

Precision is also called positive prediction rate and measures all correctly identified positives compared to all positives in the ground truth. It is defined as:

$$\textit{Precision} = \frac{TP}{TP + FP} \tag{3}$$

Specificity is also called true negative rate and is a ratio comparing the correctly classified negative elements to all negative classified elements. It is defined as:

$$\textit{Specificity} = \frac{TN}{FP + TN} \tag{4}$$

The measure Balanced Accuracy is applied in the case that the classes are not equally distributed. This takes non-symmetric distributions into account. The balance is achieved by computing the arithmetic mean of Recall and Specificity and it is defined as:

$$\textit{Balanced Accuracy} = \frac{\textit{Recall} + \textit{Specificity}}{2} = \frac{1}{2} \cdot \left(\frac{TP}{TP + FN} + \frac{FN}{FP + TN} \right) \quad (5)$$

The F-Measure is the harmonic mean of precision and recall to deal with both interacting indicators at the same time. This results in:

$$\textit{F-Measure} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (6)$$

Depending on the application scenario, a performance measure can be used for optimization. In Fig. 3, we depict all above stated performance measurements according to a filtering of our data set.

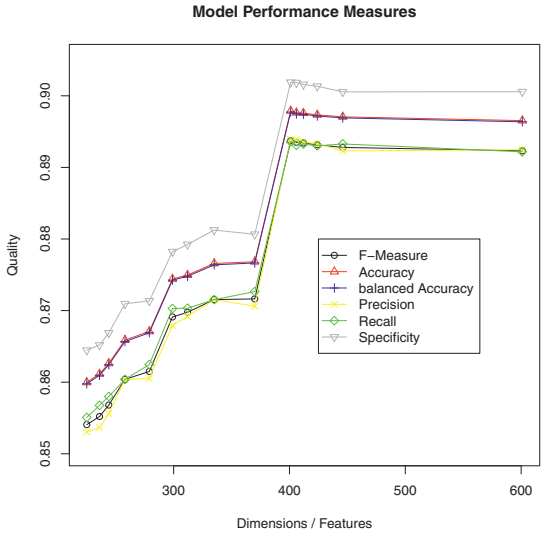

Figure 3: Performance Measures for different filters of the test case

In our evaluation, we investigate two different performance influences. On the one side, we are interested in filtering out correlated data columns. At the other side, we measure performance for a restricted data space domain. This is applied by a data discretization.

Evaluation is based on three important aspects:

- Building the model in terms of computation time,
- Testing the model in terms of computation time, and
- Quality of the model measured in model performance indicators.

From Fig. 3 it can be identified, that the computed models have a higher specificity than recall. This results also in a lower F-Measure. Furthermore, it can be seen that the training data are not imbalanced and accuracy is very close to balanced accuracy. However, all values are close and at an acceptable range. Therefore, we use for the remainder of our result presentation the F-Measure as model performance measure.

For reducing the dimensional space, we secondly discretize each feature. This is computed in such a way that the variance within a feature is retained as best as possible. Currently, there are no data structures within the WEKA environment, that use restricted data spaces efficiently. Therefore, we assume that model creation and model evaluation times are not significantly influenced. However, as a database system can be used in future, the question arises, which quality influence on model performance is achieved by discretization. Therefore, we conduct an evaluation series with discretized feature dimensions, where all feature dimensions are restricted to the following cardinalities:

- 8 values,
- 16 values,
- 32 values,
- 64 values,
- 128 values,
- 256 values,
- 512 values,
- 1,024 values,
- 2,048 values, and
- full cardinality.

4.2 Result Presentation

As a first investigation of our evaluation scenario, we present results regarding the elimination of features. For the feature elimination we decide for an statistical approach, where correlated data columns are eliminated from the data set.

In Fig. 4, we present the dimensions that are included in the data set. At the x-axis, we present the correlation criteria that are used for elimination. For instance, a correlation criteria of 99% means that all data columns are eliminated from the data set that have a correlation of 0.99 to another feature within the data set. Note, we compare every feature column with every other and at an elimination decision; we left the first in the data set. Therefore, we prefer the first data columns within the data set. Furthermore, we also tested the feature reduction for discretized data sets. With a small cardinality, the feature reduction due to correlation is lower, which means that the dimensional space is higher compared to the others.

We evaluate in the following the reduction of the feature space in terms of computational effort. We differentiate at this point two cases for this effort: On the one side the model building time represents computational performance for creating (learning) the model. As the amount of data tuples for learning the model we use 501,500 elements. As a second measurement, we present evaluation times where 501,500 further elements are used in a testing phase of the model. This additionally leads to the quality indicators of the model presented in Section 4.1. We present this information afterward.
In Fig. 5, we present the model creation times for different data sets. With a decrease of the feature space, the computation time reduces, too. However, there are some saltus identifi-

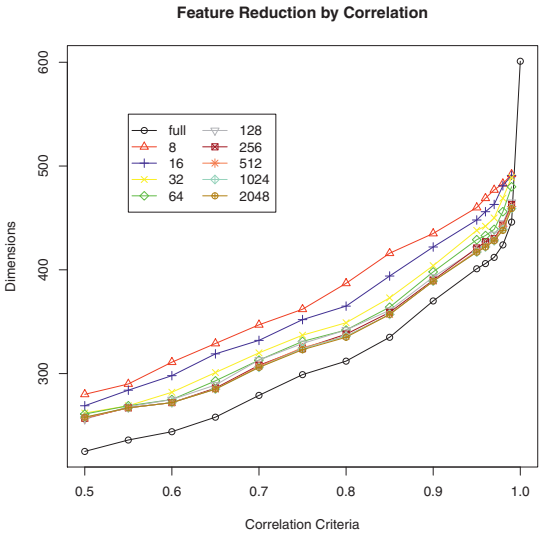

Figure 4: Feature Reduction by Correlation

able. These are related to the fact, that the algorithm has a dynamic model complexity. This means that the number of nodes within the model is not restricted and therefore, smaller models can have a faster generation time. Nevertheless, we do not focus on optimization for our case study, but we derive a general relationship. From our data, we can derive that a decrease is reduced for data that are not more than 85% correlated. This leads to a slower reduction in computation time. However, with this elimination of 85% correlated values, the computational effort is reduced to approximately one third. An important result from the model generation: a restrictive discretization (cases 8 and 16) does negatively influences the model building time. Note, we do not use in our evaluation an optimized data structure, which has a significant influence on the computational performance, see also Section 2.2. Although the underlying data structure is general, a restriction of the feature cardinalities improves model building times for the cases cardinality 32 and higher.

For evaluation times of the model a similar behavior is identifiable. In Fig. 6, we present the evaluation times for the same data sets. Two major differences can be easily seen: On the one side, the difference between the test cases is smaller and the slopes are smoother. On the other side, a reduction of the evaluation time is optimal for cardinalities of 32 and 64. An increase of the cardinalities leads to a higher computational effort. This is respected to the fact that the sequential searches within the data are quite important for the testing phase of a model. A usage of efficient data structures should therefore be in focus of future studies.

With both above presented evaluations, we only have computation time in the focus. However, we have to respect the quality of the model at the same time. Within classification applications, an increased information usage (in terms of data attributes) can increase the model quality. A reduction of the information space might lead to a lower model quality.

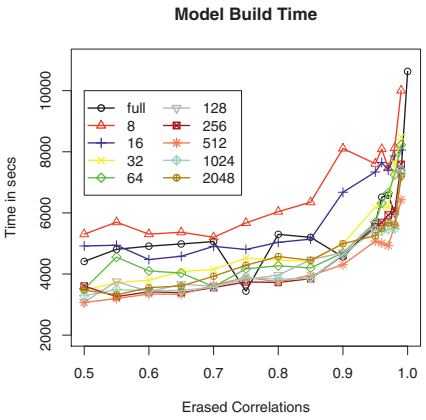

Figure 5: Model Build Time

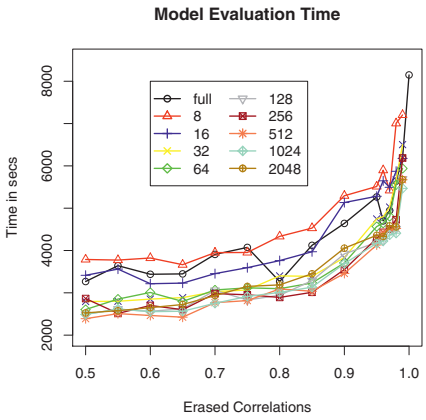

Figure 6: Model Evaluation Times

In Fig. 7, we show the relationship between erased correlations and the F-Measure. Note, that an increase in the F-Measure is also for a reduced data space possible (e.g., in the case of full cardinality). With a reduced cardinality in the information space, a lower F-Measure is achieved. This is especially true for low cardinalities (e.g., 8 or 16). However, in the case that the cardinality is reduced from a correlation of 0.95 to 0.9 within the data set a higher decrease in the F-Measure is identifiable. A second significant reduction of the F-measure is at the 0.7 correlation elimination level.

In Fig. 8, we present the relationship between model build times and the model quality. Although a negative dependency is assumed, this trend is only applicable to some parts of the evaluation space. As an optimization of model quality and computation time, the first high decrease model quality is at an elimination of 0.95 correlated values. Further eliminations do not influence the model build times in a similar decrease.

Overall, we have to state that our reduction of the data space is quite high compared to the reduction of the model quality in terms of the F-measure. Note, other model performance measures are quite similar.

4.3 Discussion

With our evaluation, we focus on the influences of the data space to model performance in terms of quality and computation times. Therefore, we reduce the information space in two ways. On the one hand, we restrict dimensionality by applying a feature reduction by correlation. This is also called canonical correlation analysis. It can be computed in a very efficient way and therefore, it is much faster than other feature reduction techniques, e.g., principal component analysis or linear discriminant analysis. Furthermore, we restrict the cardinality of the feature spaces, too. We discretize the feature space and are interested in

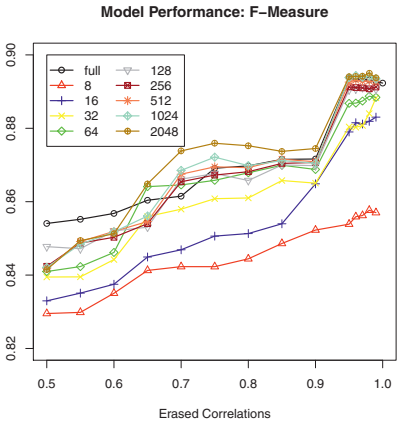

Figure 7: Model Quality and Reduction

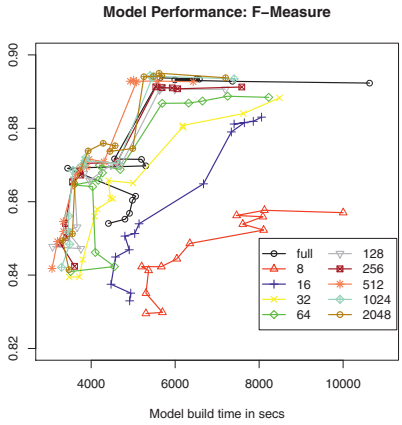

Figure 8: Model Quality and Build Times

the influence on model quality. An influence for the model build times are not assumed, due to the fact that the underlying data structures are not optimized.

We focus this in future work, cf. [BDKM14]. Due to the column-wise data processing of the classifiers, we assume that a change in the underlying storage structure, e.g., column stores or Data Dwarfs, leads to a significant computational performance increase. First analyzes of the WEKA implementation reveal a high integration effort. However, the benefits are very promising.

5 Conclusion

We present some ideas on improving model quality and computational performance for a classification problem. This work is a starting point to enhance the process with respect to optimize computation times in a biometric scenario. Additional use cases, e.g., indicator simulation [KL05], other data mining techniques [HK00], or operations in a privacy secure environment [DKK+14], can be applied to our main idea and have to be considered for filtering and reduction techniques. With our evaluation study we show that performance with respect to computation times as well as model quality can be optimized. However, a trade-off between both targets has to be achieved due to inter-dependencies.

In future work, we want to improve the process by integrating and optimizing the different steps. We assume, an efficient data access structure is beneficial for model computation times and therefore, increases the application scenario. However, this computational improvement relies on the information space, especially on dimensional cardinality and number of involved dimensions. With an easy to apply algorithm, a data processing enables a fast transformation of the feature space and smooth the way for more efficient data mining for forensic scenarios.

Acknowledgment

The work in this paper has been funded in part by the German Federal Ministry of Education and Research (BMBF) through the Research Program ”DigiDak+ Sicherheits-Forschungskolleg Digitale Formspuren” under Contract No. FKZ: 13N10818.

References

[AKZ08] Elke Achtert, Hans-Peter Kriegel, and Arthur Zimek. ELKI: A Software System for Evaluation of Subspace Clustering Algorithms. In *SSDBM*, LNCS (5069), pages 580–585. Springer, 2008.

[AMH08] Daniel J. Abadi, Samuel Madden, and Nabil Hachem. Column-Stores vs. Row-Stores: How different are they really? In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 967–980, Vancouver, BC, Kanada, 2008.

[BBK98] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The Pyramid-technique: Towards Breaking the Curse of Dimensionality. *SIGMOD Rec.*, 27(2):142–153, 1998.

[BBK01] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.

[BDKM14] David Broneske, Sebastian Dorok, Veit Köppen, and Andreas Meister. Software Design Approaches for Mastering Variability in Database Systems. In *GvDB*, 2014.

[Ben38] Frank Benford. The Law of Anomalous Numbers. *Proceedings of the American Philosophical Society*, 78(4):551–572, 1938.

[Ben75] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, 1975.

[DKK+14] Jana Dittmann, Veit Köppen, Christian Krätzer, Martin Leuckert, Gunter Saake, and Claus Vielhauer. Performance Impacts in Database Privacy-Preserving Biometric Authentication. In Rainer Falk and Carlos Becker Westphall, editors, *SECURWARE 2014: The Eighth International Conference on Emerging Security Information, Systems and Technologies*, pages 111–117. IARA, 2014.

[Fri14] Fries Research & Technology GmbH. Chromatic White Light Sensor CWL, 2014. http://www.frt-gmbh.com/en/chromatic-white-light-sensor-frt-cwl.aspx.

[FSGM+98] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing Iceberg Queries Efficiently. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB ’98, pages 299–310, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[GBS+12] Alexander Grebhahn, David Broneske, Martin Schäler, Reimar Schröter, Veit Köppen, and Gunter Saake. Challenges in finding an appropriate multi-dimensional index structure with respect to specific use cases. In Ingo Schmitt, Sascha Saretz, and Marcel Zierenberg, editors, *Proceedings of the 24th GI-Workshop ”Grundlagen von Datenbanken 2012”*, pages 77–82. CEUR-WS, 2012. urn:nbn:de:0074-850-4.

[GG98] Volker Gaede and Oliver Günther. Multidimensional Access Methods. *ACM Comput. Surv.*, 30:170–231, 1998.

[Gut84] Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD Rec.*, 14(2):47–57, 1984.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10 – 18, 2009.

[HK00] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[HKDV14] Mario Hildebrandt, Stefan Kiltz, Jana Dittmann, and Claus Vielhauer. An enhanced feature set for pattern recognition based contrast enhancement of contact-less captured latent fingerprints in digitized crime scene forensics. In Adnan M. Alattar, Nasir D. Memon, and Chad D. Heitzenrater, editors, *SPIE Proceedings: Media Watermarking, Security, and Forensics*, volume 9028, pages 08/01–08/15, 2014.

[Hu62] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[HWJ98] Lin Hong, Yifei Wan, and A. Jain. Fingerprint image enhancement: algorithm and performance evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):777 –789, aug 1998.

[KL05] Veit Köppen and Hans-J. Lenz. Simulation of Non-linear Stochastic Equation Systems. In S.M. Ermakov, V.B. Melas, and A.N. Pepelyshev, editors, *Proceeding of the Fifth Workshop on Simulation*, pages 373–378, St. Petersburg, Russia, July 2005. NII Chemistry Saint Petersburg University Publishers.

[KSS14a] Veit Köppen, Gunter Saake, and Kai-Uwe Sattler. *Data Warehouse Technologien*. MITP, 2 edition, Mai 2014.

[KSS14b] Veit Köppen, Martin Schäler, and Reimar Schröter. Toward Variability Management to Tailor High Dimensional Index Implementations. In *RCIS*, pages 452–457. IEEE, 2014.

[MKH+13] Andrey Makrushin, Tobias Kiertscher, Mario Hildebrandt, Jana Dittmann, and Claus Vielhauer. Visibility enhancement and validation of segmented latent fingerprints in crime scene forensics. In *SPIE Proceedings: Media Watermarking, Security, and Forensics*, volume 8665, 2013.

[Qui86] John Ross Quinlan. Induction of Decision Trees. *Mach. Learn.*, 1(1):81–106, 1986.

[Qui93] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[SDRK02] Yannis Sismanis, Antonios Deligiannakis, Nick Roussopoulos, and Yannis Kotidis. Dwarf: Shrinking the PetaCube. In *SIGMOD*, pages 464–475. ACM, 2002.

[SGS+13] Martin Schäler, Alexander Grebhahn, Reimar Schröter, Sandro Schulze, Veit Köppen, and Gunter Saake. QuEval: Beyond high-dimensional indexing à la carte. *PVLDB*, 6(14):1654–1665, 2013.

[WB97] Roger Weber and Stephen Blott. An Approximation-Based Data Structure for Similarity Search. Technical Report ESPRIT project, no. 9141, ETH Zürich, 1997.