ibai-publishing
Prof. Dr. Petra Perner
PF 30 11 38
04251 Leipzig, Germany
E-mail: info@ibai-publishing.org

P-ISSN 1864-9734
E-ISSN 2699-5220
ISBN 978-3-942952-82-8

www.ibai-publishing.org

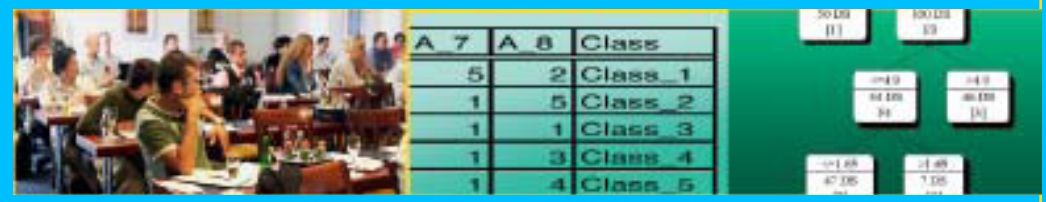
ISBN 978-3-942952-82-8

9 783942 952828

Advances in Data Mining, Proceedings, ICDM 2021 — Petra Perner

Petra Perner (Ed.)

Advances in Data Mining



20th Industrial Conference on Data Mining, ICDM 2021
New York, USA July 14-18, 2021

Proceedings

ibai - publishing

Petra Perner (Ed.)

Advances in Data Mining

Applications and Theoretical Aspects

21th Industrial Conference, ICDM 2021
New York, USA, July 14 – 18 2021
Proceedings

ibai Publishing
www.ibai-publishing.org

Volume Editor

Prof. Dr. Petra Perner
FutureLab Artificial Intelligence IBaI-2

PF 10128
01446 Radeberg
E-mail: pperner@futurelab-ai-ibai-2.de

The German National Library listed this publication in the German National Bibliography.
Detailed bibliographical data can be downloaded from http://dnb.ddb.de.

ibai-publishing
Prof. Dr. Petra Perner
PF 30 11 38
04251 Leipzig, Germany
E-mail: info@ibai-publishing.org
http://www.ibai-publishing.org

P-ISSN 1864-9734
E-ISSN 2699-5220
ISBN 978-3-942952-82-8

Preface

The twenty-one event of the Industrial Conference on Data Mining ICDM was held in New York again (www.data-mining-forum.de) running under the umbrella of the World Congress on "The Frontiers in Intelligent Data and Signal Analysis, DSA 2021" (www.worldcongressdsa.com).

After the peer-review process, we accepted twenty-four high-quality papers for oral presentation. Ten papers are presented in the proceedings by ibai-publishing. http://www.ibai-publishing.org/html/proceedings_2021/proceedings_icdm_2021.php The topics range from theoretical aspects of data mining to applications of data mining, such as in multimedia data, in marketing, in medicine and agriculture, and in process control, industry, and society. Extended versions of selected papers will appear in the international journal Transactions on Machine Learning and Data Mining (www.ibai-publishing.org/journal/mldm).

In all, six papers were selected for poster presentations and two are published in the ICDM Poster and Industry Proceeding by ibai-publishing (http://www.ibai-publishing.org/html/proceedings_2021/proceedings_poster_icdm_2021.php).

The acceptance rate was 20%. The list of accepted papers can be found after the conference under https://www.data-mining-forum.de/past_reports_ov.php. The Corona situation was still in our control after one and a half years.

The tutorial days rounded up the high quality of the conference. Researchers and practitioner`s got an excellent insight in the research and technology of the respective fields, the new trends and the open research problems that we like to study further.
A tutorial on Data Mining, a tutorial on Case-Based Reasoning, a tutorial on Intelligent Image Interpretation and Computer Vision in Medicine, Biotechnology, Chemistry and Food Industry, and a tutorial on Standardization in Immunofluorescence were held before and in between the conferences of DSA 2021.

We would like to thank all reviewers for their highly professional work and their effort in reviewing the papers.

We also thank the members of the Futurelab Artificial Intelligence IBaI-2, Radeberg, Germany (www.futurelab-ai-ibai-2.de), who handled the conference as secretariat. We appreciate the help and understanding of the editorial staff at ibai-publishing house, who supported the publication of these proceedings (http://www.ibai-publishing.org/html/proceeding.php).

Last, but not least, we wish to thank all the speakers and participants who contributed to the success of the conference. We hope to see you in 2022 in New York at the next World Congress on The Frontiers in Intelligent Data and Signal Analysis, DSA

2022 (www.worldcongressdsa.com), which combines under its roof the following three events: International Conferences Machine Learning and Data Mining, MLDM (www.mldm.de), the Industrial Conference on Data Mining, ICDM (www.data-mining-forum.de, and the International Conference on Mass Data Analysis of Signals and Images in Medicine, Biotechnology, Chemistry, Biometry, Security, Agriculture, Drug Discovery and Food Industry, MDA (www.mda-signals.de), the workshops and tutorials.

July 2021 Petra Perner

21th Industrial Conference on Data Mining
ICDM 2021

www.data-mining-forum.de

Chair

Petra Perner	FutureLab Artificial Intelligence IBaI-2,
.....	Germany

Program Committee

Ajith Abraham	Machine Intelligence Research Labs (MIR Labs), USA
Mohamed, Bourguessa	Universite du Quebec a Montreal - UQAM, Canada
Bernard Chen	University of Central Arkansas, USA
Antonio Dourado	University of Coimbra, Portugal
Jeroen de Bruin	Medical University of Vienna, Austria
Stefano Ferilli	University of Bari, Italy
Geert Gins	Glaxo Smith Kline, Belgium
Warwick Graco	ATO, Australia
Aleksandra Gruca	Silesian University of Technology, Poland
Pedro Isaias	Universidade Aberta (Portuguese Open University), Portugal
Piotr Jedrzejowicz	Gdynia Maritime University, Poland
Martti Juhola	University of Tampere, Finland
Janusz Kacprzyk	Polish Academy of Sciences, Poland
Lui Xiaobing	Google Inc., USA
Mehmed Kantardzic	University of Louisville, USA
Eduardo F. Morales	INAOE, Ciencias Computacionales, Mexico
Samuel Noriega	Universitat de Barcelona, Spain
Juliane Perner	Cancer Research, Cambridge Institutes, UK
Moti Schneider	Netanya Academic College, Israel
Rainer Schmidt	University of Rostock, Germany
Victor Sheng	University of Central Arkansas, USA
Kaoru Shimada	Section of Medical Statistics, Fukuoka Dental College, Japan
Gero Szepannek	University Stralsund, Germany

Joao Miguel Costa Sousa	Technical University of Lisbon, Portugal
Markus Vattulainen	Tampere University, Finnland
Zhu Bing	Sichuan University, China

Table of Content

adfa, p. 1, 2011.

BERTSurv: BERT-Based Survival Models for Predicting Outcomes of Trauma Patients

Yun Zhao[1], Qinghang Hong[1], Xinlu Zhang[1], Yu Deng[2], Yuqing Wang[1], and Linda Petzold[1]

[1] Department of Computer Science, University of California, Santa Barbara
[2] Feinberg School of Medicine, Northwestern University
yunzhao@cs.ucsb.edu

Abstract. Survival analysis is a technique to predict the times of specific outcomes, and is widely used in predicting the outcomes for intensive care unit (ICU) trauma patients. Recently, deep learning models have drawn increasing attention in healthcare. However, there is a lack of deep learning methods that can model the relationship between measurements, clinical notes and mortality outcomes. In this paper we introduce BERTSurv, a deep learning survival framework which applies Bidirectional Encoder Representations from Transformers (BERT) as a language representation model on unstructured clinical notes, for mortality prediction and survival analysis. We also incorporate clinical measurements in BERTSurv. With binary cross-entropy (BCE) loss, BERTSurv can predict mortality as a binary outcome (mortality prediction). With partial log-likelihood (PLL) loss, BERTSurv predicts the probability of mortality as a time-to-event outcome (survival analysis). We apply BERTSurv on Medical Information Mart for Intensive Care III (MIMIC III) trauma patient data. For mortality prediction, BERTSurv obtained an area under the curve of receiver operating characteristic curve (AUC-ROC) of 0.86, which is an improvement of 3.6% over baseline of multilayer perceptron (MLP) without notes. For survival analysis, BERTSurv achieved a concordance index (C-index) of 0.7. In addition, visualizations of BERT's attention heads help to extract patterns in clinical notes and improve model interpretability by showing how the model assigns weights to different inputs.

Keywords: Deep learning · BERT · Survival analysis · Mortality prediction.

1 Introduction

Trauma is the leading cause of death from age 1 to 44. More than 180,000 deaths from trauma occur each year in the United States [1]. Most trauma patients die or are discharged quickly after being admitted to the ICU. Care in the first few

hours after admission is critical to patient outcome, yet this time period is more prone to medical decision errors in ICUs [2] than later periods. Therefore, early and accurate prediction for trauma patient outcomes is essential for ICU decision making.

Medical practitioners use survival models to predict the outcomes for trauma patients [3]. Survival analysis is a technique to model the distribution of the outcome time. The Cox model [4] is one of the most widely used survival models with linear proportional hazards. Faraggi-Simon's network [5] is an extension of the Cox model to nonlinear proportional hazards using a neural network. DeepSurv [6] models interactions between a patient's covariates and treatment effectiveness with a Cox proportional hazards deep neural network. However, these existing models deal only with well-structured measurements and do not incorporate information from unstructured clinical notes, which can offer significant insight into patients' conditions.

The transformer architecture has taken over sequence transduction tasks in natural language processing (NLP). Transformer is a sequence model that adopts a fully attention-based approach instead of traditional recurrent architectures. Based on Transformer, BERT [8] was proposed for language representation and achieved state-of-the-art performance on many NLP tasks. There has also been increasing interest in applying deep learning to end-to-end e-health data analysis [9]. Biobert [10] extends BERT to model biomedical language representation. Med-BERT [11] modifies BERT by leveraging domain specific hierarchical code embedding and layer representation to generate sequential relationships in the clinical domain. G-BERT [12] combines Graph Neural Networks (GNNs) and BERT for medical code representation and medication recommendation. Clinical BERT [13, 14] explores and pre-trains BERT using clinical notes. Clearly, there is an unmet need to include unstructured text information in deep learning survival models for patient outcome predictions.

In this paper we propose BERTSurv, a deep learning survival framework for trauma patients which incorporates clinical notes and measurements for outcome prediction. BERTSurv allows for both mortality prediction and survival analysis by using BCE and PLL loss, respectively. Our experimental results indicate that BERTSurv can achieve an AUC-ROC of 0.86, which is an improvement of 3.6% over the baseline of MLP without notes on mortality prediction.

The key contributions of this paper are:

1. We propose BERTSurv: a BERT-based deep learning framework to predict the risk of death for trauma patients. To the best of our knowledge, this is the first paper applying BERT on unstructured text data combined with measurements for survival analysis.

2. We evaluate BERTSurv on the trauma patients in MIMIC III. For mortality prediction, BERTSurv achieves an AUC-ROC of 0.86, which outperforms baseline of MLP without notes by 3.6%. For survival analysis, BERTSurv achieved a C-index of 0.7 on trauma patients, which outperforms a Cox model with a C-index of 0.68.

3. We extract patterns in clinical notes by performing attention mechanism visualization, which improves model interpretability by showing how the model assigns weights to different clinical input texts with respect to survival outcomes.

This paper is organized as follows: Section 2 describes how we processed the MIMIC trauma dataset. We present BERTSurv in Section 3.1 and describe the background of BERT and survival analysis in Section 3.2 and Section 3.3. Evaluation and discussion are given in Sections 4 and 5, respectively.

2 Dataset

BERTSurv is applied to the data from trauma patients selected using the ICD-9 code from the publicly available MIMIC III dataset [16], which provides extensive electronic medical records for ICU admissions at the Beth Israel Deaconess Medical Center between 2001 and 2012. The measurements, clinical notes, expire flag (0 for discharge and 1 for death), and death/discharge time for each patient were used to train and test BERTSurv. The patient data were aggregated over the first 4 hours to obtain the initial state of each individual admission. We took the average for each of the measurements taken during this time period, and concatenated all of the clinical notes together. Considering the missing value issue and redundancy in MIMIC III, we selected 21 common features as our representative set: blood pressure, temperature, respiratory rate, arterial PaO2, hematocrit, WBC, creatinine, chloride, lactic acid, BUN, sodium (Na), glucose, PaCO2, pH, GCS, heart rate, FiO2, potassium, calcium, PTT and INR. Our feature set overlaps 65% of the measurements required by APACHE III [15]. We also extracted 4 demographic predictors: weight, gender, ethnicity and age.

As is common in medical data, MIMIC III contains many missing values in the measurements, and the notes are not well-formatted. Thus, data preprocessing is very important to predict outcomes. To deal with the missing data issue, we first removed patients who have a missing value proportion greater than 0.4 and then applied MICE [17] data imputation for the remainder of the missing values. For the clinical notes, we removed formatting, punctuation, non-punctuation symbols and stop words. In addition to the most commonly used English stop words, our stop word dictionary includes a few specific clinical and trauma related stop words: *doctor, nurse and measurement*, etc. Following this preprocessing, our trauma dataset includes 1860 ICU patients, with 21 endogenous measurements, 4 exogenous measurements and notes. The sample class ratio between class 0 (discharge) and class 1 (death) is $1206 : 654$.

3 Methods

In this section we first describe the framework of BERTSurv. Then we introduce some basics of BERT and survival analysis, which are the key components of BERTSurv.

3.1 BERTSurv

Our model architecture, shown in Fig 1, consists of BERT embedding of clinical notes concatenated with measurements followed by feed forward layers. The output for BERTSurv is a single node $h_\theta(x_i)$ parameterized by the weights of the neural network θ, which estimates either the probability of mortality or the hazard risk. For mortality prediction, we apply BCE loss to predict outcomes of death or discharge:

$$\text{BCELoss} := \sum_{i=1}^{n} p(y_i) \log(h_\theta(x_i)), \tag{1}$$

where x_i and y_i represent inputs and outcomes for the ith patient, respectively.

To estimate θ in survival analysis, similar to the Faraggi-Simon network [5,6], we minimize the PLL loss function, which is the average negative log partial likelihood:

$$\text{PLLLoss} := -\frac{1}{N_{D=1}} \sum_{i:D_i=1} (h_\theta(x_i) - \log \sum_{j \in R(T_i)} \exp(h_\theta(x_j))), \tag{2}$$

where $N_{D=1}$ is the number of patients with an observable death. The risk set $R_i = \{j : T_j \geq T_i\}$ is the set of those patients under risk at T_i.

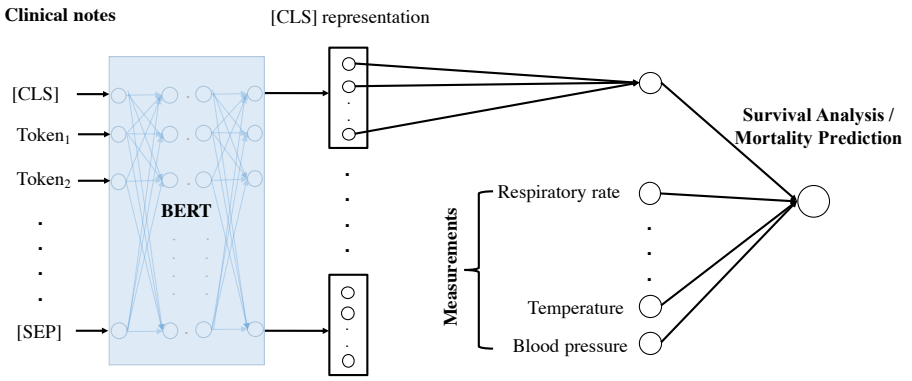


Fig. 1: The framework of BERTSurv. [CLS] is a special symbol added in front of every clinical note sample, and [SEP] stands for a special separator token. BERTSurv consists of three main parts: BERT, measurements and output layer for mortality prediction or survival analysis. First, we input a set of diagnostics and nurse notes to BERT pretrained on masked language modeling and next sentence prediction. The [CLS] representation, is treated as the representation of the input notes. Then we concatenate the [CLS] representation and measurements as input and fine-tune BERTSurv for downstream survival analysis.

We use batch normalization through normalization of the input layer by re-centering and re-scaling [18]. We apply rectified linear unit(ReLU) or scaled exponential linear units (SELU) as the activation function. For regularization, dropout [19] is implemented to avoid overfitting. Dropout prevents co-adaptation of hidden units by randomly dropping out a proportion of the hidden units during backpropagation. BCE/PLL loss is minimized with the Adam optimizer [20] for training.

BERTSurv is implemented in Pytorch [21]. We use a Dell 32GB NVIDIA Tesla M10 Graphics Card GPU (and significant CPU power and memory for pre-processing tasks) for training, validation and testing. The hyperparameters of the network include: BERT choice ($\text{BERT}_{\text{BASE}}$ or clinical BERT [13]), sequence length, batch size, learning rate, dropout rate, training epochs and activation function (ReLU or SELU).

3.2 BERT

A key component of BERTSurv is the BERT language representation model. BERT is a Transformer-based language representation model, which is designed to pre-train deep bidirectional representations from unlabeled text by jointly considering context from both directions (left and right). Using BERT, the input representation for each token in the clinical notes is the sum of the corresponding token embeddings, segmentation embeddings and position embeddings. WordPiece token embeddings [22] with a 30,000 token vocabulary are applied as input to BERT. The segment embeddings identify which sentence the token is associated with. The position embeddings of a token are a learned set of parameters corresponding to the token's position in the input sequence. An attention function maps a query and a set of key-value pairs to an output. The attention function takes a set of queries Q, keys K, and values V as inputs and is computed on an input sequence using the embeddings associated with the input tokens. To construct Q, K and V, every input embedding is multiplied by the learned sets of weights. The attention function is

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}V), \tag{3}$$

where d_k is the dimensionality of Q and K. The dimension of V is d_v. A multi-head attention mechanism allows BERT to jointly deal with information from different representation subspaces at different positions with several (h) attention layers running in parallel:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O, \tag{4}$$

where $\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. Parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are the learned linear projections from Q, K, V to d_k, d_k and d_v dimensions.

In BERTSurv, we use pretrained BERT of $\text{BERT}_{\text{BASE}}$ and clinical BERT [13] for clinical note embedding, and focus on fine-tuning for survival analysis.

3.3 Survival Analysis

Another key component of BERTSurv is survival analysis. Survival analysis [23, 24] is a statistical methodology for analyzing the expected duration until one or more events occur. The survival function $S(t)$, defined as $S(t) = P(T \geq t)$, gives the probability that the time to the event occurs later than a given time t. The cumulative distribution function (CDF) of the time to event gives the cumulative probability for a given t-value:

$$F(t) = P(T < t) = 1 - S(t). \tag{5}$$

The hazard function $h(t)$ models the probability that an event will occur in the time interval $[t, t + \Delta t)$ given that the event has not occurred before:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t} = \frac{f(t)}{S(t)}, \tag{6}$$

where $f(t)$ is the probability density function (PDF) of the time to event. A greater hazard implies a greater probability of event occurrence. Note from Equ. 5 that $-f(t)$ is the derivative of $S(t)$. Thus Equ. 6 can be rewritten as

$$h(t) = -\frac{dS(t)}{dt} * \frac{1}{S(t)} = -\frac{d}{dt} \log(S(t)). \tag{7}$$

By solving Equ. 7 and introducing the boundary condition $S(0) = 1$ (the event can not occur before duration 0), the relationship between $S(t)$ and $h(t)$ is given by

$$S(t) = \exp\left(-\int_0^t h(s)\mathrm{d}s\right). \tag{8}$$

The Cox model [4] is a well-recognized survival model. It defines the hazard function given input data $h(t \mid \mathbf{y}, \boldsymbol{\eta})$ to be the product of a baseline function, which is a function of time, and a parametric function of the input data $\mathbf{y}$ and $\boldsymbol{\eta}$. $\mathbf{y}$ and $\boldsymbol{\eta}$ denote endogenous measurements and exogenous measurements, respectively. Using the assumption of a linear relationship between the log-risk function and the covariates, the Cox model has the form

$$h(t \mid \mathbf{y}, \boldsymbol{\eta}) = h_0(t) \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta}), \tag{9}$$

where $h_0(t)$ is the baseline hazard function, and $\boldsymbol{\tau}$ and $\boldsymbol{\gamma}$ are the vectors of weights for $\mathbf{y}$ and $\boldsymbol{\eta}$.

In BERTSurv, the log-risk function $h_\theta(\mathbf{x})$ is the output node from the neural network:

$$h(t \mid \mathbf{y}, \boldsymbol{\eta}) = h_0(t) \exp(h_\theta(\mathbf{x})), \tag{10}$$

where the input $\mathbf{x}$ includes $\mathbf{y}$, $\boldsymbol{\eta}$ and clinical notes. The likelihood function for the survival model is as follows:

$$p(T, \delta) = h(T)^\delta S(T). \tag{11}$$

When $\delta = 1$, it means that the event is observed at time T. When $\delta = 0$, the event has not occurred before T and it will be unknown after T. The time T when $\delta = 0$ is called the censoring time, which means the event is no longer observable.

The Cox partial likelihood is parameterized by $\boldsymbol{\tau}$ and $\boldsymbol{\gamma}$ and defined as

$$\mathrm{PL}(\boldsymbol{\tau}, \boldsymbol{\gamma}) = \prod_{i=1}^{n} \{ \frac{\exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})}{\sum_{j \in R_i} \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})} \}^{\Delta_i}, \tag{12}$$

where $\Delta_i = I(T_i \leq C_i)$. C_i is the censoring time for the ith patient, and $I(*)$ is the indicator function.

We use the Breslow estimator [25] to estimate the cumulative baseline hazard $\widehat{H_0}(t) = \int_0^t \widehat{h_0}(u)du$:

$$\widehat{H_0}(t) = \sum_{i=1}^{n} \frac{I(T_i \leq t)\Delta_i}{\sum_{j \in R_i} \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})}. \tag{13}$$

4 Experiments and Analysis

Throughout this section, we randomly pick 70% of the trauma data as training and the rest as testing. Considering the size of our dataset and the training time, we apply 5-fold cross-validation on the trauma training dataset and grid search for hyperparameter choice. Our hyperparameters are described in Table 1. Note that the sequence length and batch size choices are limited by GPU memory.

Table 1: Hyperparameters

Hyperparameters	Survival analysis	Mortality prediction
Batch size	24	16
Sequence length	512	512
Epoch	4	4
Dropout rate	0.1	0.1
Learning rate	1e-2	4e-2
BERT choice	clinical BERT	clinical BERT
Activation	SELU	ReLU

Using the clinical notes and measurements, we formulate the mortality prediction problem as a binary classification problem. Fig. 2 shows the averaged cross validation confusion matrix for mortality prediction in the trauma training dataset. The testing confusion matrix for mortality prediction is presented in Fig. 3. Dominant numbers on the diagonals of both confusion matrices indicate that BERTSurv achieves high accuracy for both of the outcomes (death/discharge).

With BCE loss, we apply two baselines: MLP without notes and the TF-IDF mortality model. In MLP without notes, we consider only the measurements and build a MLP with 3 feed-forward layers for mortality outcomes. In the TF-IDF mortality model, we represent notes with TF-IDF vectors and build a support vector machine (SVM) on TF-IDF vectors combined with measurements for mortality prediction. We use AUC-ROC as our performance metric for mortality prediction, as it is commonly used in survival prediction [26, 27]. AUC-ROC represents the probability that a classifier ranks the risk of a randomly chosen death patient (class 1) higher than a randomly chosen discharged patient (class 0). As is shown in Fig. 5, BERTSurv achieved an AUC-ROC of 0.86, which outperforms MLP without notes by 3.6%. BERTSurv also outperforms MLP without notes, with 5-fold cross validation as shown for our trauma training dataset in Fig. 4.

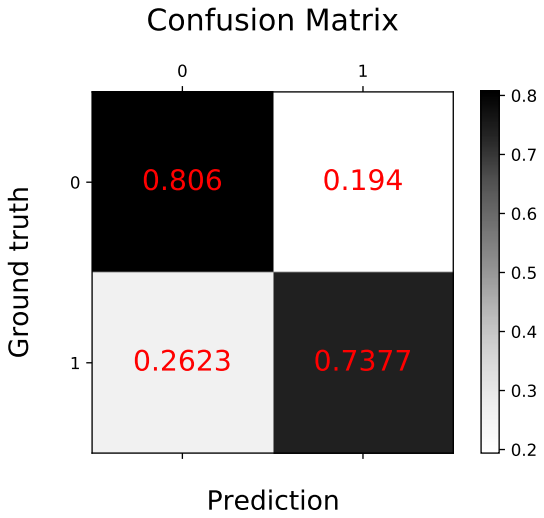


Fig. 2: Averaged confusion matrix for mortality prediction over 5-fold cross validation on our trauma training dataset.

To evaluate the model's predictive performance with PLL loss on survival analysis, we measure the concordance-index (C-index) as outlined by [28]. BERTSurv achieved a C-index of 0.7 on trauma patients, which outperforms a Cox model with a C-index of 0.68. To reduce the risk of ICU trauma patients progressing to an irreversible stage, accurate and early prediction of patient condition is crucial for timely medical decisions. Mortality and cumulative hazard curves for two patients with different outcomes from BERTSurv are shown in Fig. 6 and Fig. 7. Fig. 6(c) indicates that an earlier discharged patients have a lower risk than later discharged patients, while Fig. 6(b) shows that patients who die

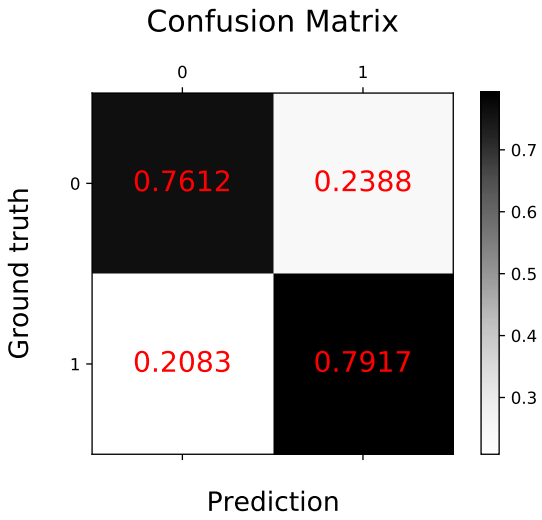


Fig. 3: Confusion matrix for mortality prediction on trauma testing dataset.

early are at a relatively higher risk compared with patients who die later. Comparing Fig. 6(a) and Fig. 6(d), the gap between early discharge vs. early death is larger than that of late discharge vs. late death. Similar conclusions can be drawn from the hazard curves in Fig. 7. Such survival and hazard curves can provide physicians with comprehensive insight into patients' condition change with time.

Fig. 8 depicts four self-attention mechanisms in BERTSurv which help to understand patterns in the clinical notes. In all of the panels, the x-axis represents the query tokens and the y-axis represents the key tokens. In panels (a) and (b), we analyze a clinical note "left apical cap and left lateral pneumothorax suggests severe chest trauma " from a patient that died at hour 76. Panels (a) and (b) are two different head attention mechanisms. Panel (a) indicates "severe chest" and panel (b) extracts "trauma" as prominent patterns, respectively. Similarly, panels (c) and (d) are two head attention mechanisms for a patient discharged at hour 85. The input note to BERTSurv is "the endotracheal tube terminates in good position approximately 4 cm above the carina". BERTSurv finds "good" and "good position" in panels (c) and (d), respectively. Both "severe chest" and "good position" help in understanding the patients' conditions and strongly correlate with the final outcomes. The indications from extracted patterns to patient outcomes show the effectiveness of BERT representation for clinical notes.

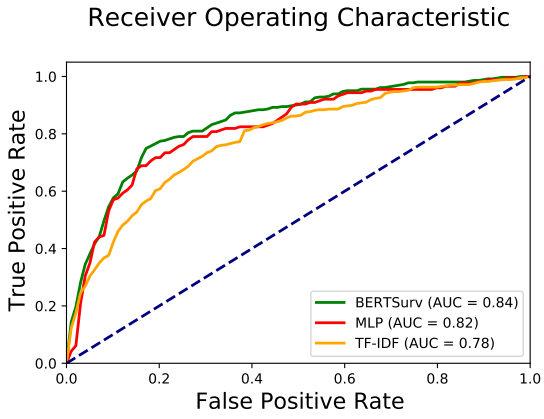


Fig. 4: Receiver operating characteristic (ROC) curve for mortality prediction over 5-fold cross validation on our trauma training dataset. BERTSurv outperforms both baselines.

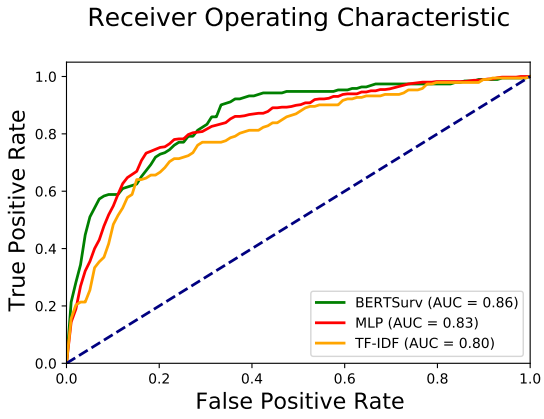
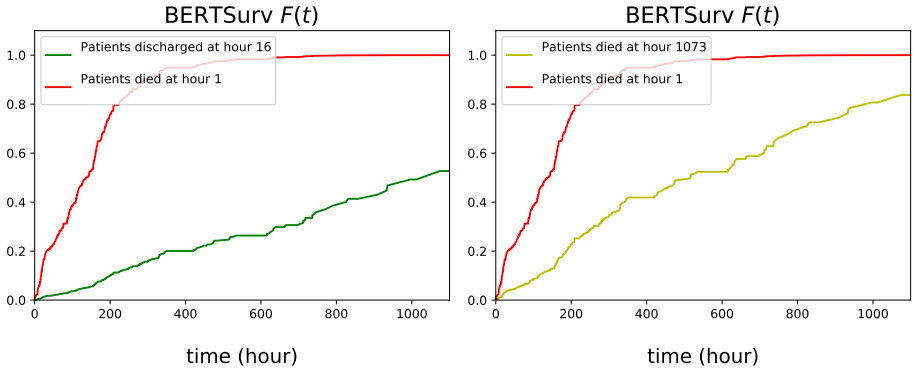


Fig. 5: Receiver operating characteristic (ROC) curve for mortality prediction in trauma testing dataset. BERTSurv outperforms both baselines.



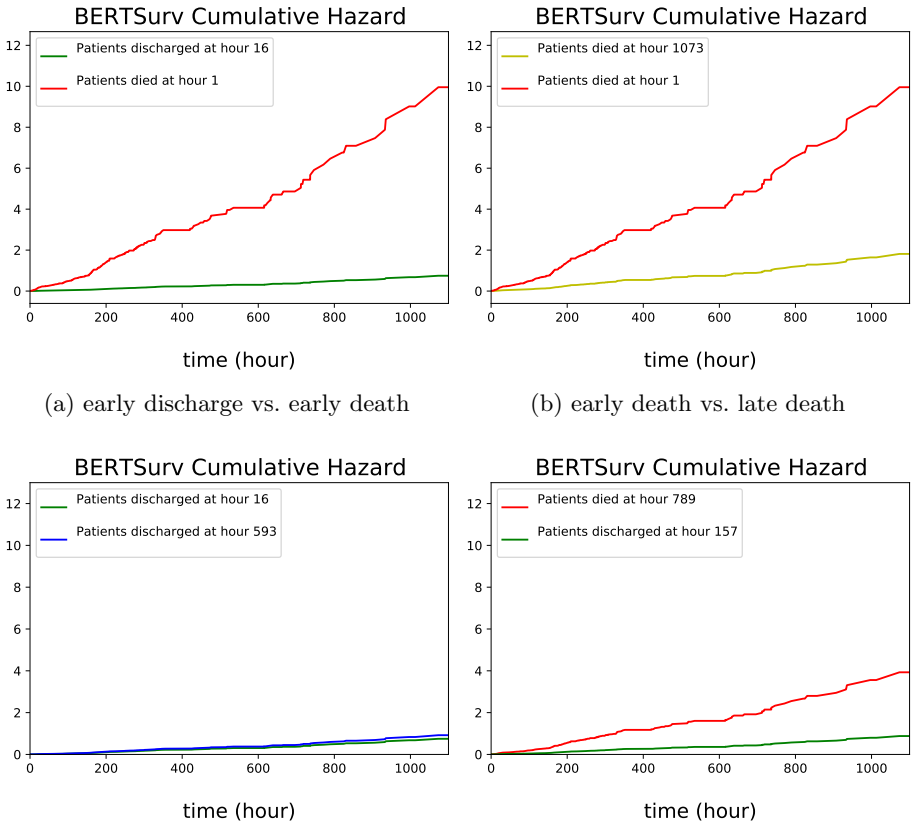
(a) early discharge vs. early death

(b) early death vs. late death

(c) early discharge vs. late discharge

(d) late discharge vs. late death

Fig. 6: Prediction of mortality as a function of time after admission to ICU using BERTSurv.



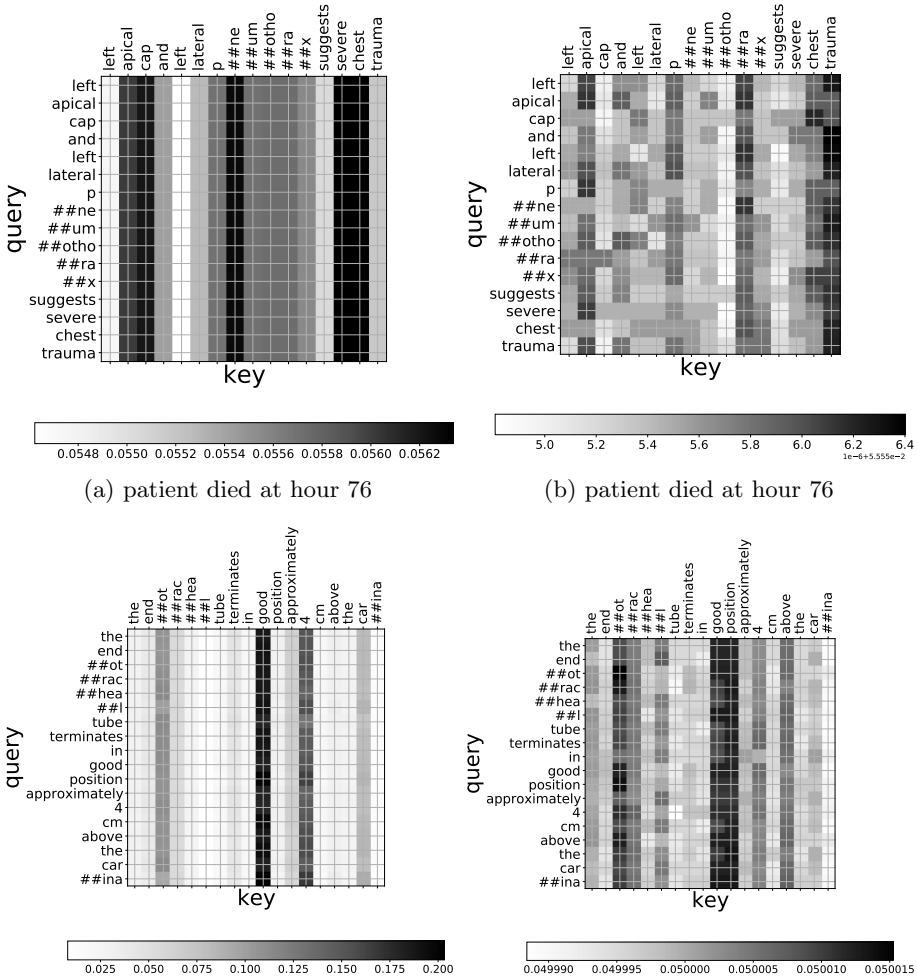
(a) early discharge vs. early death

(b) early death vs. late death

(c) early discharge vs. late discharge

(d) late discharge vs. late death

Fig. 7: Prediction of cumulative hazard function as a function of time after admission to ICU using BERTSurv.



(a) patient died at hour 76

(b) patient died at hour 76

(c) patient discharged at hour 85

(d) patient discharged at hour 85

Fig. 8: BERT visualization. The x-axis are the query tokens and the y-axis are the key tokens. Panels (a) and (b) are two head attention mechanisms for a patient that died at hour 76. The input notes to BERTSurv read "left apical cap and left lateral pneumothorax suggests severe chest trauma". Panels (a) and (b) extract "severe chest" and "trauma" as prominent patterns from the two heads, respectively. "severe chest" and "trauma" provide insight on the patient's critically ill condition. Similarly, panels (b) and (c) are two head attention mechanisms for a patient discharged at hour 85. The input notes include "the endotracheal tube terminates in good position approximately 4 cm above the carina". "good" stands out in panel (c) and "good position" emerges in panel (d). Both "good" and "good position" are strong indications that the patient is in a relatively benign condition.

5 Discussion

We have proposed a deep learning framework based on BERT for survival analysis to include unstructured clinical notes and measurements. Our results, based on MIMIC III trauma patient data, indicate that BERTSurv outperforms the Cox model and two other baselines. We also extracted patterns in the clinical texts with attention mechanism visualization and correlated the assigned weights with survival outcomes. This paper is a proof of principle for the incorporation of clinical notes into survival analysis with deep learning models. Given the current human and financial resources allocated in preliminary clinical note analysis, our method has foreseeable potential to save labor costs, and further improve trauma care. Additional data and work are needed, however, before the extent to which survival analysis can benefit from deep learning and NLP methods can be determined.

6 Acknowledgments

This work was funded by the National Institutes for Health (NIH) grant NIH 7R01HL149670. We acknowledge helpful discussions from Dr. Rachael A. Callcut of the University of California, Davis.

References

1. WISQARS Data Visualization, https://wisqars-viz.cdc.gov:8006/lcd/home.
2. Cullen, D., Sweitzer, B., Bates, D., Burdick, E., Edmondson, A., Leape, L.: Preventable adverse drug events in hospitalized patients. Critical Care Medicine. 25, 1289-1297 (1997).
3. Zhang, Y., Jiang, R., Petzold, L.: Survival topic models for predicting outcomes for trauma patients. In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 1497-1504 (2017).
4. Cox, D.: Regression models and life-tables. Journal of the Royal Statistical Society: Series B (Methodological). 34(2), 187-202 (1972).
5. Faraggi, D., Simon, R.: A neural network model for survival data. Statistics in Medicine. 14, 73-82 (1995).
6. Katzman, J., Shaham, U., Cloninger, A., Bates, J., Jiang, T., Kluger, Y.: DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC Medical Research Methodology. 18 (2018).
7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention is all you need. In Advances in Neural Information Processing Systems, 5998-6008 (2017).
8. Devlin, J., Chang, M., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
9. Darabi, H., Tsinis, D., Zecchini, K., Whitcomb, W., Liss, A.: Forecasting mortality risk for patients admitted to intensive care units using machine learning. Procedia Computer Science, 140, 306-313 (2018).

10. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C., Kang, J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4), 1234-1240 (2020).
11. Rasmy, L., Xiang, Y., Xie, Z., Tao, C., Zhi, D.: Med-BERT: pre-trained contextualized embeddings on large-scale structured electronic health records for disease prediction. arXiv preprint arXiv:2005.12833 (2020).
12. Shang, J., Ma, T., Xiao, C., Sun, J.: Pre-training of graph augmented transformers for medication recommendation. arXiv preprint arXiv:1906.00346 (2019).
13. Alsentzer, E., Murphy, J., Boag, W., Weng, W., Jin, D., Naumann, T., McDermott, M.: Publicly available clinical BERT embeddings. arXiv preprint arXiv:1904.03323. (2019).
14. Huang, K., Altosaar, J., Ranganath, R.: Clinicalbert: Modeling clinical notes and predicting hospital readmission. arXiv preprint arXiv:1904.05342 (2019).
15. Wa, K., Wagner, D., Draper, E.: The APACHE III prognostic system. Risk prediction of hospital mortality for critically ill hospitalized adults. Chest, 100(6) 1619-1636 (1991).
16. Johnson, A., Pollard, T., Shen, L., et al.: MIMIC-III, a freely accessible critical care database. Scientific data, 3(1), 1-9 (2016).
17. Buuren, S., Groothuis-Oudshoorn, K.: mice: Multivariate imputation by chained equations in R. Journal of statistical software, 1-68. Chicago (2010).
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015).
19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958 (2014).
20. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
21. Paszke, A., Gross, S., Massa, F., et al.: Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems, 8026-8037 (2019).
22. Wu, Y., Schuster, M., Chen, Z., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016).
23. Klein, J., Moeschberger, M.: Survival analysis: techniques for censored and truncated data. Springer Science & Business Media (2006).
24. Kalbfleisch, J., Prentice, R.: The statistical analysis of failure time data. Vol. 360. John Wiley & Sons (2011).
25. Prentice, R., Breslow, N.: Retrospective studies and failure time models. Biometrika, 65(1), 153-158 (1978).
26. Hung, H., Chiang, C.: Estimation methods for time-dependent AUC models with survival data. Canadian Journal of Statistics, 38(1), 8-26 (2010).
27. Chambless, L., Cummiskey, C., Cui, G.: Several methods to assess improvement in risk prediction models: extension to survival analysis. Statistics in Medicine, 30(1), 22-38 (2011)..
28. Harrell Jr, F., Lee, K., Califf, R., Pryor, D., Rosati, R.: Regression modelling strategies for improved prognostic prediction. Statistics in Medicine, 3(2), 143-152 (1984).

Method of Computing Similarity with Clusters

Arthur Yosef[1] and Moti Schneider[2]

[1]Tel Aviv-Yaffo Academic College, Israel, ✉ yusupoa@yahoo.com
[2]Netanya Academic College, Israel, ✉ profmoti@gmail.com

Abstract. In this study, we explore the possibility of computing similarity between numerical vectors by utilizing their division into clusters. The advantage of such procedure is apparent mostly in the cases where the data is very unreliable and distorted, so that the cluster represents approximate value of its elements in a very broad form. Measuring similarity between numerical vectors following their division into clusters, provides additional method for similarity measurement, which might be a preferable method when our lack of confidence in the measurements of individual elements is excessive.

Keywords: Data Mining, Soft Computing, fuzzy logic, similarity measure, clusters.

1 Introduction

When computing similarity between two numerical vectors, the general practice is to perform a measurement of distance between the corresponding elements of these vectors. However, when the data is characterized by severely unreliable and distorted measurements, taking the measurements of individual data elements, means that we are making an implicit assumption that the measurements are either precise, or close enough to being precise so that the results are still expected to be reliable. This is often self-illusion in the cases of severe distortion (or intentional dis-information). In most cases of problematic data, the distortions are nevertheless limited for most of the data elements, such that when domain expert goes over the numbers, vast majority of them seem to be in the right order of magnitude. In cases like that, it is possible to approach the computation of similarity between two numerical vectors by dividing the numerical vectors into clusters and treating the whole cluster, where a given data element is located as an approximate measure representing its value. Obviously, such approach takes into account our lack of confidence in the measurements of the individual data elements. Using the values of the whole cluster as a substitute for individual elements contained in it, allows more reasonable evaluation of similarity of vectors. The broader approach masks imprecision and data distortion as long as they are not unreasonable. However, when interpreting the results, the broad nature of treating the measurements must be kept in mind.

There are many different approaches to perform clustering. The literature is very extensive, and only a small sample is presented here. There are algorithms such as K-means [4] and Clustering Large Applications based on Randomized Search [5], which are based on a partitioning approach; Gaussian mixture models ([6],[7]) are associated with a model-based approach; Divisive Analysis [8] and Balanced Iterative Reducing

and Clustering using Hierarchies [9] are based on a hierarchical approach; Statistical Information Grid [14] and Clustering in Quest [10] are based on a grid-based approach. Density-Based Spatial Clustering of Applications with Noise [11] and Ordering Points to Identify the Clustering Structure [12] are examples of a density-based approach. Density-based clustering creates clusters of arbitrary shape, is robust to noise, and does not require prior knowledge regarding the number of clusters [13].

2. K-means Clustering [1], [2], [3]

The first step in performing cluster-based measure of similarity is to divide both numerical vectors into clusters. In this study we apply K-means clustering method, which is a well-known and widely used method. K-Means clustering method is a very basic and simple. There are 2 variations to the K-means approach: (i) Fixed size clustering method (ii) Varied size clustering.

(i) Fixed size clustering method requires that the user decides in advance regarding the amount of clusters in the numerical vector. The formal description of the algorithm is as follows:

Algorithm 1: Construction of a fixed size cluster set

Input: $A = (a_1, a_2, \dots, a_n)$
Output: K clusters of A

1. Select K elements as the initial means
2. Repeat
    a. Form K clusters by assigning all elements to the closest mean
    b. Recompute the mean of each cluster

Until the means don't change

(ii) The varied size clustering differs from the fixed size methods in one important issue: there is no need for the user to determine in advance the amount of clusters for a numerical vector. However, there is a requirement to determine in advance a maximum threshold distance between the value of the element and the value of the center (the mean). It essentially means that all the data elements having a distance below the given threshold from the center of the cluster – are similar to each other. The formal description of the algorithm is as follows:

Let r_{max} be the maximum threshold distance between 2 points such that they are considered similar

Algorithm 2: Construction of a varied size cluster set.

Input: $A = (a_1, a_2, \dots, a_n)$, r_{max}- the maximum threshold distance (radius) between 2 points

Output: The clusters of A (denoted by $\{C_j^A\}$)

1. Create the first cluster with the first element as its mean and $k = 1$
2. For $i \leftarrow 1$ to n do
   a. Find a cluster j ($j \in \{1, \dots, k\}$) such that $dist_j(a_i)$ is minimum where $dist_m(a_i) = |mean_m - a_i|$ and $mean_m$ is a mean of cluster m, for $m \in \{1, \dots, k\}$
      (i.e., find a cluster j such that $|mean_j - a_i| = min_{m=1}^{k} |mean_m - a_i|$).
   b. If $dist_j(a_i) \leq r_{max}$ then add a_i to cluster j
      Else, create a new cluster ($k \leftarrow k + 1$) and set its mean to be the value of a_i.
3. Recompute all the cluster means and repeat from stage 2, until the cluster means don't change

In this paper, we use four numerical vectors in our case study. The data was downloaded from the Data Base of the World Bank. The following variables (numerical vectors) were downloaded:

- GDP per capita (denoted by GDP)
- High Tech Exports per capita (denoted by High-Tech)
- Secondary Education Enrollment (denoted by Secondary)
- Birth Rate

Each of the numerical vectors consists of cross-national data (for the year 1985), by country. The data in all the numerical vectors used in this study were normalized, and hence brought to the same scale. All the values in the numerical vectors are between 0 and 1, which allows us to use the same maximum threshold distances for construction the clusters in all four data series.

Table 1: Possibilities of dividing the numerical vectors into clusters

		Number of clusters			
		GDP	High-Tech	Secondary	Birth Rate
r_{max}	0.10	8	7	8	9
	0.15	6	5	6	6
	0.20	5	5	5	4
	0.25	5	4	4	4
	0.30	4	4	4	4

Table 1 displays the results of dividing the numerical vectors into clusters, using different distance thresholds, ranging from 0.1 to 0.3. It can be observed, that as maximum threshold increases, the amount of clusters decreases. In such cases, domain experts will have to make a decision, what maximum threshold (and the resulting amount) of clusters is the most appropriate for a specific model under consideration.

3. Using clusters to compute similarity

The first step in computing similarity between two numerical vectors is constructing the clusters. The general idea of similarity by clusters is as follows: we start with a given element from one of the data series, and measure whether that element could belong to the cluster containing the parallel data element from the other data series. If it could belong to that cluster, then we assign it a score of 1 (the highest score, since the two elements can be associated with the same cluster). If on the other hand, the element could not belong to the relevant cluster, then score is less than 1, depending on the algorithm used.

The algorithm 3 presented below actually represents 3 different models to compute score. In other words, all the components of the algorithm are the same for all three different models, except the component that computes the scores.

Algorithm 3: Three models (Boolean, Exponential and Linear)

Input: $A = (a_1, a_2, \dots, a_n)$, $B = (b_1, b_2, \dots, b_n)$, r_{max}- the maximum distance threshold between 2 points

Output: $S(A, B)$

1. Based on Algorithm 2, divide A into clusters (denoted by $\{C_j^A\}_{j=1}^{l}$)
2. $Sum_A \leftarrow 0$
3. For $i = 1$ to n do
   a. Suppose that a_i belongs to the cluster C_j^A. Find a cluster k ($k \in \{1, \dots, l\}$) such that $dist_k(b_i)$ is minimum where $dist_j(b_i) = |mean_j^A - b_i|$ and $mean_j^A$ is a mean of cluster C_j^A
   b. $Sum_A \leftarrow Sum_A + \varphi(k, j)$

Similarly, repeat steps 1-3 with reverse between A and B:

4. Based on Algorithm 2, divide B into clusters (denoted by $\{C_j^B\}_{j=1}^{r}$)
5. $Sum_B \leftarrow 0$
6. For $i = 1$ to n do
   a. Suppose that b_i belongs to the cluster C_j^B. Find a cluster k ($k \in \{1, \dots, r\}$) such that $dist_k(a_i)$ is minimum where $dist_m(a_i) = |mean_m^B - a_i|$ and $mean_m^B$ is a mean of cluster C_m^B
   b. $Sum_B \leftarrow Sum_B + \varphi(k, j)$
7. $S(A, B) \leftarrow \frac{1}{2n}(Sum_A + Sum_B)$

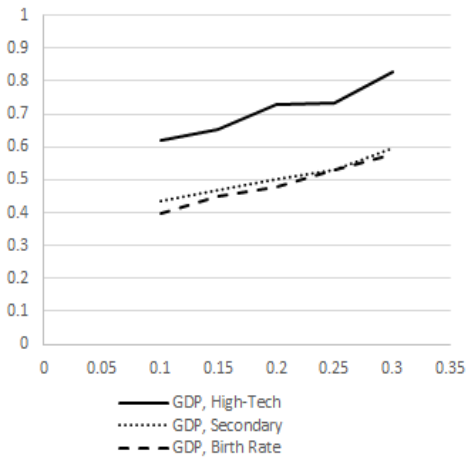
Boolean score function:
$$\varphi(k, j) = \begin{cases} 1 & , k = j \\ 0 & , k \neq j \end{cases} \quad (1)$$

Exponential score function:
$$\varphi(k, j) = \begin{cases} 1 & , k = j \\ 2^{-|j-k|} & , k \neq j \end{cases} \quad (2)$$

Linear score function:
$$\varphi(k, j) = \begin{cases} 1 - 0.25|k - j| & , |k - j| = 0,1,2,3 \\ 0 & , else \end{cases} \quad (3)$$

Note: If $k = j$ then $\varphi(k, j) = 1$. Else, $0 \leq \varphi(k, j) < 1$.

In other words, in Boolean model, if the distance is not 0, the score function (in short, score) is zero ($\varphi = 0$). In two other models, we are basically assigning count number for each cluster in both data series. Then we compute score based on the difference in count numbers for the clusters containing the equivalent data elements

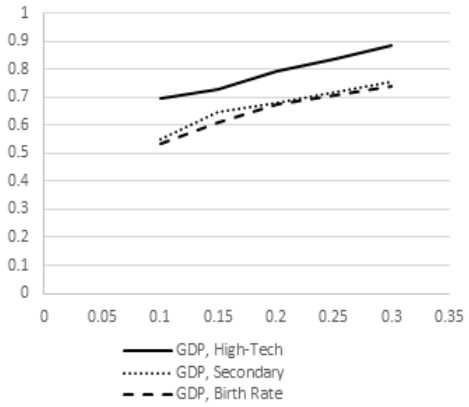


Graph 1: Results of similarity measures using Boolean Model

Table 2: Results of similarity measures using Boolean Model

A	B	r_{max}	S_{BOOL}
GDP	High-Tech	0.10	0.620
		0.15	0.655
		0.20	0.730
		0.25	0.735
		0.30	0.830
	Secondary	0.10	0.435
		0.15	0.467
		0.20	0.500
		0.25	0.528
		0.30	0.594
	Birth Rate	0.10	0.399
		0.15	0.449
		0.20	0.479
		0.25	0.530
		0.30	0.576

Graph 1 and Table 2 above display the results of our example in the case of a Boolean model. As could be expected – when the maximum distance threshold increases, the similarity increases as well. Obviously, when the amount of clusters decreases, the size of the clusters increases, thus increasing the possibility that two parallel data elements will become part of two parallel clusters (clusters having the same count). The results of the Boolean model are lower in comparison to the other two models due to the distortions implied by its Boolean nature: when the element could not belong to the relevant cluster, then no matter how far it is from the closest edge of that cluster, the score is always 0.

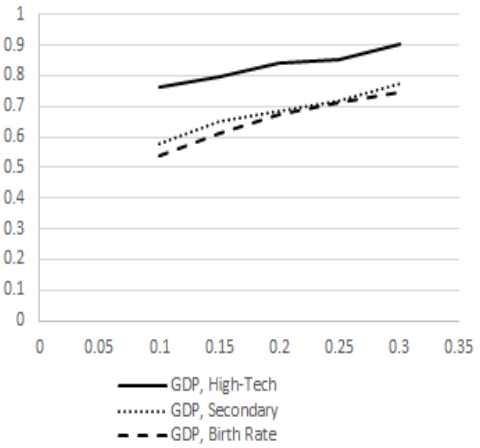


Graph 2: Results of similarity measures using Exponential Model

Table 3: Results of similarity measures using Exponential Model

A	B	r_{max}	S_{EXP}
GDP	High-Tech	0.10	0.698
		0.15	0.731
		0.20	0.792
		0.25	0.839
		0.30	0.883
	Secondary	0.10	0.550
		0.15	0.647
		0.20	0.680
		0.25	0.716
		0.30	0.757
	Birth Rate	0.10	0.532
		0.15	0.610
		0.20	0.673
		0.25	0.707
		0.30	0.740

Graph 2 and Table 3 above display results of our example in the case of Exponential Model. The idea behind determining the score in the exponential model is as follows: when an element of the first numerical vector could belong to the cluster containing the parallel data element of the second numerical vector, then the score is 1. Else, if the parallel element is not a member of the cluster having the same count number, but is a member of the neighboring cluster, the score is 0.5. If the parallel element is contained in the further cluster, the score is 0.25, if it is even further, the score is 0.125 and so on (see equation (2)).



Graph 3: Results of similarity measures using linear model

Table 4: Results of similarity measures using Linear Model

A	B	r_{max}	S_{LIN}
GDP	High-Tech	0.10	0.763
		0.15	0.795
		0.20	0.840
		0.25	0.852
		0.30	0.905
	Secondary	0.10	0.579
		0.15	0.648
		0.20	0.684
		0.25	0.717
		0.30	0.773
	Birth Rate	0.10	0.537
		0.15	0.613
		0.20	0.673
		0.25	0.714
		0.30	0.745

Models 2 and 3 are based on the very similar concepts. When parallel data elements are contained in the corresponding clusters, in both models thee score is 1. However, when they are not located in the corresponding clusters, the score is lower than 1. As data elements are contained in clusters which are further apart (according to their count number), the exponential model assigns greater penalty, which means lower partial score that very quickly approaches 0. In contrast, the linear model is more moderate, offers higher partial scores, which decline towards 0 more gradually (0.75, 0.5, 0.25 – see equation (3)).

4. Summary and conclusions

Table 5 (below) summarizes all the similarity results generated by the 3 models. It is easy to see that the Boolean model generated the lowest results. The reason for lower results is due to the distortions which are a direct result of a Boolean approach, as was explained above.

Table 5: Summary of similarity results

A	B	r_{max}	S_{BOOL}	S_{EXP}	S_{LIN}
GDP	High-Tech	0.10	0.620	0.698	0.763
		0.15	0.655	0.731	0.795
		0.20	0.730	0.792	0.840
		0.25	0.735	0.839	0.852
		0.30	0.830	0.883	0.905
	Secondary	0.10	0.435	0.550	0.579
		0.15	0.467	0.647	0.648
		0.20	0.500	0.680	0.684
		0.25	0.528	0.716	0.717
		0.30	0.594	0.757	0.773
	Birth Rate	0.10	0.399	0.532	0.537
		0.15	0.449	0.610	0.613
		0.20	0.479	0.673	0.673
		0.25	0.530	0.707	0.714
		0.30	0.576	0.740	0.745

We can also observe that consistently, linear model generates higher results than the exponential model. The reason for that is (as was explained above) the more restrictive nature of the exponential model. Nevertheless, both (Exponential and Linear) models generate similar results. It is up to the user to decide which one of the two models is more appropriate for their purposes: either more restrictive score computation, or more generous one.

The process of computing similarities between two numerical vectors is consistent and involves the following steps:

1. The data of all numerical vectors used in this study was normalized and brought to the same scale [0.1].
2. We utilized K-means Clustering method based on the idea that clusters could be of different (varied) magnitude (Algorithm 2).

3. Three different models of similarity by clusters were presented.
4. We utilized four data series (numerical vectors) to demonstrate the application of the 3 models.

The study demonstrated the possibility to compute similarity between numerical vectors by clusters. In other words, when there is a strong suspicion, that the data series are severely distorted and unreliable, we can view each cluster as broadly representing the values of all its data elements.

References

1. H.-P. Kriegel, P. Kröger, J. Sander and A. Zimek, "Density-based Clustering," *WIREs Data Mining and Knowledge Discovery,* p. 1 (3): 231–240., 2011.
2. R. Ng and J. Han, "Efficient and effective clustering method for spatial data mining," in *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.
3. T. Zhang, R. Ramakrishnan and M. Livny, "An Efficient Data Clustering Method for Very Large Databases.," in *Proc. Int'l Conf. on Management of Data*.
4. J. A. Hartigan and M. A. Wong, "Algorithm as 136: A K-means clustering algorithm," *J. of the Royal Stat. Soc. Ser. C (Applied Statistics),* pp. vol. 28, no. 1, pp. 100–108, 1979.
5. R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *," IEEE Trans. on Knowl. and Data Eng.,* pp. vol.14, no. 5, pp. 1003–1016, 2002.
6. C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *J. of the Am. Stat. Assoc.,* pp. vol. 97,no. 458, pp. 611–631, 2002.
7. D. H. Fisher, "Improving inference through conceptual clustering," in *Proc. 6th Nat. Conf. Artificial Intell. (AAAI-87)*, Seattle, WA, 1987.
8. L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.
9. T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD'96)*, Montreal, Canada, 1996.
10. R. Agrawal, J. E. Gehrke, D. Gunopulos and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD Int. Conf. Manage.Data (SIGMOD'98)*, Seattle, WA, 1998.
11. M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Data Min. and Knowl. Discov.,* pp. vol. 96, no. 34, pp. 226–231, 1996.
12. M. Ankerst, M. M. Breunig, H. P. Kriege and J. Sande, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD'99)*, Philadelphia, PA, 1999.
13. H. P. Kriegel, P. Kroger, J. Sander and A. Zimek, "Density-based clustering," *WIREs: Data Min. and Knowl. Discov.,* pp. vol. 1, no. 3, pp.231–240, 2011.
14. W. Wang, J. Yang and R. R. Muntz, "STING: A statistical information grid approach to spatial data mining," *Proc. 23rdConf. Very Large Data Bases (VLDB)*, Athens, Greece1997.

Empirical Analysis of Machine Learning Configurations for Prediction of Multiple Organ Failure in Trauma Patients

Yuqing Wang[1⋆], Yun Zhao[1⋆], Rachael Callcut[2], and Linda Petzold[1]

[1] Department of Computer Science, University of California, Santa Barbara
[2] UC, Davis Health
wang603@ucsb.edu, yunzhao@cs.ucsb.edu

Abstract. Multiple organ failure (MOF) is a life-threatening condition. Due to its urgency and high mortality rate, early detection is critical for clinicians to provide appropriate treatment. In this paper, we perform quantitative analysis on early MOF prediction with comprehensive machine learning (ML) configurations, including data preprocessing (missing value treatment, label balancing, feature scaling), feature selection, classifier choice, and hyperparameter tuning. Results show that classifier choice impacts both the performance improvement and variation most among all the configurations. In general, complex classifiers including ensemble methods can provide better performance than simple classifiers. However, blindly pursuing complex classifiers is unwise as it also brings the risk of greater performance variation.

Keywords: Machine learning · Multiple organ failure · Trauma.

1 Introduction

Multiple organ failure (MOF) is a clinical syndrome with variable causes including pathogens [1], complicated pathogenesis [2], and a major cause of mortality and morbidity for trauma patients who are admitted to Intensive Care Units (ICU) [3]. Based on recent studies on ICU trauma patients, up to 47% have developed MOF, and MOF increased the overall risk of death 6 times compared to patients without MOF [4]. To prevent the development of MOF for trauma patients from progression to an irreversible stage, it is essential to diagnose MOF early and effectively. Many scoring systems have been proposed to predict MOF [5–8] and researchers have attempted to predict MOF on trauma patients using predictive models in an early phase [9, 10].

The rapid growth of data availability in clinical medicine requires doctors to handle extensive amounts of data. As medical technologies become more complicated, technological advances like machine learning (ML) are increasingly needed to improve real-time analysis and interpretation of the results [11]. In recent years, practical uses of ML in healthcare have grown tremendously, including

⋆ These two authors contributed equally to this paper.

cancer diagnosis and prediction [12–14], tumor detection [15, 16], medical image analysis [17], and health monitoring [18, 19].

Compared to traditional medical care, ML-assisted clinical decision support enables a more standardized process for interpreting complex multi-modality data. In the long term, ML can provide an objective viewpoint for clinical practitioners to improve performance and efficiency [20]. ML is often referred to as a black box: explicit input data and output decisions, but opaque at intermediate learning process. Additionally, in medical domains, there is no universal rule for selecting the best configuration to achieve the optimal outcome. Moreover, medical data has its own challenges such as numerous missing values [21] and colinear variables [22]. Thus it is difficult to process the data and choose the proper model and corresponding parameters, even for a ML expert. Furthermore, detailed quantitative analysis of the potential impacts of different settings of ML systems on MOF has been missing.

In this paper, we experiment with comprehensive ML settings for prediction of MOF, considering 6 different dimensions from data preprocessing (missing value treatment, label balancing, feature scaling), feature selection, classifier choice, to hyperparameter tuning. To predict MOF for trauma patients at an early stage, we use only initial time measurements (hour 0) as inputs. We mainly use area under the receiver operating characteristic curve (AUC) to evaluate MOF prediction outcomes. We focus on analyzing the relationships among configuration complexity, predicted performance, and performance variation. Additionally, we quantify the relative impacts of each dimension.

The main contributions of this paper include:

(1) To the best of our knowledge, this is the first paper to conduct a thorough empirical analysis quantifying the predictive performance with exhaustive ML configurations for MOF prediction.
(2) We provide general guidance for ML practitioners in healthcare and medical fields through quantitative analysis of different dimensions commonly used in ML tasks.
(3) Experimental results indicate that classifier choice contributes most to both performance improvement and variation. Complex classifiers including ensemble methods bring higher default/optimized performance, along with a higher risk of inferior performance compared to simple ones on average.

The remainder of this paper is organized as follows. Section 2 describes the dataset and features we use. All of the ML configurations are available in Section 3. Experimental results are discussed in Section 4. Finally, our conclusions are presented in Section 5.

2 Dataset

Our dataset, collected from the San Francisco General Hospital and Trauma Center, contains 2190 highest level trauma activation patients evaluated at the

level I trauma center. Due to the urgency of medical treatment, there are numerous missing values for time-dependent measurements. Thus we have chosen to consider only those features with a maximum missing value percentage of 30% over all patients. To obtain a timely prediction, early lab measurements (hour 0) as well as patients' demographic and illness information were extracted as the set of features. Detailed feature statistics are available in Table 1.

Feature type	# of extracted features	Features
Demographic	5	*gender*, age, weight, *race, blood type*
Illness	2	*comorbidities, drug usage*
Injury factors	4	*blunt/penetrating trauma*, # of rib fractures, *orthopedic injury*, *traumatic brain injury*
Injury scores	8	injury severity score, 6 abbreviated injury scale (head, face, chest, abdomen, extremity, skin), Glasgow coma scale score
Vital sign measurements	4	heart rate, respiratory rate, systolic blood pressure, mean arterial pressure
Blood-related measurements	13	white blood cell count, hemoglobin, hematocrit, serum CO_2, prothrombin time, international normalized ratio, partial thromboplastin time, blood urine nitrogen, creatinine, blood pH, platelets, base deficit, *factor VII*

Table 1: MOF dataset statistics. Italicized features are categorical.

Our target variable consists of binary class labels (0 for no MOF and 1 for MOF). Then, the data with feature and target variables is randomly split into training and testing sets at the ratio of $7:3$.

3 Methods

Based on ML pipelines and special characteristics of our data such as large number of missing values and varying scales in feature values, we consider comprehensive ML configurations from the following 6 dimensions: data preprocessing (missing value treatment (MV), label balancing (LB), feature scaling (SCALE)), feature selection (FS), classifier choice (CC), and hyperparameter tuning (HT). In the remainder of the paper, we will interchangeably use the full name and corresponding abbreviations shown in parentheses. Further details on each dimension are described below.

3.1 Data Preprocessing

Methods to handle the dataset with missing values, imbalanced labels, and unscaled variables are essential for the data preprocessing process. We use several different methods to deal with each of these problems.

Missing Value Treatment In our dataset, numerous time-dependent features cannot be recorded on a timely basis, and missing data is a serious issue. We consider three different ways to deal with missing values, where the first method serves as the baseline setting for MV, and the latter two methods are common techniques of missing value imputation in ML.

1. Remove all patients with any missing values for the features listed in Section 2.
2. Replace missing values with mean for numerical features and mode for categorical features over all patients.
3. Impute missing values by finding the k-nearest neighbors with the Euclidean distance metric for each feature respectively.

Label Balancing Our dataset is imbalanced as the sample class ratio between class 0 and class 1 is $11:1$. Keeping imbalanced class labels serves as the baseline setting for LB. Three different ways are considered to resample the training set.

1. Oversampling the minority class (label 1)
   1.1 Method: SMOTE (synthetic minority over-sampling technique) [23].
   1.2 Explanation: choose k-nearest neighbors for every minority sample and then create new samples halfway between the original sample and its neighbors.
2. Undersampling the majority class (label 0)
   2.1 Method: NearMiss [24].
   2.2 Explanation: when samples of both classes are close to each other, remove the samples of the majority class to provide more space for both classes.
3. Combination of oversampling and undersampling
   3.1 Method: SMOTE & Tomek link [25].
   3.2 Tomek link: two samples are k-nearest neighbors to each other but come from different classes.
   3.3 Explanation: first create new samples for the minority class and then remove the majority class sample in any Tomek link.

Feature Scaling Since the range of feature values in our dataset varies widely, we perform feature scaling. No scaling on any feature serves as the baseline setting for SCALE. Two common scaling techniques are used for numerical features.

1. Normalization: rescale values to range between 0 and 1.
2. Standardization: rescale values with mean 0 and standard deviation 1.

3.2 Feature Selection

In medical datasets, there usually exist many highly correlated features, and some features that are weakly correlated to the target [22, 26]. Thus it is essential to identify the most relevant features that may help to improve the outcome of the analysis. Using all of the features described in Section 2 serves as the baseline setting for FS. We consider two main feature selection techniques: filter and wrapper methods.

1. Filter-based methods (independent of classifiers):
   1.1 Use correlation between features and the target to select features which are highly dependent on the target.
   1.2 Filter out numerical features using ANOVA F-test and categorical features using χ^2 test.
2. Wrapper-based methods (dependent on classifiers):
   2.1 Method: RFE (recursive feature elimination) in random forest.
   2.2 Explanation: perform RFE repeatedly such that features are ranked by importance, and the least important features are disregarded until a specific number of features remains.

3.3 Classifier Choice

We experimented with 15 classifiers on the dataset. In general, these classifiers can be divided into two main categories: single and ensemble. Lists of all classifiers are available in Table 2. For ensemble classifiers (combination of individual classifiers), we tried bagging (BAG, RF, ET), boosting (GB, ABC, XGB, LGBM), voting (VOTE) and stacking (STACK). In bagging, DT is a homogeneous weak learner. Multiple DTs learn the dataset independently from each other in parallel and the final outcome is obtained by averaging the results of each DT. In boosting, DT also serves as a homogeneous weak learner. However, DTs learn the dataset sequentially in an adaptive way (new learner depends on previous learners' success), and the final outcome is determined by weighted sum of previous learners. In voting, heterogeneous base estimators (LR, RF, SVM, MLP, ET) are considered, where each estimator learns the original dataset and the final prediction is determined by majority voting. In stacking, several heterogeneous base learners (RF, KNN, SVM) learn the dataset in parallel, and there exists a meta learner (LR) that combines the predictions of the weak learners. Abbreviations of classifiers shown in parentheses for voting and stacking are the ones we use.

Single classifiers	Ensemble classifiers
Logistic Regression (LR) Support Vector Machine (SVM) Naive Bayes (NB) K-nearest Neighbors (KNN) Decision Tree (DT) Multi-layer Perceptron (MLP)	Bagged Trees (BAG) Random Forest (RF) Extra Trees (ET) Gradient Boosting (GB) Adaptive Boosting (ABC) Extreme Gradient Boosting (XGB) Light Gradient Boosting Machine (LGBM) Voting (VOTE) Stacking (STACK)

Table 2: List of 6 single classifiers and 9 ensemble classifiers. Corresponding abbreviations of each classifier are shown in parentheses.

3.4 Hyperparameter Tuning

Hyperparameters are crucial for controlling the overall behavior of classifiers. Default hyperparameters of classifiers serve as the baseline setting for HT. We apply grid search to perform hyperparameter tuning for all classifiers. Detailed information about tuned hyperparameters is available in Table 3.

Classifiers	♯ of tuned hyperparameters	Hyperparameter lists
LR	3	C, class_weight, penalty
SVM	4	C, gamma, kernel, class_weight
KNN	3	n_neighbors, weights, algorithm
NB	1	var_smoothing
DT	5	min_samples_split, max_depth, min_samples, leaf_max_features, class_weight
MLP	3	activation, solver, alpha
BAG	2	base_estimator, n_estimators
RF	2	n_estimators, max_features
ET	2	n_estimators, max_features
GB	2	n_estimators, max_depth
ABC	3	base_estimator, n_estimators, learning_rate
XGB	2	min_child_weight, max_depth
LGBM	4	num_leaves, colsample_bytree, subsample, max_depth
VOTE	2	C (SVM), n_estimators (ET)
STACK	2	C (SVM), n_neighbors (KNN)

Table 3: Detailed configurations of tuned hyperparameters for all classifiers. All of the hyperparameter names come from *scikit-learn* [27].

4 Experiments and Results

We formulated MOF prediction as a binary classification task. All of the experiments in this paper were implemented using *scikit-learn* [27]. As mentioned in Section 2, our training and testing dataset is randomly split with a ratio of 7 : 3. One-hot encoding is applied to all categorical features. For each classifier, we use the same training and testing dataset. We use AUC as our main performance metric, as it is commonly used for MOF prediction in the literature [6, 28, 29]. It provides a "summary" of classifier performance compared to single metrics such as precision and recall. AUC represents the probability that a classifier ranks a randomly chosen positive sample (class 1) higher than a randomly chosen negative sample (class 0), and thus useful for imbalanced datasets. In this section, we quantify the impacts (improvement and variation) of each dimension on the predicted performance over our testing dataset.

4.1 Influence of Individual Dimensions

First, we evaluate how much each dimension contributes to the AUC score improvement and variation respectively, and find the correlation between performance improvement and variation over all dimensions.

Performance Improvement across Dimensions For HT, MV, LB, SCALE, and FS, we define the ***baseline*** as default hyperparameter choices, using no missing value imputation, no label balancing, no feature scaling, and no feature selection, respectively. For CC, we choose SVM, which achieves the median score among all classifiers, as the ***baseline***. Then we quantify the performance improvement of each dimension. Fig. 1 shows the percentage that each dimension contributes to the improvement in the AUC score over baseline by tuning only one dimension at a time while leaving others at baseline settings. We observe that CC contributes most to the performance improvement (15.00%) for MOF prediction. After CC, LB (10.81%), FS (10.09%), MV (7.90%), HT (6.94%), and FS (2.45%) bring decreasing degrees of performance improvement in the AUC score.

Table 4 shows the improvement of every single dimension on each classifier over the baseline. In general, MV and LB tend to provide the greatest performance improvement for most classifiers. For RF, ET, and LGBM, FS contributes the most to improvement in performance since these classifiers require feature importance ranking intrinsically, and external FS improves their prediction outcomes to a large extent. Note that the classifier for which SCALE has the largest impact is KNN, as it is a distance-based classifier which is sensitive to the range of feature values. Also, due to instability and tendency to overfit, HT is the most critical for DT improvement.

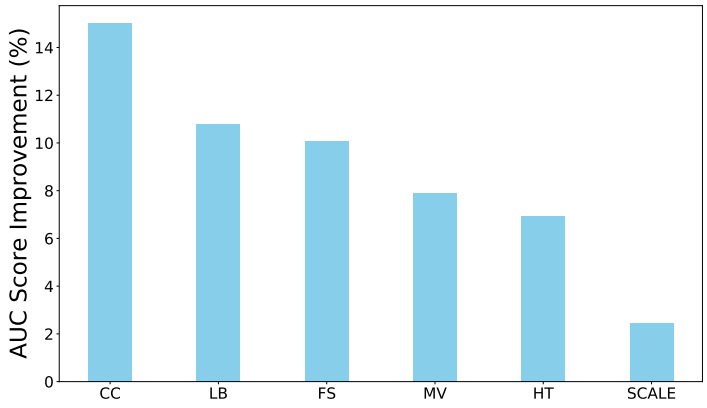


Fig. 1: Performance improvement in the AUC score of each dimension over the baseline when tuning only one dimension at a time while leaving others at baseline settings. CC brings the greatest performance improvement, followed by LB, FS, MV, HT, and SCALE in decreasing order of improvement.

Classifier	MV (%)	LB (%)	SCALE (%)	FS (%)	HT (%)
LR	2.78	**11.48**	0.30	5.50	3.03
SVM	3.37	**26.83**	2.38	20.95	3.21
KNN	13.60	11.85	**17.68**	13.12	15.60
NB	0.60	**38.90**	0.17	4.12	2.84
DT	12.87	16.22	0.42	15.34	**38.85**
BAG	2.94	**8.91**	0.28	7.05	5.43
RF	4.13	5.34	0.28	**5.85**	1.04
ET	3.82	7.87	0.00	**18.96**	1.33
ABC	**19.33**	7.02	0.00	16.99	12.99
GB	**12.44**	3.81	0.02	6.63	4.08
LGBM	7.03	1.85	2.75	**10.39**	3.13
XGB	**11.46**	3.97	0.02	7.47	4.27
MLP	**10.78**	5.08	6.05	7.53	5.69
STACK	6.94	**8.94**	4.32	5.48	1.82
VOTE	**6.38**	4.00	2.11	6.04	0.85

Table 4: Column 1 shows a total of 15 classifiers. Columns 2 to 6 represent the percentage (two decimal places accuracy) of AUC score improvement when tuning each individual dimension while leaving other dimensions at baseline settings for each classifier. Bold entries represent the dimension that contributes to the largest improvement for the specific classifier. MV and LB tend to dominate in performance improvement for most classifiers.

In addition to AUC, 6 other performance metrics are used to measure the performance improvement degree of each dimension. The results in Table 5 reveal that CC brings the greatest improvement regardless of the metrics we use. Contributions from HT and SCALE are relatively small compared to other dimensions.

	AUC	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
CC (%)	15.00	15.58	10.50	16.41	10.50	11.86	10.50
LB (%)	10.81	11.34	9.33	13.27	9.33	10.72	9.34
FS (%)	10.09	7.33	6.30	10.61	6.30	6.94	6.30
MV (%)	7.90	5.30	4.60	5.83	4.59	4.95	4.59
HT (%)	7.46	2.11	3.21	3.41	3.20	4.64	3.20
SCALE (%)	2.45	1.04	0.65	3.03	0.65	0.48	0.65

Table 5: Performance improvement in different metrics of each dimension. The performance improvement of each dimension on other metrics displays an order consistent with that of the AUC score.

Performance Variation across Dimensions For all of the ML configurations, we further investigate how much each dimension contributes to the performance variation in the AUC score. By tuning only one dimension at a time while leaving other dimensions at baseline settings, we obtain a range of AUC scores. Performance variation is the difference between the maximum and the minimum score of each dimension. Fig. 2 shows the proportion of each dimension that brings the performance variation in the AUC score. Based on Fig. 2, we notice that CC, which brings the largest performance improvement, also brings the largest performance variation (10.98 %). After CC, LB (7.00 %), FS (6.93 %), MV (5.64 %), HT (4.97 %), and SCALE (1.66 %) bring decreasing degrees of performance variation in the AUC score.

Table 6 shows the variation of every single dimension on each classifier over the baseline. We observe that for each classifier, if one dimension brings a larger performance improvement, it also results in a larger performance variation. For our assessment of performance variation, the same metrics as above are used for evaluation on each dimension. Using the same metrics as above, Table 7 shows that the proportion of performance variation in different metrics from each dimension follows an order that is consistent with the performance improvement in Table 5. Thus, for different metrics, greater improvement brings greater variation of each dimension. For every step that researchers take when predicting MOF using ML, they should always be aware of the trade-off between benefits (improvement in performance) and risks (variation in performance) when adjusting each dimension.

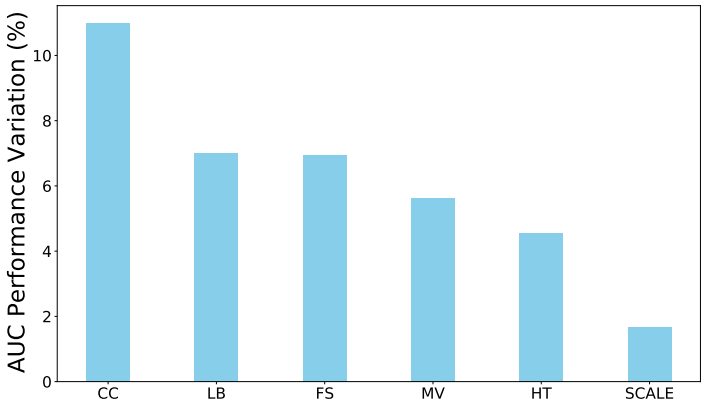


Fig. 2: Performance variation in the AUC score when tuning only one dimension at a time while leaving others at baseline settings. CC brings the greatest performance variation, followed by LB, FS, MV, HT, and SCALE in decreasing order of variation. Larger improvement also brings the risk of larger variation for each dimension.

Classifier	MV (%)	LB (%)	SCALE (%)	FS (%)	HT (%)
LR	2.28	**8.44**	0.25	4.27	2.49
SVM	2.45	**16.87**	1.79	13.04	2.42
KNN	7.97	6.95	**10.36**	7.69	9.14
NB	0.48	**22.22**	0.13	3.25	2.25
DT	7.13	8.99	0.23	8.50	**21.53**
BAG	2.22	**6.17**	0.21	5.26	4.10
RF	3.35	4.14	0.23	**4.49**	0.84
ET	3.22	6.14	0.00	**13.41**	1.12
ABC	**13.09**	4.61	0.00	11.51	9.40
GB	**9.59**	2.83	0.02	5.11	3.14
LGBM	5.54	1.43	2.16	**7.76**	2.47
XGB	**8.70**	3.01	0.02	5.59	3.24
MLP	**7.89**	3.55	4.43	5.30	4.16
STACK	5.47	**6.47**	3.41	4.20	1.43
VOTE	**5.19**	3.13	1.71	4.63	0.69

Table 6: Columns 2 to 6 represent the proportion (two decimal places accuracy) of each dimension that contributes to the performance variation in the AUC score. Bold entries represent the dimension that contributes to the largest variation for the specific classifier. MV and LB tend to result in larger performance variation for most classifiers.

	AUC	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
CC (%)	10.98	11.87	8.57	12.86	8.57	10.60	8.57
LB (%)	7.00	7.29	6.83	9.02	6.83	10.14	6.82
FS (%)	6.93	5.27	4.38	7.62	4.37	4.70	4.38
MV (%)	5.64	4.36	3.69	4.77	3.69	2.98	3.68
HT (%)	4.87	2.55	3.52	1.54	3.52	2.72	3.53
SCALE (%)	1.66	0.88	0.57	1.47	0.57	0.46	0.57

Table 7: Performance variations in different metrics of each dimension. The performance variation of each dimension on other metrics displays an order that is consistent with that of the AUC score.

4.2 Performance Comparison across Classifiers

We have shown that classifier choice is the largest contributor to both performance improvement and variation in the AUC score. Hence, we further investigate the performance differences among classifiers. Specifically, we investigate the relationships among classifier complexity, performance, and performance variation.

Default versus Optimized Performance ***Default*** classifiers are defined as classifiers with default parameters, while ***optimized*** classifiers are those for which hyperparameter tuning with 10-fold cross validation is applied using grid search. We compare the performance of default and optimized classifiers in consideration of all other dimensions, i.e., MV, LB, SCALE, and FS. The average AUC scores of all classifiers with default and optimized settings are shown in Fig. 3. In general, ensemble classifiers perform better than single classifiers regardless of default or optimized performance.

In addition to AUC, 6 other performance metrics are used to evaluate the performance of all classifiers. We use the median score to rank classifiers with both default and optimized settings. Then, NDCG (normalized discounted cumulative gain), one of the most prevalent measures of ranking quality [30], is used to compare classifier rankings between each of these metrics and the AUC score. Detailed relevance scores are shown in Table 8. The result indicates that the median performance of each classifier is similar no matter which metric is used. This also suggests that the AUC score can represent classifiers' overall performance well.

Based on the above experiments, ensemble classifiers should be prioritized in MOF prediction since they usually bring better predictive performance than single classifiers.

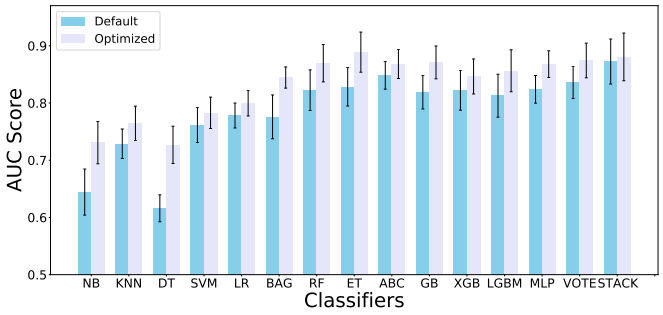


Fig. 3: Comparison of default and optimized performance over all classifiers. Classifiers listed on the left-hand side of BAG are single while the ones on the right-hand side are ensemble and MLP. Overall, ensemble methods have better default and optimized performance compared with single classifiers.

	Default (%)	Optimized (%)
F-score	96.92	97.92
G-mean	96.46	97.63
Precision	95.49	90.01
Sensitivity/Recall	98.42	97.59
Specificity	95.35	97.46
Accuracy	96.46	97.59

Table 8: Column 1 represents 6 other performance metrics. Columns 2 and 3 show the NDCG score between each of these metrics and the AUC score when ranking 15 classifiers by their median performance in default and optimized settings, respectively. Median performance of classifiers is similar regardless of which metric to use.

Performance Variation across Classifiers We measure the performance variation for each classifier in consideration of all other dimensions, i.e., MV, LB, SCALE, FS, and HT. For each classifier, we get a range of AUC scores. The size of the range determines the extent of performance variation. Fig. 4 shows the performance variation in the AUC score of all classifiers. The order of listed classifiers on the x-axis is based on increasing model complexity, which is measured by classifier training time with default settings. The complexity of classifiers and performance variation demonstrates an evident 'U-shaped' relationship. When the classifier is 'too simple', its performance variation is relatively large. When the complexity of the classifier is 'appropriate', the performance variation is relatively small. If the classifier becomes 'too complex', it is also at the risk of larger performance variation. Therefore, classifiers with 'appropriate' complexity are more stable, with smaller changes in performance, while 'too simple' or 'too complex' classifiers are relatively unstable with larger changes in performance in general.

In addition to AUC, the same metrics as above were used to validate the performance variation of all of the classifiers. We use the range (difference between maximum and minimum scores) to rank classifiers in consideration of MV, LB, SCALE, FS, and HT. Then, NDCG is used to compare classifier rankings between each of these metrics and the AUC score. Table 9 displays detailed relevance scores. The result suggests that other metrics show a similar 'U-shaped' relationship between classifier complexity and performance variation as the AUC score. When predicting MOF, it is inappropriate for clinical practitioners to choose 'too simple' and 'too complex' classifiers since they may run the risk of underfitting and overfitting, respectively.

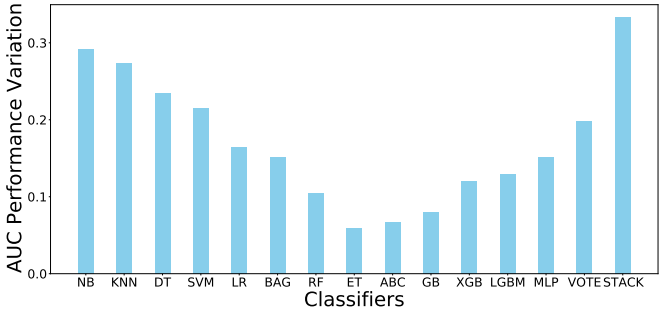


Fig. 4: Performance variation comparison over all classifiers. The order of classifiers listed on the x-axis is based on increasing model complexity. 'Too simple' and 'too complex' classifiers result in larger performance variation. The performance variation of classifiers with 'appropriate' complexity is relatively small.

	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
Relevance (%)	93.15	94.98	94.37	93.24	93.13	93.77

Table 9: NDCG score between each of 6 other performance metrics and the AUC score in terms of classifier complexity and performance variation. Different metrics show a similar 'U-shaped' relationship.

5 Discussion

We have provided a timely MOF prediction using early lab measurements (hour 0), patients' demographic and illness information. Our study quantitatively analyzes the performance via the AUC score in consideration of a wide range of ML configurations for MOF prediction, with a focus on the correlations among configuration complexity, predicted performance, and performance variation. Our results indicate that choosing the correct classifier is the most crucial step that has the largest impact (performance and variation) on the outcome. More complex classifiers including ensemble methods can provide better default/optimized

performance, but may also lead to larger performance degradation, without careful selection. Clearly, more MOF data is needed to provide a more general conclusion. Our work can potentially serve as a practical guide for ML practitioners whenever they conduct data analysis in healthcare and medical fields.

6 Acknowledgments

This work was funded by the National Institutes for Health (NIH) grant NIH R01 - HL149670.

References

1. Harjola, V.P., Mullens, W., Banaszewski, M., et al.: Organ dysfunction, injury and failure in acute heart failure: from pathophysiology to diagnosis and management. A review on behalf of the Acute Heart Failure Committee of the Heart Failure Association (HFA) of the European Society of Cardiology (ESC). European Journal of Heart Failure, vol. 19, pp. 821-836 (2017).
2. Wang, Z.K., Chen, R.J., Wang, S.L., et al.: Clinical application of a novel diagnostic scheme including pancreatic β-cell dysfunction for traumatic multiple organ dysfunction syndrome. Molecular Medicine Reports, vol. 17, pp. 683-693 (2018).
3. Durham, R. M., Moran, J. J., Mazuski, J. E., et al.: Multiple organ failure in trauma patients. Journal of Trauma and Acute Care Surgery, vol. 55, pp. 608-616 (2003).
4. Ulvik, A., Kvåle, R., Wentzel-Larsen, T., et al.: Multiple organ failure after trauma affects even long-term survival and functional status. Critical Care, vol. 11, pp. R95 (2007).
5. Barie, P.S., Hydo, L.J., Fischer, E.: A prospective comparison of two multiple organ dysfunction/failure scoring systems for prediction of mortality in critical surgical illness. The Journal of Trauma, vol. 37, pp. 660-666 (1994).
6. Bota, D.P., Melot, C., Ferreira, F.L., et al.: The Multiple Organ Dysfunction Score (MODS) versus the Sequential Organ Failure Assessment (SOFA) score in outcome prediction. Intensive Care Medicine, vol. 28, pp. 1619-1624 (2002).
7. Dewar, D.C., White, A., Attia, J., et al.: Comparison of postinjury multiple-organ failure scoring systems. Journal of Trauma and Acute Care Surgery, vol. 77, pp. 624-629 (2014).
8. Hutchings, L., Watkinson, P., Young, J.D., et al.: Defining multiple organ failure after major trauma. Journal of Trauma and Acute Care Surgery, vol. 82, pp. 534-541 (2017).
9. Sauaia, A., Moore, F.A., Moore, E.E., et al.: Multiple Organ Failure Can Be Predicted as Early as 12 Hours after Injury. Journal of Trauma and Acute Care Surgery, vol. 45, pp. 291-303 (1998).
10. Vogel, J.A., Liao, M.M., Hopkins, E., et al.: Prediction of postinjury multiple-organ failure in the emergency department. Journal of Trauma and Acute Care Surgery, vol. 76, pp. 140-145 (2014).
11. Obermeyer, Z., Emanuel, E.J.: Predicting the Future — Big Data, Machine Learning, and Clinical Medicine. New England Journal of Medicine, vol. 375, pp. 1216-1219 (2016).
12. Cruz, J.A., Wishart, D.S.: Applications of Machine Learning in Cancer Prediction and Prognosis. Cancer Informatics, vol. 2, pp. 117693510600200030 (2006).

13. Kourou, K., Exarchos, T.P., Exarchos, K.P., et al.: Machine learning applications in cancer prognosis and prediction. Computational and Structural Biotechnology Journal, vol. 13, pp. 8-17 (2015).
14. Asri, H., Mousannif, H., Al Moatassime, H., et al.: Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis. Procedia Computer Science, vol. 83, pp. 1064-1069 (2016).
15. Sharma, K., Kaur, A., Gujral, S.: Brain Tumor Detection based on Machine Learning Algorithms. International Journal of Computer Applications, vol. 103, pp. 7-11 (2014).
16. Wang, Z., Yu, G., Kang, Y., et al.: Breast tumor detection in digital mammography based on extreme learning machine. Neurocomputing, vol. 128, pp. 175-184 (2014).
17. De Bruijne, M.: Machine learning approaches in medical image analysis: From detection to diagnosis. Medical Image Analysis, vol. 33, pp. 94-97 (2016).
18. Farrar, C.R., Worden, K.: Structural health monitoring: a machine learning perspective, https://onlinelibrary.wiley.com/doi/book/10.1002/9781118443118 (2013).
19. Worden, K., Manson, G.: The application of machine learning to structural health monitoring. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 365, pp. 515-537 (2007).
20. Ahmed, Z., Mohamed, K., Zeeshan, S., et al.: Artificial intelligence with multifunctional machine learning platform development for better healthcare and precision medicine. Database, vol. 2020 (2020).
21. Janssen, K.J., Donders, A.R.T., Harrell Jr, F.E., et al.: Missing covariate data in medical research: To impute is better than to ignore. Journal of Clinical Epidemiology, vol. 63, pp.721-727 (2010).
22. Tuba, E., Strumberger, I., Bezdan, T., et al.: Classification and Feature Selection Method for Medical Datasets by Brain Storm Optimization Algorithm and Support Vector Machine. Procedia Computer Science, vol. 162, pp. 307-315 (2019).
23. Chawla, N.V., Bowyer, K.W., Hall, L.O., et al.: SMOTE: Synthetic Minority Oversampling Technique. Journal of Artificial Intelligence Research, vol. 16, pp. 321-357 (2002).
24. Mani, I., Zhang, I.: kNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceedings of workshop on learning from imbalanced datasets, vol. 126 (2003).
25. Batista, G.E., Bazzan, A.L., Monard, M.C.: Balancing Training Data for Automated Annotation of Keywords: a Case Study. In: WOB, pp. 10-18 (2003).
26. Dağ, H., Sayin, K.E., Yenidoğan, I., et al.: Comparison of feature selection algorithms for medical data. In: 2012 International Symposium on Innovations in Intelligent Systems and Applications, pp. 1-5 (2012).
27. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: Machine learning in Python. the Journal of machine Learning research, vol. 12, pp. 2825-2830 (2011).
28. Bakker, J., Gris, P., Coffernils, M., et al.: Serial blood lactate levels can predict the development of multiple organ failure following septic shock. The American Journal of Surgery, vol. 171, pp. 221-226 (1996).
29. Papachristou, G.I., Muddana, V., Yadav, D., et al.: Comparison of BISAP, Ranson's, APACHE-II, and CTSI Scores in Predicting Organ Failure, Complications, and Mortality in Acute Pancreatitis. American Journal of Gastroenterology, vol. 105, pp. 435-441 (2010).
30. Chen, W., Liu, T.Y., Lan, Y., et al.: Ranking measures and loss functions in learning to rank. In: Advances in Neural Information Processing Systems, vol. 22, pp. 315-323 (2009).

Cloud Supportive Multi-Stage Selection Model for Multiple Imputation in Large Scale Study: Non-Ignorable NMAR

Jagan Mohan Reddy D[0000-0002-3894-6315], Seelam Srinivasa Reddy[0000-0002-3190-5166] and G Vijaya Suresh[0000-0003-4086-4164]

Lakireddy Bali Reddy College of Engineering, Mylavram, Andhra Pradesh, India - 521230
{jagan.reddy507,seelamsrinivasareddy,vijaysuresh.g}@gmail.com

Abstract. Today, in most of the applications that are deal with large volumes of data, a lack of response may lead to biased and inconsistent if the characteristics of respondents differ systematically from those who refuse to participate. It has become a more dangerous issue to extract meaningful conclusions from such kinds ofdata, which are largely affected by missing data in large volumes, while resulting in biased outcomes. Another important consideration of missing data is the assumption of ignorable or non-ignorable. However, data in the clinical studies include a high proportion of missing data with much varied distributions. Hence, it is necessary include NMAR (Not missing at Random) analysis in regular methods to improve their effectiveness. Sample selection, a popular method applied in the analysis of medical data is focused on the fact that the missing data cannot be ignored. The study focuses on the non-ignorable data to fit to models used for large samples. To simplify the model complexity, the proposed method is run over cloud framework that distributes regularized models among workers having totally shared probability. The results show that the proposed method has a fast convergence and remains unbiased. It can accurately calculate missing values, which in turn indicates that the proposed method provides an accurate distribution value.

Keywords: Data Missing; Multivariate Data; Cloud Computing; Maximum Likelihood

1 Introduction

Most recent applications demand complete case analysis, especially in medical case studies, which otherwise affects the final hypothesis. In complete case analysis, it is important to deal with missing data that tends to get affected in various distributions including Nominal, Gaussian, Bivariate etc [1][2]. Moreover, it is also much difficult in case of non-response and imputation demands for clearly tracking both missing and observation parts [6] [7]. The selection model offers better solution to derive imputation strategy for such kinds of data. But the model limits itself to just two steps. Hence, the proposed study seeks to use the experimental results to create multi selection model for handling missing data with different distributions [3][4][5]. In order to

fit the model to higher dimensional studies, it necessarily requires some additional assumption than what is found in the regular multiple imputation strategy i.e. the regularized method [8], which are better suited to large samples and enforce direct mechanism of applying such methods to the selection model. Though a number of alternatives are available to estimate unknown parameters in regularized methods, yet this study is focused on the maximum likelihood to fit large samples only. Then, the study is extended to run complete analysis over cloud supportive environment in order to reduce model complexity while executing joint modeling in large data. Section 2 describes the related works, while the selection model is also introduced. Section 3 presents Multiple Imputation using Multi-Stage Selection Model, while the experimental results analysis and discussion is done in section 4. This study concludes with section 5.

2 Selection of Model

Usually, a selection model estimates maximum likelihood under the assumption of normality. Later, others have applied normally applied skew distribution using the log likelihood function [9]. Sample selection modeling framework is used as a model for MI imputation method and implemented with full information maximum likelihood method (FIML) [10]. Standard selection models consist of two parts viz., measurement and missing. In the standard two step model, single distribution is applied to the missing part, which is handled in two steps. However, this model does not provide solution to a model having multiple distributions. Such distribution studies are handled by applying multi stage selection that includes multiple parts viz., measurement and missing. However, this work includes missing which includes normal and bivariate t-distribution affected with two variables. Missing expresses different processes when the selection model uses factorization. Considering cases where the missing data is first distributed completely in random missing i.e. (MCAR) $\mathbf{P(\Re \mid X, \psi) = P(\Re \mid \psi)}$, second at random (MAR): $\mathbf{P(\Re \mid Z, \psi) = P(\Re \mid Z_o, \psi)}$ and last in not random $\mathbf{P(\Re \mid Z, \psi) = P(\Re \mid Z_o, Z_m \psi)}$. Let representing complete model with linear with multiple regression in which outcome to be considered $\mathbf{Z_i^*}$ and set of correlated predictors considered to be $\mathbf{x_i}$.

$$\mathbf{Z_i^* = \beta' x_i + \sigma \varepsilon_{1i} \;\; i = 1,2,...,K} \qquad (1)$$

Suppose the main pattern is complemented by a multiple equation (missing), then;

$$\mathbf{S_i^* = \gamma_1' u_i + \varepsilon_{2i} \;\; i = 1,2,...,K} \qquad (2)$$

This is true in small samples but in the case of the current high dimensional model, where large number of covariate variables (xi) are effected with different distributions missing. It is essential to extend two model selection step multistage model to handle more selection equations. The equation (2) represents selection model normal to handle for which Helmans introduced selection distribution. Now, the multi-stage study is applied to different selection models. The equation (3) used for the distribution of two adjustment variables for model selection;

$$\mathbf{R_i^* = \gamma_2' v_i + \varepsilon_{3i} \quad i = 1,2,...,K} \qquad (3)$$

where $\beta, \gamma_1, \gamma_2$ and σ parameters known to estimate prior unknown; ($\mathbf{x_i}, \mathbf{u_i}$ and $\mathbf{v_i}$), set of parameter to relate to distribution of missing; ($\boldsymbol{\varepsilon_{1i}}, \boldsymbol{\varepsilon_{2i}}$ and $\boldsymbol{\varepsilon_{3i}}$) are random errors with zero means , variances one and correlation ρ. If we look (4)

$$\mathbf{S_i = I(S_i^* > 0)} \text{ and } \mathbf{R_j = J(R_i^* > 0)} \text{ , } \mathbf{Z_i = Z_i^* S_i R_j} \quad (4)$$

It is observed that distribution of model is affected by bivariate normal distribution and is represented as:

$$\begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \end{pmatrix} \sim N_2 \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right\}$$

$$\begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \end{pmatrix} \sim t_2 \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}, v \right\}$$

The range of ρ in bound with -1 to +1 and is used to estimate relation of correlation among observed and missing data. To represent PDF of bivariate t distribution t_2 is used; and finally the degree of freedom is mentioned as $\mathbf{v}$.

The joint density distribution of $\mathbf{Y^*, S^*, Y, R^*}$ with selection framework to be

$$\mathbf{P(Z^*, S^*, R^* \mid x, u, v, \beta, \gamma_1, \gamma_2) = P(Z^* \mid x, \beta) P(S^* \mid Z^*, u, \gamma_1) P(R^* \mid Z^*, v, \gamma_2)}$$

Further, it is represented as:

$$\mathbf{P(z \mid x, S^* > 0, R^* > 0) = \frac{P(z \mid x) P(S^* > 0 \mid z, u) P(R^* > 0 \mid z, v)}{P(S^* > 0 \mid u) P(R^* > 0 \mid v)}} \quad (5)$$

The density of observation part is expressed as:

$$\mathbf{P\left(z \mid x, S=1; \Theta_1\right) = \frac{1}{\lambda} \varphi\left(\frac{z - \beta' x}{\lambda}\right) \varnothing \left(\frac{\gamma_1' u + \rho\left(\frac{y - \beta' x}{\sigma}\right)}{\sqrt{1 - \rho^2}} \right) \Big/ \varnothing(\gamma_1' u)} \quad (6)$$

$$\mathbf{P\left(z \mid x, R=1; \Theta_2\right) = \frac{1}{\lambda} \varphi\left(\frac{z - \beta' x}{\lambda}, v\right) T\left(\frac{\gamma_2' w + \rho\left(\frac{z - \beta' x}{\lambda}\right)}{\sqrt{1 - \rho^2}} \right) \left(\frac{v+1}{v + \left(\frac{z - \beta' x}{\lambda}\right)^2} \right)^{1/2}, v+1 \Big/ T(\gamma_2' w, v)} \quad (7)$$

Where $\Theta_1 = (\beta, \lambda, \gamma_1, \rho)$ and $\Theta_2 = (\beta, \lambda, \gamma_2, \rho)$ are the densities specified by distributions related to the observed data. **T(.)** is the cumulative of prediction distribution related to error with degree of freedom. To consider the case of missing done at random cases, equation (6) fitted with correlation at zero, while in non-random cases, it is other thanzero, which helps model to derive unbiased results. The final density becomes possible with a combination of density(s) mentioned in (6) and with discrete componentof **P (S = 1,)** and **P (R = 1,).** Sample selection model with likelihood inference to mentioned that:

$$\int_L (\Theta) = \sum_{i=1}^{k} \mathbf{S_i} (\ln \mathbf{P(z_i \mid x_i, S_i = 1, \Theta_1; R_i = 1, \Theta_2)} + \sum_{i=1}^{k} \mathbf{S_i} \left(\ln \varnothing \left(\gamma_1' \mathbf{u_i} \right) \right) + \sum_{i=1}^{k} R_i \left(\ln \mathbf{T} \left(\gamma_2' \, \mathbf{w_i}, \mathbf{v} \right) \right)$$

$$+ \sum_{i=1}^{k} \mathbf{S_i} \left(\mathbf{1 - S_i} \right) \ln \left(\varnothing \left(-\gamma_1' \mathbf{u_i} \right) \right) + \sum_{i=1}^{k} \mathbf{R_i} \left(\mathbf{1 - S_i} \right) \ln \left(\mathbf{T} \left(-\gamma_2' \mathbf{w_i}, \mathbf{v} \right) \right) \quad (8)$$

In multi-step estimator, observed data conditional expectation to be specified as:

$$\prod\left(z \mid x, S^{*}>0, R^{*}>0\right)=\beta' x+\sigma\rho\Lambda_{1}\left(\gamma_{1}' u\right)^{*}\Lambda_{v}\left(\gamma_{2}' w\right), v>1 \qquad (9)$$

To estimate $\Lambda(.)$ used inverse Mills ratio and $\Lambda_{v}(k)=\frac{v+k^{2}}{v-1}\frac{t(k;v)}{T(k;v)}$. To calculate $\sigma\rho$ much of additional information need to be included in (9) and are to be specified a $\Lambda(\hat{\gamma}_{1}' u_{1}) \& \Lambda_{v}(\hat{\gamma}_{2}' w_{2})$.

3 MI with Multi-Stage Selection Model for NMAR Missing Data

Multiple imputation (MI) method where number M of independent draws extracted Y_{mis} from $f(Y_{obs}|Y_{mis})$ and generates **M** complete data sets. After that for each data sets unknown parameters estimated and results single inference. There are number of approaches used by MI among one is Joint Modelling (JM) which generates posterior distribution of missing data over observed data. Multiple imputations are based on the strategy of generating multiple observations for each missing data point, resulting in the creation of several complete data sets. It is observed that imputation done at case of MAR where both observed data distribution and response both are together same and it to be represented as $P(z \mid x, S^{*}>0, R^{*}>0) \leq P(z \mid x, S^{*} \leq 0, R^{*}<0)$. The case is not existed with the assumption of MNAR. However, it is observed that there exists positive correlation among outcome and selection error in case where $S^{*} \leq 0$ and $R^{*}<0$. Considering all above assumption and defined imputation model for missing to be represented as:

$$\prod\left(z \mid x, S^{*}<0, R^{*}<0\right)=\beta' x-\lambda\rho\Lambda_{1}\left(-\gamma_{1}' u\right)^{*}\Lambda_{v}\left(-\gamma_{2}' w\right) \qquad (10)$$

By using (10), the last generated equation is used to impute missing data by using multi-stage selection method as:

$$Z_{i}^{*}=\beta^{*\prime} x-(\lambda\rho)^{*}\Lambda_{1}\left(-\gamma^{*}{}_{1}' u\right)^{*}\Lambda_{v}\left(-\gamma^{*}{}_{2}' w\right)+\eta^{*} \qquad (11)$$

Where $\eta^{*}=\mathbb{N}(0,\lambda_{\eta}^{2*})$ $Y^{\lambda_{\eta}^{2*}}$, β^{*}, $(\lambda\rho)^{*}$ and are all drawn using approximate proper allocation.

3.1 Multiple Imputation using Multi-Stage Selection Model with Maximum Likelihood (ML)

Most of the cases, the standard methods are generally replaced with likelihood calculations to improve efficiency and accuracy of algorithms. Similarly, generally a two-

step method is less efficient when compared to ML assumption. Considering this, we have proposed a MI algorithm with the assumption of maximum likelihood. Let's observe the missing data of $\mathbf{Z}^*$ to be considered as $\mathbf{Z}_{i,o}$ and $\mathbf{Z}_{i,m}$ then imputes missing values using random draws into $\mathbf{M}$ times and it can be represented as $\mathbf{Z}_{i,m}^{(m)} = \mathbf{P}(\mathbf{Z}_{i,m} \mid \mathbf{Z}_{i,o}, \mathbf{x}_i)$ where $\sum_{i=1}^{M} \mathbf{m}$ and $\mathbf{P(.)}$ represents the posterior predictive distribution. Always it is not possible to draw such kinds of distribution with a normal process, which can be done with iterative strategies like data augmentation, and chain concepts. It is also observed that such kindsof methods consume heavy computation.

Algorithm: MI with Inference with Maximum Likelihood in Large Samples

1. Consider set of $\mathbf{r}$ complete observed variables in $\mathbf{Z}_1$ and perform imputation with $\mathbf{m}$ times and to be denoted as $\mathbf{Z}^{(m)}$ which is a combination of $(\mathbf{Z}^{(m)}_{o,1}\mathbf{Z}^{(m)}_{o,-1})$ .After that extract with highly predictor observable data which is to known active set and is denoted as $\hat{\mathbf{Z}}$.

2. The distribution of estimation can be done with using proposed MLE and it was mentioned $\mathbf{P}\left(\theta \mid \mathbf{Z}_{o,1}, \mathbf{Z}_{o,\hat{Z}}\right)$

3. Finally missing values are $\mathbf{Z}_{m,1}^{(m)}$ imputed with $\mathbf{m}$ times with random draws using distribution $\mathbf{P}(\mathbf{Z}_{o,1}\mathbf{Z}_{m,-1}, \Theta^{\dagger(m)})$, and estimator which is to be drawn with $\Theta^{\dagger(m)}$ using $\mathbf{P}\left(\Theta^{\dagger(m)} \mid \mathbf{Z}_{o,1}, \mathbf{Z}_{o,\hat{Z}}\right)$

Therefore, it demands new kinds of inferences which include maximum likelihood, and Bootstrap. General mechanism of imputation with multi step model which was mentioned in (10) demands for true value estimation represented as θ and it's practically unknown. To make process of model estimation with less complex study, we consider maximum likelihood estimator (MLE) Θˆ, which is an alternative θ. Now, the model with conditional distribution can be represented as:

$$\mathbf{P}(\mathbf{Z}_{i,m} \mid \mathbf{Z}_{i,o}, \mathbf{x}_i) = \int \mathbf{P}(\mathbf{Z}_{i,m} \mid \mathbf{Z}_{i,o}, \mathbf{x}_i, \hat{\Theta}) \prod (\hat{\Theta}) d\hat{\Theta} \qquad (12)$$

The distribution of MLE is to be considered as $\prod(\hat{\Theta})$but the problem with this estimator is not fit for model selection mentioned in (10). To make estimator closed to (10) derived an alternative estimator which was represented as $\Theta^{\dagger}$ and it helps to draw missing data which was mentioned earlier in (10).

$$\mathbf{Z}_{i,m}^{(m)} = \mathbf{P}(\mathbf{Z}_{i,m} \mid \mathbf{Z}_{i,o}, \mathbf{x}_i, \Theta^{\dagger(m)}) \qquad (13)$$

The newly estimated MLE is a composed estimator and all are drawn with MLE and to be considered as $\Theta^{\dagger(m)} = (\beta^{\dagger(m)}, \lambda^{\dagger(m)}, \gamma^{\dagger(m)}, \rho^{\dagger(m)})$.But in case of large samples

need to consider additional estimation to be represented as $\Theta^{\dagger}_{ML}$ along with covariance matrix and is represented $C(\Theta^{\dagger}_{ML})$ and finally derive information with $\Theta^{\dagger}_{ML} = N((\Theta^{\dagger}_{ML}, C(\Theta^{\dagger}_{ML}), C(\Theta^{\dagger}_{ML})$ using Fully Inverse Maximum Likelihood(FIML).Consider full set of variables Z where $Z_{o,-1}$ represents all observations except $Z_{o,1}$ and also mentioned known active set $|Z| = w$ which are most preferable predictor variables to impute Z_1, with $Z_{o,\hat{Z}}$. Now the model of regularized MI with active set to be represented as:

$$P\left(\Theta^{\dagger(m)} \mid Z_{o,1}, Z_{o,\hat{Z}}\right) \tag{14}$$

3.2 Multiple Imputation using Multi-Stage Selection Model with Bootstrap

An extension to MLE estimation with bootstrap used as a second approach to perform predictive distribution (10) and is drawn to be mentioned here:

$$Z^{(m)}_{i,m} = P(Z_{i,m} \mid Z_{i,o}, x_i, \Theta^{\propto(m)})$$

Algorithm: MI with inference with Bootstrap in Large Samples

1. Consider $Z^{(m)}$ with complete case observation of size n and m to be number of imputations.
2. Generate $\hat{\beta}$ bootstrap samples are drawn $Z^{*(m)}_{B}$ (missing data included).
3. Next, estimate $\Theta^{\propto(m)}$ with random draw using $P(\Theta^{\propto(m)} \mid Z_{o,1}, Z_{o,-1})$.
4. Finally, missing data $Z^{(m)}_{miss,1}$ to be imputed into m times using predictive distribution $P(Z_{m,1} \mid Z_{m,-1}, \Theta^{\propto(m)})$ and return imputed data sets $Z^{*(m)}_{I,B}$ of $X^{*(m)}_{B}$.Where $\Theta^{\propto(m)} = \Theta^{*\propto(m)}_{I,B}$ is an estimator corresponds to bootstrap data set and it to represented as.

Where $\Theta^{\propto(m)} = (\beta^{\propto(m)}, \sigma^{\propto(m)}, \gamma^{\propto(m)}, \rho^{\propto(m)})$ are to be MLE estimators on $B^{(m)}_{boot}$ the set. The idea of implementing MI is now done with estimator of original set represented as $\theta^{(m)}$ on missing data $Z^{(m)}_{i,m}$ with boot estimator $\Theta^{\propto(m)}$.

4 Framework for Cloud assisted Multiple Imputation using Multi-Stage Selection Models

Cloud computing is now an emerging area that offers resources with huge data centers to support huge applications in large scale operations, especially in healthcare analysis. In this approach, locally owned infrastructure is replaced with private resources for computational cost minimization. To retain workflow execution in standard physical environment, it consumes more time to finish. Imputation task, also known as one workflow execution and adaptive cloud supportive environment is the ideal platform to run such kind of executions. However, resources from single cloud sometimes do not fulfill the avail ability of resources and raises a number issues like unused, network delay and etc. The approach of using a single cloud has several limitations in terms of availability, avoiding [11] [12]. The better solution is to make arrange of number of alternative clouds called multi cloud environment and helps to solve the issues of single cloud with support of functionality interoperability [13] and the framework shown in Figure 1.

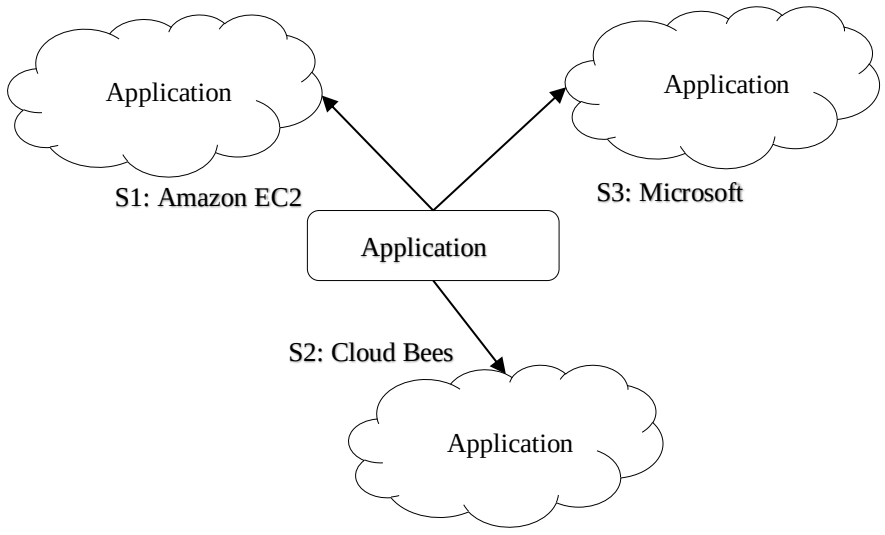


Figure 1. Heterogeneous Cloud Provisioning Workflow Applications

Table 1: Cloud Services

Service	Kind	Configuration
S1	Eucalyptus Machines:c1.xlarge	1 GB of RAM, the worker has a CPU node I5, Ubuntu 11.04
S2	OpenStack Machines:m1.large	1 GB of RAM, worker node has a CPU I5, Ubuntu 11.04
S3	AmazonEC2	1 GB of RAM, the worker node has a

	Machines:m5.xlarge	CPU I3, Ubuntu 11.04
S4	Amazon EC2 Machines: f1.16xlarge	2GB RAM, worker node has a CPU I7 and 16 GB RAM), Ubuntu 11.04

Table 1 shows cloud frame supported configurations with different services and it is served to implement the proposed algorithms. Among the various provisioned services, we have chosen heterogeneous cloud services offered by Amazon and Eucalyptus. It is noted that with this type of delivery framework, execution will never stop, even with the turn's laps due to the availability of cloud service. There is a possibility to extend the functionality of the execution of the imputation workflow as well. However, there is a problem with latency issues foe which we need to explore methodologies to manage.

Algorithm: Cloud Provisioned Multiple Imputation

1. **Entry:** Task Allocation Workflow Tw
2. **Entry:** Set of Resources Workflow Ew
3. **Exit:** Time(T), Speed(S) of n tasks with R
4. $m=0$ //initialize number allocations
5. $(T,S) = \phi$ //unknow
6. While $j=n$ do
   If any $(\Delta j(Ew))$
   $(X_I^{(m)},T,S) = X_M^{(m)}$ //Imputation Algorithm
   end if
   end while
7. Return (EW=CreateResource (R=[C,CC,P,M]) finish

5 Results and Discussion

5.1 Simulation and Results of Medical Data

Lets' consider the medical data set Y for the model of regression, whose result may depend on many explanatory variables x1, x 2 ,...., xi of X . The result of Y is measured for all variables, while some others in common case variables in the design matrix X missing. It is common in imputation model to gather relevant outcome earlier for the computation of results. We compared the performance of the proposed method based on the regularized strategy with the inference of bootstrap and maximum likelihood in multistep selection-based imputation methods. Consider the setting wherein, the simulation models and selection results have multivariate distribution error. The result is in the equation $\mathbf{Y_i = 0.1 - 1.5x_{1i} + 1.2x_{2i} + \varepsilon_{1i}}$ where $\mathbf{x_{1i} = x_{2i} = N(0,1)}$ and

$\sigma = 1$.The selection models affected with normal and bivariate distribution error are represented individual as, $S_i = 1 + 2x_{1i} + 0.2x_{2i} + 1.5u_i + \varepsilon_{2i}$ and $R_i = 1 + 3x_{1i} + 1.2x_{2i} + 3w_i + \varepsilon_{3i}$ $(u_i, w_i) = N(0,1)$, $\beta_0 = (0.5,1.5,1.0)$, Y , $\gamma_0 = (1,1,0.2,1.5)$ and $(1,1,0.2)$, $v = 5$, $\rho = 0.3,0.5,0.7$, In covariance matrix is σ = 1 . It is also mentioned clearly in both distributions that the observed values are positive and affected with 30% and 25% of missing distribution. When studied with a range of combinations, including small simulations to large simulations within each of the possible combinations of the correlation coefficient and missing is detailed in Table 2 with three missing mechanisms including MNAR.

Table 2: Test Cases in Medical Study

Case	Coefficient	Miss-Prop	Imputations	Simulations
A	0.1	0.1	1-10	1000
B	0.3	0.3	50-100	5000
C	0.5	0.5	100-500	10000

Table 3: Performance of the Proposed use of Cloud Services Algorithm

Services	No. of Cores	CASE: A Speed (in sec.)	CASE: B Speed (in sec.)	CASE: C Speed (in sec.)
1 x S1	4	2.32	1.59	0.66
2 × S3	4	3.17	2.10	0.83
3 × S3	6	4.32	2.79	1.06
2 x S4	8	5.89	3.73	1.38
2 x 1 x S1 and S2	12	9.46	5.88	2.09
2 x S4	16	10.92	6.75	2.38
4 x S4	32	23.89	14.53	4.98

The service, 4 × S4, which should show better performance compared to all other cases. In Amazon Elastic Compute Cloud, Case C gives lower speed, wherein the algorithm did not reach its theoretical speed up (i.e., 6). However, the study produced promising outcome by combining multiple services provisioned from multiple clouds. The study focuses entirely on the missing data that can be ignored, particularly MNAR and general results shown in Table.3 in which totally standard methods are compared with proposed one. The methods tested includes M1: Series Regression Multiple Imputation (SRMI), M2: Bayes Multiple Imputation, M3: Bootstrap MI, M4: Lasso MI, M5: Bayesian Inference to Additive MI, M6: Additive Bootstrap MI, M7: Two Stage Regularized MI with Bootstrap, M8: Two Stage Regularized MI with Maximum Likelihood, M9: Cloud Assisted Multistage Regularized MI with Bootstrap, M10: Cloud Assisted Multistage Regularized MI with Maximum Likelihood. The results produced with three cases A, B and C and each possible combination includes i) the sample sizes (low = 1000, medium = 5000, high = 10,000) ii) auxiliary

variables (small = 50, medium = 100, high = 500) (iii) levels of correlation (low = 0.1, medium = 0.3 and high = 0.5) (iv) the percentage of missing observations (low = 0.1, medium = 0.3 and high = 0.5).

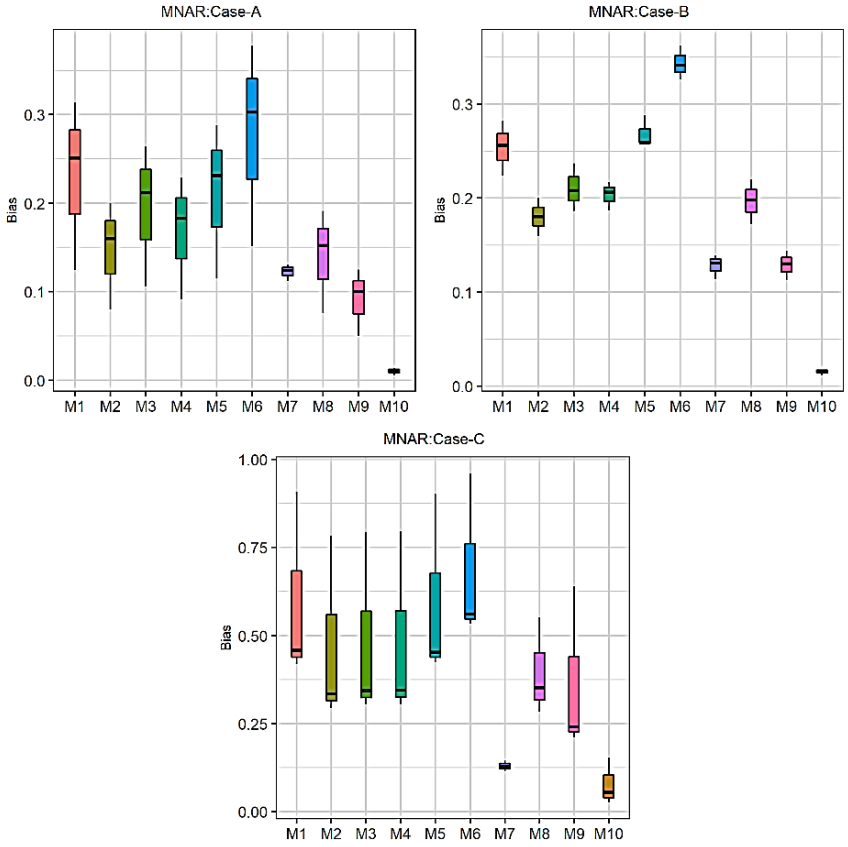


Figure 2. Proposed Outcome Bias in the Cloud Environment

The model is followed directly by fitting a linear regression in iterative approach. The presentation of the results is oriented with benchmarks that includes bias and coverage. Figures 2 & 3 shows the simulation results (i.e., the cases A, B and C) with assisted cloud that proposed multiple selection MI over normal and bivariate error in terms of bias and coverage performance. The first observation made is the proportion of missing affected seriously over the bias performance and it also decreases with the high rate of missing. Moreover, the same results can be obtained in case of coverage ratio too. The results show the contribution on methods of glass between me with cloud service provisioning offer better results, both for polarization and coverage. The second important observations are with small amount of missing rate that results in much higher coverage and constraint of higher correlation. The results from the Cloud

supportive multiple imputation models are less biased and provided more coverage in all scenarios. However, a correlation of 0.7 gave only better gains in terms of bias and coverage. When the case correlation and missing is high, the results from the proposed cloud-assisted multiple imputation were very similar in both Case B and C.

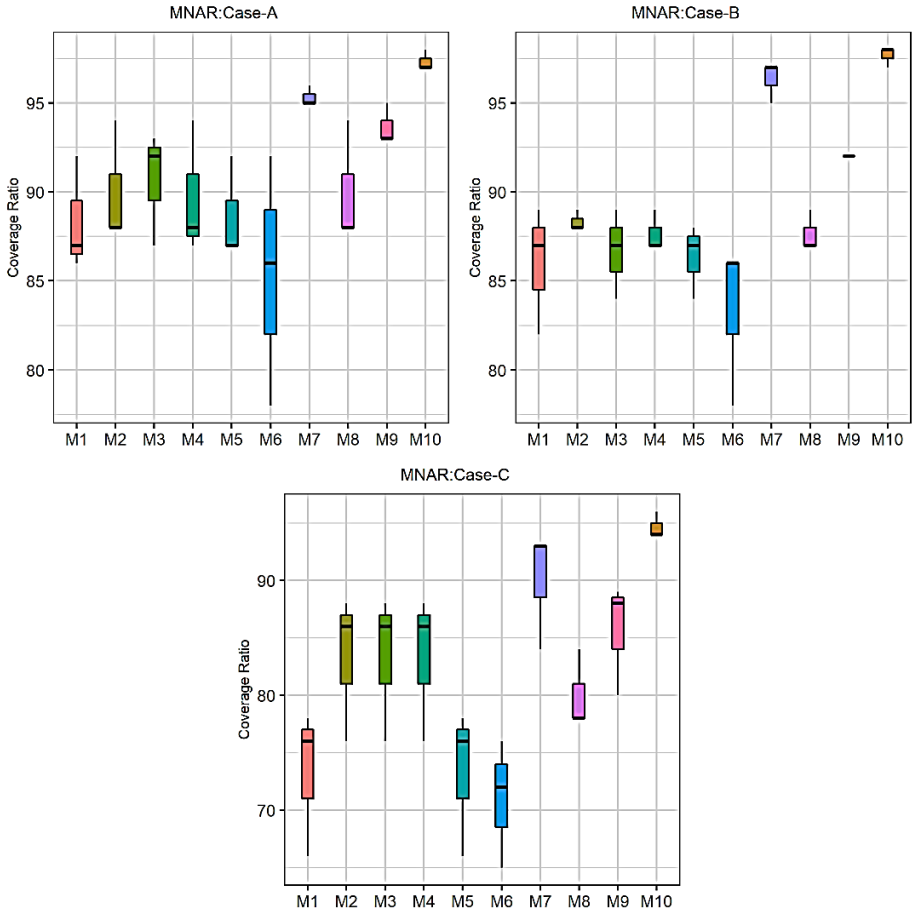


Figure 3. The proposed algorithm coverage in the cloud environment.

6 Conclusion

A Multi-Stage Selection model with maximum likelihood and bootstrap estimator for imputing missing data with various distributions in NMAR is reviewed and compared in a simulated study. The general standard MI analysis gives biased and inconsistent estimates under NMAR mechanism. To use cloud services, the regularized methods must assume that imputation can be distributed over multiple workers. The assump-

tion is valid for the multivariate data that reduced model complexity. In essence, the following two points are worth mentioning here:
(1) imputation of multivariate missing data is crucial in reaching the conclusion and;
(2) The proposed method fits large samples by joining selection model with regularized equation.
However, as the study does not completely focus on other patterns, especially (MAR and MCAR), there is a need to further extend this work. To sum up, it may be noted that multi cloud framework issues need to be further handled in future studies.

References

1. P. M. t. K. C. S. d. V. a. M. E. P. Kei Long Cheung, "The impact of non-response bias due to sampling in public health studies: A comparison of voluntary versus mandatory recruitment in a Dutch national survey on adolescent health," BMC Public HealthBMC series, vol. 17, no. 276, pp. 1-10, 2017.
2. Y.-H. C. Chi-hong Tseng, "Regularized approach for data missing not at random," Statistical Methods in Medical Research, vol. 28, no. 1, pp. 134-150, 2019.
3. Y. L. Little R, "Intent-to-treat analysis for longitudinal studies with dropouts.," Biometrics., vol. 52, pp. 1324-1333., 1996.
4. C. S. P. C. M. R.-R. Galimard J, "A multiple imputation approach for MNAR mechanisms compatible with Heckman's model.," Stat Med., vol. 35, pp. 2907-2920., 2016.
5. F. F. J. Shao, "Model selection with nonignorable nonresponse," Biometrika, vol. 103, no. 4, p. 861–874, 2016.
6. R. S. R. Devi Priya, "Pre-processing of microarray gene expression data for classification using adaptive feature selection and imputation of non-ignorable missing values," International Journal of Data Mining and Bioinformatics, vol. 16, no. 3, p. 183, 2016.
7. S. A, "An Analysis of Selection Models For Non-ignorable Dropout: An Application to Multi-centre Trial Data.," J Biom Biostat , vol. 6, p. 246, 2015.
8. Y.-H. C. Chi-hong Tseng, "Regularized approach for data missing not at random," Statistical Methods in Medical Research, vol. 28, no. 1, pp. 134-150, 2017.
9. N. B. A. Jamalizadeh, "A two-parameter generalized skew-normal distribution," Statistics & Probability Letters, vol. 78, no. 13, pp. 1722-1726, 2008.
10. G. M. a. S. R. L. Bart Michiels, "Selection Models and Pattern-Mixture Models for Incomplete Data with Covariates," Biometrics, vol. 55, no. 3, pp. 978-983, 1999.
11. " Summary of the Amazon EC2 and Amazon RDS Service Disruption ,https://aws.amazon.com/message/65648/," Amazon , 2014. [Online].
12. K. Gai, M. Qiu and H. Zhao, ""Cost-Aware Multimedia Data Allocation for Heterogeneous Memory Using Genetic Algorithm in Cloud Computing," IEEE Transactions on Cloud Computing, p. 99.
13. D. Petcu., "Multi-Cloud: Expectations and current approaches.," in In Proceedings of the International Workshop on Multi-Cloud Applications and Federated Clouds (Multi-Cloud'13). ACM, New York, NY, 2013.

Assessing long term degradation of industrial assets based on their production output curves

Pierre Dagnely, Peter Van Hese, Tom Tourwé, and Elena Tsiporkova

Sirris - Elucidata Innovation Lab, A. Reyerslaan 80, 1030 Brussels, Belgium,
(pierre.dagnely, peter.vanhese, tom.tourwe, elena.tsiporkova)@sirris.be

Abstract. Detecting and quantifying long-term degradations is an important problem in many industrial settings as it helps planning maintenance and assessing performance. Asset components wear over time which will increase the likelihood of failures and down time if they are not replaced in time. In this paper, we present a novel methodology for long-term degradation detection. Our methodology is based on the evaluation of the evolution of the linear relationship of the production output curve, i.e., the relation between the asset's production and the main feature impacting it, such as the Irradiation for photovoltaic (PV) plants. We have implemented this methodology for the PV domain, and we have benchmarked it with the current state-of-the-art methodologies, relying on trend detection, using real-life datasets. We have shown that our methodology allows to detect similar degradation than the state-of-the-art methodologies but was able to do so earlier on, e.g. to detect the degradation after 6 months of asset's usage, while the other methods needed up to 2 years of data before detecting real gradual degradations.

Keywords: Long term degradation · Linear regression · Photovoltaic domain

1 Introduction

Long-term asset degradation is an important challenge in many industrial settings, and detecting and quantifying it can help planning maintenance and assessing performance more correctly. Asset components wear over time, which increases the likelihood of failures and down time if they are not maintained in time. Typically, manufacturers guarantee that an asset will perform correctly for a number of cycles, e.g. a number of days or kilometers. This number of cycles is only an estimation based on the asset's characteristics (e.g. type of materials used) and the results of lab tests performed in general operating conditions. However, the wear is strongly influenced by the asset's actual operating conditions and the actual wear can thus significantly diverge from the theoretical one. Therefore, it would be useful to be able to infer the exact wear from the actual monitoring data generated by the asset.

Nowadays, most industrial assets are equipped with sensors. These sensors provide continuous monitoring of both the asset production/usage and the en-

vironmental context. Based on this monitoring, a more precise estimation of the actual wear is possible.

In this paper, we will focus on the detection of long-term degradation in photovoltaic (PV) plants. In this context, traditional methods focus on detecting trends in the asset's Key Performance Indicator (KPI) such as the Energy Performance Index (EPI) or the Performance Ratio (PR). A wear should imply a slow degradation of the plant production and should therefore be reflected in the KPI. However, these KPIs are often inaccurate and noisy, with error margins of up to 20%[3]. Traditional trend detection methods used in the PV domain are not sufficiently robust to such noisy data. Hence, they often struggle to assess the long-term degradation accurately. In addition, some methods only detect degradation, but are not able to precisely quantify it, e.g. by outputting a degradation percentage representing the amount of increase/decrease in efficiency. Without that precise estimation, domain experts can not easily decide if an asset need to be replaced or not. On top of that, the existing methods only provide an indication of degradation after a significant period in time, e.g. 2 years, because they rely heavily on trend detection techniques which do not work well on shorter periods of time. Although one could argue the effects of long-term degradation take time to impact an asset's performance, it is relevant to monitor degradation early in the lifetime of the asset. Assets operating in very harsh evironments, for example, could show signs of degradation very early, or faulty installations that create an important strain on the asset could also be detected early on.

In this paper, we present a novel methodology addressing these three aspects, ensuring its applicability in an industrial setting. The methodology is based on the evaluation of the evolution of the production output curve, i.e., the linear relation between the asset's production and the main feature impacting it, such as the Irradiation for PV plants. This methodology has been implemented and validated in the PV domain but can be applied to any type of assets where the production is mainly dictated by a single factor, e.g. for wind turbines that are mainly impacted by wind speed.

This paper is organized as follows. The application domain is explained in Section 2; in Section 3, the state of the art in long-term degradation detection in the PV domain is presented; in Section 4, we explain our methodology for long term degradation detection; in Section 5, we benchmark our methodology against the traditional methods using a real-world dataset. Finally, we conclude the paper with a discussion in Section 6.

2 Application domain

PV plants are composed of several PV modules that convert irradiation into direct current. These modules are connected to one or several inverters that convert the direct current to alternate current, which is sent to the grid. These plants are now continuously monitored at the inverter level, as this is the most important device. In addition, various sensors are present in the plant, measuring the irradiation, electricity production, etc. Weather data, such as irradiation or

rainfall, are typically also collected from nearby weather stations or satellite measurements.

Two KPIs are usually used in the PV domain: 1) Performance Ratio (PR) and 2) Energy Performance Index (EPI). Both KPIs output a percentage value, where 100% corresponds to perfect production, i.e., the inverter is producing as expected given operational context (e.g., weather conditions) and 0% indicating that the inverter is not producing. The two KPIs differ in the way they represent the operational context. PR only considers the irradiation reaching the plant while EPI considers the irradiation and the temperature. The formulas to compute the two KPIs are:

$$\text{PR} = \frac{\text{capacity} \times \text{irradiation received} \times \text{correction factors}}{\text{actual production}}$$

$$\text{EPI} = \frac{\text{capacity} \times \text{irradiation received} \times \text{temperature} \times \text{correction factors}}{\text{actual production}}$$

The correction factors are small coefficients that can be applied, depending on the plant characteristics.

These KPIs can be computed for various granularities. The sensor measurements often have a 15 minutes granularity, hence the granularity of the KPI start at 15 minutes.

One challenge with computing the KPIs is the (in)accuracy of the irradiation and temperature measurements. Irradiation is often measured using satellite data, which may have a margin of error of up to 20% for 15 minute data. This margin of error can be decreased at the cost of precision, as the margin of error decreases when the granularity increases with, e.g. up to 5% of error for monthly data. The impact of these inaccuracies can be observed in Figure 1. This figure displays the hourly PR for one inverter over 6 years. The KPIs are supposed to be in the range 0 to 100. However, due to the inaccuracies, PR can reach 120. In addition, the daily variation due to the weather is clearly visible in the KPI. Precisely assessing long-term degradation in such noisy signals is therefore challenging.

3 Literature review

Long term degradation is an important topic in many industrial domains. Assets, e.g. cars, wind turbines or assembly lines, will progressively see their performance decrease over time. This aging usually varies from one asset to another based on usage condition, environmental factors and other contextual aspects. In addition, this long-term degradation can easily be hidden by the short-term variation of the asset production, e.g. the daily variations due to weather for PV plants or the seasonality.

A simple statistical approach is to compute the mean production at various periods by only considering "perfect conditions", e.g. by only considering days with similar high amount of irradiation reaching the plant. However, this approach often fails in industrial settings due to the noisy aspect of the data.

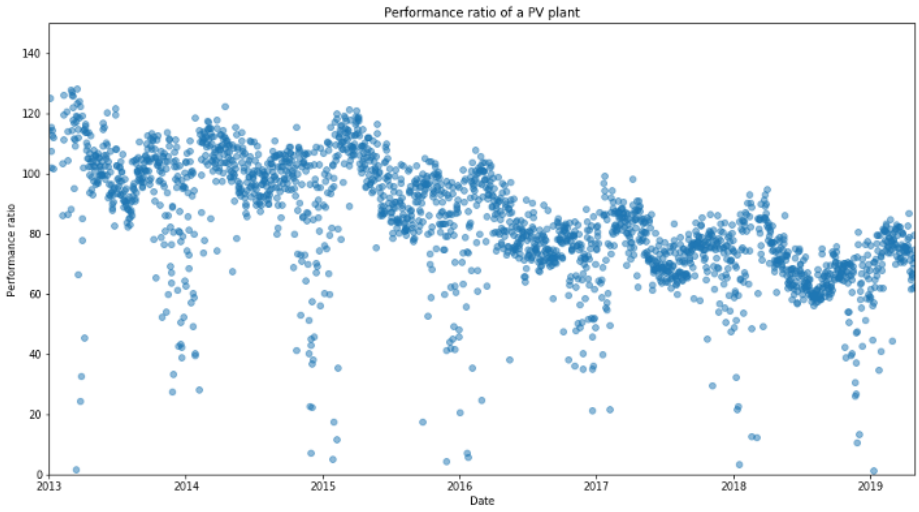


Fig. 1. Example of KPI: PR for one inverter over 6 years

3.1 Methods from the PV domain

There are three types of methods traditionally used for long-term degradation detection in the PV domain [2]. All these methods try to extract the degradation from the KPI signal using different approaches:

- Fitting a linear regression in the signal
- Decomposing the signal into trend, seasonal component and noise
- Comparing the evolution of the signal over specific periods

Methods that fit a linear regression in the signal These methodologies try to model the relationship between the asset performance and time. Most methodologies use the least square approach to do so. This approach consists of creating a model minimizing the sum of squared residuals, a residual being the difference between an observed value, and the fitted value provided by the model. Conceptually, the problem can be seen as fitting a line in the plot of the KPI over time that minimizes the distance between the line and all the data points of the plot. Two methods are usually used [6]:

- Ordinary least square (OLS) regression. OLS is one of the most used least square approaches. Geometrically, OLS tries to minimize the sum of the squared distances, parallel to the axis of the dependent variable, between each data point in the set and the corresponding point on the regression surface.
- Quantile regression. Whereas the method of least squares relies on the mean, quantile regression relies on the median (or other quantiles). In our setting,

the main advantage of quantile regression is that it is more robust against outliers, i.e., it will be less impacted by days with failures or abnormal behavior.

These methods are simple to implement, have a very low computation time and typically are able to deal with noise if enough data are provided. However, they often struggle with seasonality.

Methods that decompose the signal into trend, seasonal component and noise A widespread method to decompose time series data into 3 components containing seasonality, trend and residual is STL[1]. STL is an acronym for "Seasonal and Trend decomposition using Loess", while Loess is a method for estimating nonlinear relationships. The seasonal component will contain all the seasonal patterns, e.g. the lower performance of a PV plant in winter due to the cold climate. The trend component will contain the long-term pattern, e.g. the aging of the plant. The residual component will contain the noise, i.e., the remaining signal after having removed the seasonal and trend components. For long-term degradation purposes, only the trend component is relevant. STL allows to extract any seasonality from the trend and is therefore more suited for assets experiencing any kind of seasonality, such as PV plants.

Methods that compare the evolution of the signal over specific periods Another widespread approach to detect long-term degradation is to directly compare the performance over specific time windows. One flavor of these methods is the XbX methods[4], for instance the day-by-day method or the month-by-month method. All these methods are conceptually similar. For instance, the day-by-day method compares the performance of one day with the performance of the same day of the previous month. Similarly, month-by-month compares the performance of one month with the one of the same month of the previous year.

The intuition behind the XbX method is to hide seasonality by using a time window of the length of that seasonality. For instance, if there is a monthly seasonality, day-by-day will compare the same days between two months, while if there is a yearly seasonality, it will compare days between two years. By comparing periods that experience the same seasonal effect, the impact of that effect is mitigated. The PV domain suffers from yearly seasonality with lower performances in winter. Therefore, the Year-by-Year (YoY) method should be used.

However, an additional issue is the impact of failures that are present in the dataset. For instance, the same day of the previous year may have seen the occurrence of a shutdown of the asset. Therefore, the performance would wrongly appear to have significantly increased. However, the method expects that by considering larger periods, e.g. 365 days, the impact of the faults would be smoothed. The shutdown in the previous year would be mirrored by a shutdown in the second year during another day (while there was no shutdown that day in the previous year). Hence, by aggregating all these XbX differences and

considering the median of these differences, the method should output a representative estimate of the difference in terms of performance. The long-term degradation is then the mean difference between all the same days of all successive years (or any time window used).

Jia et al. [5] have proposed a new flavor of this approach. It relies on the assumption that the asset production reflects a linear correlation with a main feature, i.e., the parameter strongly impacting the asset production/performance (for example, wind speed for turbines and irradiation for PV plants). This relationship is often called the power-curve. When applying PCA, the 1st principal component will be that production output curve. Then, the 2nd principal component will be a measure of the dispersion of the production output curve, i.e., the relation between the production and the main feature. Hence, the standard deviation (STD) of the 2nd principal component will quantify how wide or narrow is the dispersion of the production output curve. It can be used to detect if the behavior of the asset has changed over time.

The process can be visualized in Figure 2. It displays the power-curve of one inverter over 5 years (with different colors per years). The first and second PCA component of the last year are shown with the red lines. A difference of narrowness between each year is clearly visible in that figure and Jia's method is dedicated to quantifying it.

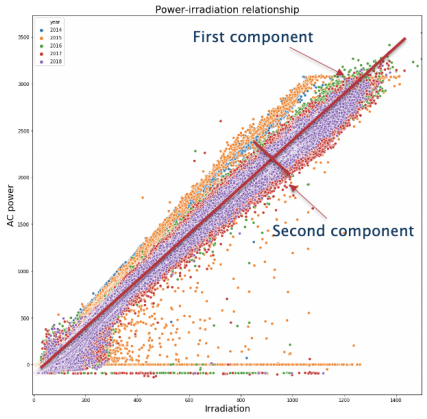


Fig. 2. Power-curve of one inverter with the first and second PCA component of the last year (each year is indicated by a different color)

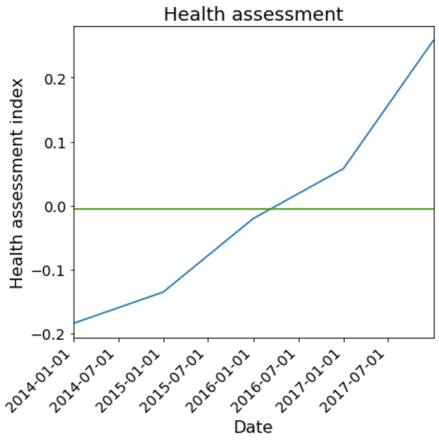


Fig. 3. HAI of one inverter over 4 years

The second component can then be used to compute a health assessment index (HAI) by comparing the evolution of the STD over time. The STD of each year is compared to the STD of the first year, used as a baseline as it is expected to have no degradation. The HAI is computed using the formula:

$$\text{HAI} = \frac{STD(2_{nd} \text{ PCA of the target period})}{STD(2_{nd} \text{ PCA of the baseline period})} - 1$$

Figure 3 shows the HAI computed for one inverter over 4 years. The HAI range from -0.2 to 0.2. The seasonality issues are then addressed as the production is always considered in regard to the amount of irradiation (the seasonality is actually influenced by the irradiation as it is lower in winter).

However, Jia's method has two important drawbacks. First, the STD only indicates if the performance of the asset changed but does not indicate if it decreased or increased. Second, it does not indicate in which proportion the asset's production has changed. For instance, domain experts cannot easily decide if an HAI of 0.2 requires to replace the asset or if they could wait to have an HAI of 0.3 as they do not know the relation between the HAI and the actual degradation. Therefore, this methodology has limited applicability in industrial context.

3.2 Methods from other domains

Other methods have been developed in other domains but have never been adapted and evaluated in the PV domain. In [9], Ulanova et al. formulated the problem of degradation detection as a Quadratic Programming (QP), which can be solved with existing QP-solvers. They were able to detect long-term degradation on chemical plants and in financial data.

Liu et al. [7] developed a model-free data analytic methodology for long-term degradation of gas turbines based on a dedicated performance index (specific to the gas turbine domain).

In [8], Sipos et al. used a totally different methodology and relied on the events generated by the assets. They associated a severity score to the events and used it to detect if an asset starts to degrade by analyzing its overall score. However, they do not quantify this degradation as they do not analyze time series of sensor data.

4 Methodology

We propose a novel approach inspired by Jia's method where, instead of relying on the PCA, the angle between the linear regression and one of the axis is evaluated. The axis used (x in our case) is the axis where the data points would be in case of non performing assets, e.g. when AC power is always zero, regardless of the irradiation, for the PV domain. The intuition behind our method is that, with the degradation, the power given for a specific amount of irradiation will decrease. Hence, the production output curve will get closer to the x axis. By analyzing the angle with the x axis and using the first period of production, e.g. year, of the asset as a baseline, it is possible to infer the percentage of performance degradation.

Our methodology follows three steps: 1) Data cleaning, 2) Data segmentation by meaningful periods and 3) Degradation quantification.

4.1 Data Cleaning

A correct degradation measure can only be obtained if the noise in the data has been removed. This cleaning may be domain specific, e.g. for the PV domain, a frequent cleaning is to remove the data points linked to a low irradiation, i.e., the data points in the morning and evening when the measures of the irradiation are more difficult and can lead to higher errors. Therefore, this data cleaning step may need to be refined for some application domain. For other application domains a cleaning based on removing the data points outside three standard deviations may be preferred.

4.2 Data segmentation

The method relies on an analysis of the evolution of the angle between the linear regression and the x axis over multiple periods. Therefore, the definition of the periods, e.g. weeks, months or years. is an important factor that will impact the accuracy of the method. The periods need to be large enough so that there is sufficient time for a potential degradation to manifest itself and significantly impact the asset performances, e.g. comparing on week to week basis is meaningless as e.g. degradation of 0.01% cannot be detected in the data (which often has higher measure error margin than such low degradation).

On the other hand, if the period is too large, the method results may lack granularities. For instance, for a 4 years old asset, if the period considered is one year, i.e., the first year is compared to the second year, then the second to the third and the third to the fourth, only three degradation measures will be available. Depending on the application domain, such low granularity may not be sufficient. Maintenance teams may need finer granularity, e.g. an estimate of the degradation every 6 months if the asset may age quickly e.g. for assets in harsh environments.

Therefore, the definition of the periods will be a trade-off between the granularity needed by the maintenance teams to more optimally perform the maintenance and the minimal duration needed to notice the impact of the degradation in the data. This step is heavily impacted by the application domain and should therefore be decided by domain experts.

4.3 Degradation quantification

Based on the cleaned data and the period length previously defined, the degradation can then be computed. A period is used as reference, e.g. the first month or year of performance or any other period where the asset is expected to perform at its maximum. The intuition is to have an example of perfect behaviour of the asset that could be used as a baseline to compare the evolution of the asset performance over time.

Then, the angle between the x axis and the linear regression is computed for any other period, e.g. for each month or year, the linear regression of the data points of that period and the angle to the x axis are computed. It is showcased

in Figure 4. The figure depicts the production output curve of one inverter (each year is indicated by a different color) and the computation of the angle for the last period (the data points in purple).

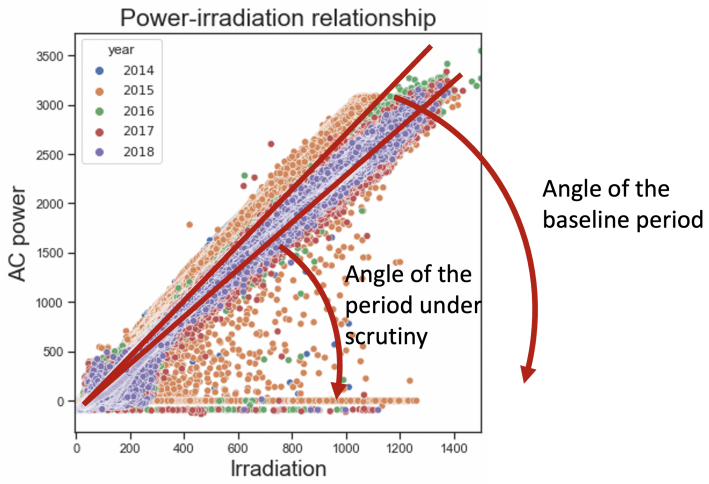


Fig. 4. Production output of one inverter (each year is indicated by a different color) and the computation of the angle for the last period

By comparing the angle of each period to the baseline, it is possible to evaluate a percentage of degradation for the asset over these periods. The baseline is used as a base 100 and the x axis as zero. Given α as the baseline angle and β as the angle of the period under scrutiny, then the degradation is simply:

$$-(100 - 100/(\alpha * \beta))$$

5 Validation and benchmarking

The method has been implemented and benchmarked in the PV domain against the state-of-the art methods (of the PV domain). The benchmarking has been realized using six real-life PV plants for which four years of data are available. The plants have been extracted from the Australian PV institute free dataset[1]. Unfortunately, there is no ground truth data about the long term degradation.

The cleaning has been made by removing data points outside 2 standard deviation. This aggressive data cleaning has been used as some of the plants in our dataset are particularly noisy. For the state-of-the-art methods, the data are cleaned by removing all data points where the irradiation was below 600kw/m2, as usually done in the PV domain.

[1] https://pv-map.apvi.org.au/performance#4/-28.77/134.91

Two aspects have been evaluated: 1) the accuracy of our methodology is compared to the other state-of-the-art methods used in the PV domain in terms of accuracy of the long-term degradation measured 2) the impact of the period length on the methodology is evaluated. The accuracy of our method on shorter datasets is explored to determine if our method is able to already detect long term degradations after e.g. 9 months or 12 months. We expect that our methodology should be superior to the state-of-the art methods on that aspect due to its design.

Accuracy benchmarking The 4 state-of-the art methods used in the benchmarking are: YoY, STL, OLS and quantile regression. There is an important difference between these methods and our methodology (labelled as angular methodology). All these methodologies intend to detect the trend in the PR or EPI data while the angular methodology is based on measuring the change of behavior of the relation between the AC power and the irradiation.

One challenge is the absence of ground truth as the exact degradation is not known. Therefore, it is impossible to assess which method is closer to the actual degradation. However, in this benchmarking, we intend to show that our methodology provide an estimation of the degradation which is close to the degradation detected by the other methods. Hence, the benchmarking intends to show that the angular methodology has a similar efficiency than the methods that have been shown as successful in the PV domain. The main interest of our methodology is its ability to detect degradations over shorter periods, which will be evaluated in the next section.

Figure 5 shows the degradation derived by the angular methodology and the state-of-the-art ones when applied on the EPI signal (left plot) or on the PR signal (right plot). Note that the angular methodology estimates the same degradation in both plots as it does not rely on these signals. The long-term degradations are computed at the end of the dataset, i.e., using the 4 years of data. Hence, the degradation represents how much efficiency has been lost after 4 years. The plants are on the x axis, the measured long-term degradation is indicated on the y axis and each method is indicated by a different color. For instance, on the first plant (wxzsjaf), all methodologies detect a degradation between -1% and -1.5% (in the left plot).

It appears from the plots that the angular methodology has very similar results than (at least some of) the other methodologies for the 6 plants. There are 3 plants (t3pg1sv, wca0c5m and z0aygry) where the methodologies do not agree with differences of around 1.5% of degradation between the various measures. The largest difference between the methodologies occur for plant t3pg1sv. OLS, Quantreg and STL, when applied on the PR signal, measure the same degradation as the angular methodology (around 0%) while YoY measure around -1% of degradation. However, OLS, Quantreg and STL, when applied on EPI signal, measure the same degradation as YoY. Unfortunately, without ground truth the evaluation of the methodologies cannot be further explored. Nevertheless, this benchmarking shows that the angular methodology tends to detect the lowest

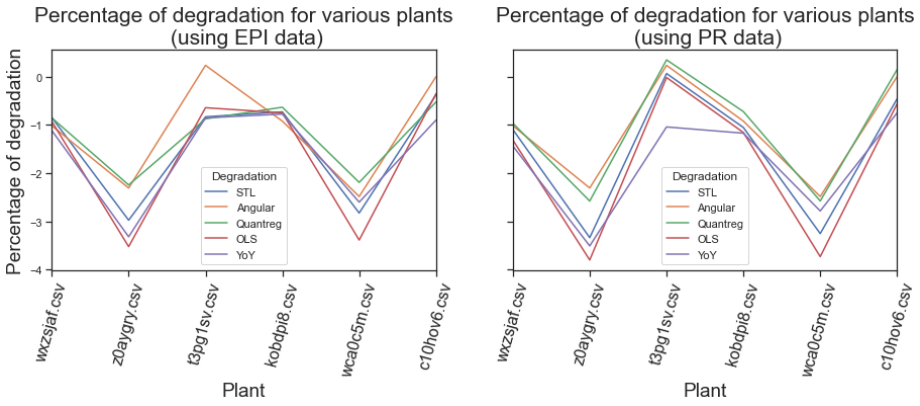


Fig. 5. Benchmarking of the methods using the EPI (left) and PR (right) signals

degradation but often estimates the same degradation than the quantile regression methodology for all plants.

5.1 Period length evaluation

The second aspect evaluated is the accuracy of the angular methodology when applied to smaller periods. An early detection of long term degradation can still be relevant for assets experiencing harsh environment (such as high thermal amplitude) or to evaluate whether the asset has been installed correctly, which if not the case could create an important strain on the asset. The previous benchmarking has been made on plants for which 4 years of data were available. Therefore, the state-of-the-art methodologies, using trend-detection methods, had plenty of data to accurately detect the trends. However, these methods are typically not well perfroming when less data points are available. Therefore, their accuracies could be impacted, especially for noisy plants, while we expect that our methodology would keep its accuracy.

To visualize that aspect, we can focus on a noisy plant, wca0c5m, and compare the degradation measured after various time periods. Figure 6 shows the results. For each method, the degradation is first computed after 3 months of exploitation, i.e., using the first 3 months of data. The degradation is then computed every 3 months until the end of the first three years of exploitation. Note that the YoY method is not used as it requires at least 2 years of data to measure the degradation. The x axis indicates the amount of months used to compute the long-term degradation and the y axis indicates the long-term degradation value. The left plot relies on the EPI, while the right one relies on the PR.

It clearly appears (on this extreme case) that the angular methodology is far more stable and has a far more realistic estimation of the degradation over these three years. The degradation ranges between +1% to -2% while the degradation measured by the other methods go up to - 35% after 6 months of exploitation

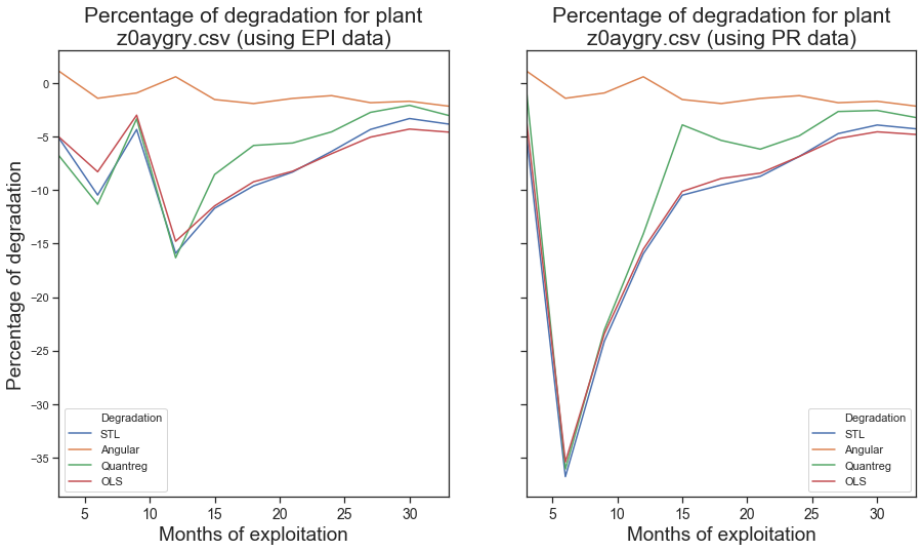


Fig. 6. Benchmarking of the method for plant wca0c5m

(using the PR). The angular methodology seems therefore more robust to noisy "shorter" datasets. The other methods were not able to detect early degradation on that plant. Moreover, if these methods would have been included in an automated O&M tool detecting and aggregating various losses, the -35% of long-term degradation would probably have triggered many false alarms.

The distribution of the variation of the degradation measured can be further explored for all plants. The degradation measured after 3 years is considered as the correct degradation (as all methods agree). The difference between that correct degradation measurement and each 3-monthly degradation measurements is then computed. These differences represent the error of measurement of the methodologies. The distribution of these differences is then displayed in Figure 7. The x axis indicates the method used while the y axis indicates the degradation error. The left plot relies on the EPI while the right one relies on the PR.

It clearly appears that the angular methodology has far less variability than the others, especially when they rely on the PR. The other methodologies have similar distribution for their degradation errors. It highlights the fact that these methods rely on the same assumption, i.e., the degradation is reflected by the trend, while our method is different in nature.

Figure 7 also highlights the fact that two very noisy plants (Z0aygry and wca0c5m) have the widest distribution for the state-of-the-art methodologies. Only one plant (kobdpi8) has an identical distribution of degradation error for all methodologies.

This analysis showcases that the angular methodology seems more efficient to detect degradation after a shorter period of exploitation. Our analysis has shown

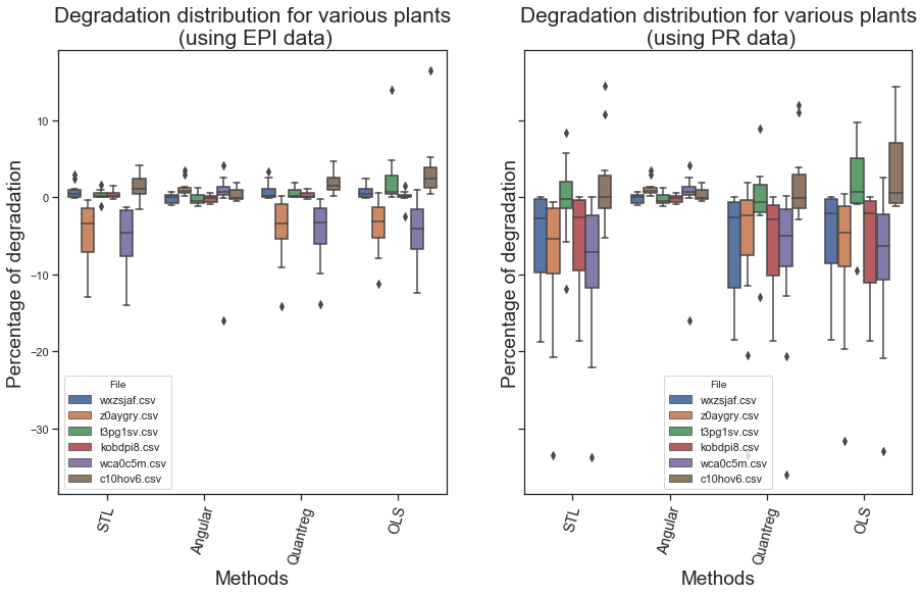


Fig. 7. Distribution of the measurement errors for all time periods

that for noisy plants the state-of-the-art methodologies may need around 2 years of data to accurately detect the degradation while the angular methodology can better estimate it far earlier.

We suppose that the detection of the trend is heavily sensitive to noise and seasonality, especially with incomplete datasets. For instance, applying trend detection on a dataset of one year and a half with one winter and two summers may create biases.

The angular methodology is far less impacted by that aspect. The methodology can afford a heavier cleaning (removing any data points outside 2 standard deviation) as it only requires enough data points to reflect the linear relationship between AC power and Irradiance. Therefore, it significantly reduces the impact of noise. In addition, the seasonality does not impact the angular method as it does not impact the linear relationship between AC power and Irradiance while it impacts the PR or EPI (to a lower extend). Hence, the summer or winter periods can easily be compared without impact on the methodology accuracy.

In the absence of ground truth, the angular methodology cannot be further evaluated. However, this benchmarking has shown that the angular methodology has a similar accuracy as the state-of-the-art methodologies but requires less data points to correctly detect the degradation. Note that the angular methodology has only been applied on PV plants, i.e. electronic devices without moving parts, were the long-term degradation is slow and steady. Many other industrial assets may have a different type of long-term degradation as the stress on the moving parts creates more and more degradation over time. However, we expect that the

angular methodology will perform as well on those types of assets as the steady aspect of the degradation do not play any role in the methodology. This intuition is also based on the efficiency of Jia et al. methodology on wind turbines, i.e, assets with moving parts. Our angular methodology is relying on the same basis thus the efficiency should be similar.

The only constraint on the angular methodology that it is applicable in contexts where an asset is mainly impacted by only one feature. For instance, inverters are mainly impacted by irradiation and wind turbines are mainly impacted by wind speed. Therefore, the methodology should perform well on any type of assets for which there is a production output curve.

6 Conclusion

In this paper, we have presented a novel methodology for long-term degradation detection. Our methodology is based on the evaluation of the evolution of the asset production curve over time. We have realised this methodology for the PV domain, and we have benchmarked it with the current state-of-the-art methodologies, relying on trend detection, using real-life datasets. We have shown that our methodology allows to detect similar degradation than the state-of-the-art methodologies. Moreover, our methodology was able to do so with less data, e.g. 3 months of data, while the other methods needed up to 2 years of data before detecting realistic long-term degradations. Therefore, our methodology is more suited for industrial contexts where accurate measurements are needed early on.

In terms of future work, we envision:

- Evaluating the angular methodology with real-life PV plants were the actual degradation is known. In the absence of ground truth, it was unfortunately impossible to precisely assess which of the methods had the most accurate measure of the degradation and we were only able to evaluate if all methods agreed on the degradation. Real-life datasets would allow to better benchmark the methods. Unfortunately, long term degradation is difficult to measure and the use of synthetic dataset has many shortcomings such as the difficulty to generate representative noise or degradation.
- Evaluating the accuracy of the angular methodology in other application domains. The methodology should work in any domain where the asset's production is heavily impacted by only one factor. Therefore, it could also be applied, e.g. in the wind turbine domain where the AC power is mainly impacted by the wind speed.
- Further explore the data cleaning step. Our data cleaning step (removing data points outside 2 standard deviation) has been demonstrated to be efficient, as the angular methodology seems more robust to noise than the state-of-the art methodologies. However, this step is currently quite straightforward and more advanced cleaning could be explored. Especially, cleaning strategies dedicated to highlight the linear relationship between the production and the main factor could be explored.

Acknowledgements

This work was subsidised by the Region of Bruxelles-Capitale - Innoviris.

References

1. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: Stl: A seasonal-trend decomposition. Journal of official statistics **6**(1), 3–73 (1990)
2. Curran, A.J., Jones, C.B., Lindig, S., Stein, J., Moser, D., French, R.H.: Performance loss rate consistency and uncertainty across multiple methods and filtering criteria. In: 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC). pp. 1328–1334. IEEE (2019)
3. Gueymard, C.A., Myers, D.R.: Evaluation of conventional and high-performance routine solar radiation measurements for improved solar resource, climatological trends, and radiative modeling. Solar Energy **83**(2), 171–185 (2009)
4. Hasselbrink, E., Anderson, M., Defreitas, Z., Mikofski, M., Shen, Y.C., Caldwell, S., Terao, A., Kavulak, D., Campeau, Z., DeGraaff, D.: Validation of the pvlife model using 3 million module-years of live site data. In: 2013 IEEE 39th Photovoltaic Specialists Conference (PVSC). pp. 0007–0012. IEEE (2013)
5. Jia, X., Jin, C., Buzza, M., Wang, W., Lee, J.: Wind turbine performance degradation assessment based on a novel similarity metric for machine performance curves. Renewable Energy **99**, 1191–1201 (Dec 2016)
6. Kim, W., He, T., Wang, D., Cao, C., Liang, S.: Assessment of long-term sensor radiometric degradation using time series analysis. IEEE Transactions on Geoscience and Remote Sensing **52**(5), 2960–2976 (2013)
7. Liu, Y., Banerjee, A., Hanachi, H., Kumar, A.: Data-Driven Model Selection Study for Long-Term Performance Deterioration of Gas Turbines. In: 2019 IEEE International Conference on Prognostics and Health Management (ICPHM). pp. 1–8. IEEE (2019)
8. Sipos, R., Fradkin, D., Moerchen, F., Wang, Z.: Log-based predictive maintenance. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1867–1876. ACM (2014)
9. Ulanova, L., Yan, T., Chen, H., Jiang, G., Keogh, E., Zhang, K.: Efficient Long-Term Degradation Profiling in Time Series for Complex Physical Systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15. pp. 2167–2176. ACM Press, Sydney, NSW, Australia (2015)

Detecting Anomalous Invoice Line Items in the Legal Case Lifecycle

Valentino Constantinou[0000−0002−5279−4143]
Mori Kabiri[0000−0002−5745−0948]

InfiniGlobe, LLC, Newport Beach, CA 92660, USA
Mori.Kabiri@infiniglobe.com

Abstract. The United States is the largest distributor of legal services in the world, representing a $437 billion market [3]. Of this, corporate legal departments pay law firms $80 billion for their services [4]. Every month, legal departments receive and process invoices from these law firms and legal service providers. Legal invoice review is and has been a pain point for corporate legal department leaders. Complex and intricate, legal invoices often contain several hundred line-items that account for anything from tasks such as hands-on legal work to expenses such as copying, meals, and travel. The man-hours and scrutiny involved in the invoice review process can be overwhelming. Even with common safeguards in place, such as established billing guidelines, experienced invoice reviewers (typically highly paid in-house attorneys), and rule-based electronic billing tools ("e-billing"), many discrepancies go undetected. Using machine learning, our goal is to demonstrate the current flaws of, and to explore improvements to, the legal invoice review process for invoices submitted by law firms to their corporate clients. In this work, we detail our approach, applying several machine learning model architectures, for detecting anomalous invoice line-items based on their suitability in the legal case's lifecycle (modeled using a set of case-level and invoice line-item-level features). We illustrate our approach, which works in the absence of labeled data, by utilizing a combination of subject matter expertise ("SME") and synthetic data generation for model training. We characterize our method's performance using a set of model architectures. We demonstrate how this process advances solving anomaly detection problems, specifically when the characteristics of the anomalies are well known, and offer lessons learned from applying our approach to real-world data.

Keywords: Deviation and Novelty Detection · Legal Informatics · Time Series · Anomaly Detection · Law Firm Invoice · Line-Items

1 Introduction

Most legal invoices are intricate, containing hundreds of line-items with detailed information about dates of service, quantity, rate, billing codes, and task descriptions. Generally, a line item could be a task or expense. The task or expense is

identified by a standardized billing code that reflects the type of work done by the "timekeeper", who is the law firm personnel who performed the task and reported the time.

Date	Type	Timekeeper	Qty.	Rate	Amt.	Code	Actibity	Description
13-Jan-20	Fee	JOHN DOE	0.3	$250	$75	L120	A104	REVIEW ANSWER TO COMPLAINT FILED BY J. BROWN AND FOLLOW-UP
14-Jan-20	Fee	JOHN DOE	1.5	$250	$375	L190	A103	REVISE AND FINALIZE MOTION FOR DEFAULT JUDGMENT AGAINST J. GREEN.
16-Jan-20	Expense	JANE DOE	1	$65	$65	E110		UBER - TRANSPORTATION TO/FROM AIRPORT - 16-JAN-20
...	...	...	...	...	...	...	...	...

Fig. 1. An example of line-items contained within legal invoices.

In the late 1990's, *e-billing* (applying a set of standardized codes to electronically-submitted invoices in an attempt to simplify and standardize the legal billing process and facilitate the ease of review and approval) was developed. Over the past two decades, most legal departments have shifted from paper invoices to e-billing tools to help validate invoices against their agreed terms with law firms and vendors, called "billing guidelines".

Based on an InfiniGlobe-led panel discussion with legal industry executives, most participants reported that of their law firms, 80% maintained "good compliance with billing guidelines" and 20% exhibited "poor billing hygiene". Although the majority of law firms consistently submit accurate invoices based on actual work done, some submitted invoices contain irregularities, intentional or unintentional. Common issues are improper codes, block billing, insufficient or vague descriptions, and math errors when indicating cost shares (sometimes in the firm's favor). Undetected discrepancies can lead to a variety of issues such as over- or under-payment by the corporation, extra manual administration for both the law firm and corporation, lost attorney time trying to rectify errors, and damaged relationships between the firm and its client.

While e-billing tools have helped with identifying some invoice issues, most of the rules employed in these systems are both too crude to recognize nuances and too quick to throw alerts. Often, in-house attorneys' time is wasted reviewing minor alerts from systems that trigger a significant number of false positive alerts while simultaneously failing to catch bigger issues such as overbilling (false negatives).

The barrage of alerts generated by e-billing tools can quickly become burdensome for invoice reviewers and most switch back to manual review. The problem is, however, that reviewing an invoice often requires deep knowledge of context, the law involved, the history of the case, and familiarity with key parties to verify that the tasks performed and billed were necessary and appropriate. This means that final approvers may need to be high-rate attorneys or management in charge of the case. The effort required combined with the high volume of invoices results in either lost productivity due to the significant time and effort spent on manual review, or overpayment due to rushed invoice approval without proper review.

In this paper, we employ anomaly detection techniques tuned for law firm invoices, mitigating some of the challenges mentioned to reduce the corporate legal department effort needed for review and approval. We describe our use of SME to generate synthetic anomalies which are representative of an anomaly type with high importance of detection (anomalies in the legal case lifecycle). Once a dataset containing representative synthetic anomalies is generated, we apply several well-known and readily available model architectures to this task. Methods for evaluating the performance of these models are detailed and utilized for reporting results. We then present experimental results using real-world data from invoice line-items created by law firm billing customers for their services. While this work is presented through its application to the legal industry, it may be applied more generally to other similar types of data and in use cases where data patterns of the anomalies are well-known.

2 Background and Related Work

2.1 Global and Local Anomalies

The significant depth and breadth of anomaly detection research offers many anomaly types. With regard to detecting anomalies in line-items - which often contain significant amounts of categorical data - it is useful to consider two categories of anomalies: global and local [30]. *Global anomalies* exhibit unusual or rare individual attribute values (across the set of available features). These types of anomalies typically relate to rare attributes [14] and, when flagged, are often the source of significant false positive alerts [30]. *Local anomalies* exhibit unusual or rare combinations of attribute values while the individual attribute values themselves occur more frequently. These types of anomalies - while more difficult to detect - can be tied to behaviors typical of those conducting fraud [30, 32]. We use these characterizations throughout this work.

2.2 Supervised and Unsupervised Modeling

Supervised classification models have been used in health care fraud detection [34] and other industries such as computer security, telecommunications, etc. [30, 31] due to their ability to detect specific fraud schemes. A major advantage of supervised learning is that the labels used for model training can be easily interpreted by humans for discriminative pattern analysis.

Supervised learning models, however, can't always be applied. First, it can be prohibitively expensive or difficult for many organizations to obtained labeled training data [7, 29]. Second, even labeled data may contain ambiguities which stem from loose labeling guidelines or varying interpretations between data labelers. This results in less robust model decision boundaries. Additionally, new fraud schemes, differing in distribution from normal data and considered anomalies, are not immediately detectable due to the lag between discovering and subsequently labeling data as representing fraud [34].

For this reason, research attention has also focused on unsupervised approaches for anomaly detection [7,12,14,29]. Unsupervised approaches overcome the labeling challenge, however, they have challenges related to the interpretability of results and ease of application. Unsupervised approach results will always require subject matter expert interpretation, negating some of their benefits. Additionally, many different types of anomalies may exist within a dataset, which makes interpretation more onerous. The need to interpret the results of unsupervised approaches provides the potential for mischaracterizations, such as considering many anomalous clusters to be a single anomalous group. This contrasts with supervised approaches, where models are typically trained to identify a single type of anomalous behavior.

2.3 Electronic Invoicing in the Legal Industry

Instead of mailing paper invoices, most law departments now require law firms to submit billing information in electronic format using e-billing systems and according to corporate billing guidelines.

In order to standardize the categorization of legal work and expenses and facilitate billing analysis, the Uniform Task-Based Management System (UTBMS) codes [1] were developed in the mid1990s [6] through a collaborative effort by the American Bar Association Section of Litigation and the American Corporate Counsel Association, among others. UTBMS is a series of codes used to classify legal services performed by a legal vendor in an electronic invoice submission. Attorneys record their time and classify tasks and expenses using the appropriate code from the UTBMS code set. UTBMS codes can be useful for both law firm and legal department data analytics purposes, for example when reviewing how much time has been spent on each type of activity (e.g. the percent of time spent on L200 Pre-Trail Pleadings and Motions vs. L400 Trail Preparation and Trail). In more advanced analyses, UTBMS codes can reveal the ordering of the activities throughout a legal case, such as showing the typical distribution of work (recorded by time) given a legal case's attributes. An individual line-item can be assessed as to its suitability in the distribution of work (which defines the typical legal case lifecycle).

UTBMS codes offer many more benefits, however, there are some challenges. First, the use of UTBMS codes is inconsistent and largely dependent on each law firm's individual timekeepers (e.g. attorneys, paralegals). Due to the sheer number of codes for different tasks in different phases on different case types, it is simply too difficult to memorize and utilize them effectively. Additionally, timekeepers may unintentionally assign incorrect codes because they are either too busy to search for the correct code or are indifferent to how these codes may benefit them or their clients. And in certain cases, some may intentionally bill fees or expenses with alternate codes to bypass e-billing alerts. When corporate legal departments do not recognize or acknowledge the negative impact of incorrect billing practices by their law firms, it may encourage timekeepers to continue ignoring or not following billing guidelines. As a result, on the corporate side, billing and vendor management departments are left with muddled data

L100	**Case Assessment, Development and Administration**	L320	Document Production	**A100**	**Activities**
L110	Fact Investigation/Development	L330	Depositions	A101	Plan and prepare for
L120	Analysis/Strategy	L340	Expert Discovery	A102	Research
L130	Experts/Consultants	L350	Discovery Motions	A103	Draft/revise
L140	Document/File Management	L390	Other Discovery	A104	Review/analyze
L150	Budgeting	**L400**	**Trial Preparation and Trial**	A105	Communicate (in firm)
L160	Settlement/Non-Binding ADR	L410	Fact Witnesses	A106	Communicate (with client)
L190	Other Case Assessment, Development and Admin...	L420	Expert Witnesses	A107	Communicate (other outside counsel)
L200	**Pre-Trial Pleadings and Motions**	L430	Written Motions and Submissions	A108	Communicate (other external)
L210	Pleading	L440	Other Trial Preparation and Support	A109	Appear for/attend
L220	Preliminary Injunctions/Provisional Remedies	L450	Trial and Hearing Attendance	A110	Manage data/files
L230	Court Mandated Conferences	L460	Post-Trial Motions and Submissions	A111	Other
L240	Dispositive Motions	L470	Enforcement	**E100**	**Expenses**
L250	Other Written Motions and Submissions	**L500**	**Appeal**	E101	Copying
L260	Class Action Certification and Notice	L510	Appellate Motions and Submissions	E102	Outside printing
L300	**Discovery**	L520	Appellate Briefs	E103	Word processing
L310	Written Discovery	L530	Oral Argument	...	...

Fig. 2. The above table shows available UTBMS codes for Litigation cases. The codes themselves are broken into phases, e.g. L100, L200, etc.

containing many incorrect UTBMS codes, making accurate analytics and reporting a significantly more difficult task and negating many of the afore-mentioned UTBMS system benefits.

3 Methods

The following methods form the core components of a supervised anomaly detection approach that isolates preferred modeling data through dimensionality reduction and clustering, generates synthetic (anomalous) data based on information provided by SMEs, and utilizes well-known and easily accessible models for anomaly detection according to the characteristics of legal case lifecycles.

3.1 Selecting Modeling Data

A large corporation will typically hire law firms to provide legal services in different practice areas for open cases. These firms will submit invoices each month with line-items for each service provided. We sampled invoices with these line-items for our experiments, specifically line-items from *Litigation* type cases (the most popular UTBMS case type). We applied a series of methods to select data that is suitable for the modeling process, i.e. data that would include the types of anomalies we were seeking to detect.

Anomalies of interest to our study include line-items which, given the set of categorical attributes for the line-item, fall outside of the typical distribution of services billed in the legal case lifecycle. Cases that do not utilize codes spread across typical case phases may be ill-defined in the context of lifecycle anomaly detection as there is limited billing information. Such cases are often small in scope or settled early in the case lifecycle and do not cycle through later stages. These cases cannot contain an anomalous invoice line-item with respect to the case lifecycle, thus we removed these observations from the training set.

We then applied a two-step approach utilizing dimensionality reduction and unsupervised clustering to isolate Litigation type legal cases which are suitable for the modeling process. More specifically, we used features (generated from UTBMS codes) that indicate the distribution of billing charges across case phases. *Phases* are higher-level categories defined in UTBMS codes (e.g. L100, L200, etc.). The following steps describe how the percentage of each phase charged per case is captured and processed.

We used Singular Value Decomposition (SVD) [20] and T-Distributed Stochastic Neighbor Embedding (T-SNE) [21] to reduce data dimensionality and visualize the charging patterns across cases. Subsequently we used Density-based Spatial Clustering of Applications with Noise (DBSCAN) [16] to group line-items according to related groups of charging patterns. Groups (clusters) containing Litigation type legal cases that exhibited well-distributed charging behaviors were retained for subsequent use in the experimental process, with "well-distributed" groups identified through a manual examination of the charging patterns (e.g. 60% or more of charge codes utilized). Cases contained within groups of suitable data were utilized in the remainder of the experiment. All other cases were removed from the analysis.

When applying SVD, we selected the number of principal components to use based on value, resulting in principal components that captured at least 95% of the variation in the data. When applying T-SNE, a sufficiently low number of components were specified to simplify the clustering task. This approach combines the effectiveness of SVD with the interpretability benefits provided by T-SNE. By utilizing DBSCAN, we based the clustering results on the minimum distance between observations to be considered a "new" cluster, without the need to specify the number of clusters, and free from any assumptions about cluster distributions used by common approaches such as K-Means [24,26,33] or Spectral Clustering [11,15,17].

3.2 Synthetic Anomaly Generation

Labeled training data was not available for our use case. To overcome this challenge, we utilized available SME to generate synthetic lifecycle anomalies in the line-item data which are representative of the types of anomalies we were seeking to detect. *Lifecycle anomalies* are defined as anomalies which, with the exception of the number of days since the start of the legal case, are otherwise normal. That is, the line-item is anomalous with respect to the time in which it is present in the life cycle of the legal case and not according to its attribute values. Since we suspected that a very small number of these anomalies exist in the dataset, we employed an approach which generates synthetic anomalies to provide a basis for training supervised models specific to this task, similar to work performed by Schreyer et al. [30].

First, we used the previously mentioned characterizations of global and local anomalies to identify global anomalies from the modeling dataset. Any combination of specified categorical feature (variable) values that occurs less than a specified number of times in the dataset can be considered rare occurrences, or

global anomalies. These line-items were ignored later in the process. Lifecycle anomalies - whose individual values are not rare alone, but which are rare in combination - are considered local anomalies, like our anomalies of interest.

Once the rare and unusual line-items were marked, we generated a specified number of anomalies by manipulating the feature indicating the number of days since the legal case was opened. Until the specified number of anomalies was generated, a randomly selected line-item was drawn from the modeling dataset and adjusted so that the only feature out of place (given the other features) was the number of days since the case opened (more specifically, minimum and mean values for the days since the case started is calculated). We then utilized those values to modify line-items whose combination of feature values never occurs too early or too late in the lifecycle. The number of days since the case started was then adjusted for the selected line-item, such that it occurs far earlier in the lifecycle than is typical for the line-item's combination of feature values. This results in a set of local anomalies, together with normal data, to utilize for model training and evaluation. These anomalies are considered local as the feature values themselves are common, but the number of days since the legal case started in combination with the feature values is considered out of place.

3.3 Model Training and Testing

We utilized Random Forest [10], Gradient Boosted Tree [9], and Support Vector Machine (SVM) [13] model architectures in our experiments as these architectures are widely available, well-understood, and relatively performant across a broad variety of use cases. Additionally, we sought to apply a set of varying model architectures to provide several points of comparison.

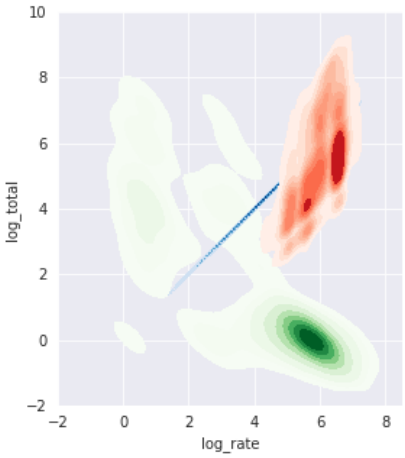


Fig. 3. The results of applying Gaussian Mixture Models (GMMs) for clustering of charging data.

Decision trees are widely employed given their simplicity, interpretability, and intelligibility. These models - using tree structures - represent class labels as leaves and conjunctions (combinations) of features that lead to those labels as branches. Decision tree modeling can however tend to overfit the data [19]. Random Forest Classifiers aim to overcome this tendency and reduce the variance in the predictions by constructing many decision trees (a "forest") and taking the mode (classification) or mean (regression) of the individual trees. These trees are fully grown and independently trained. Additionally, Random Forests provide a robust means to gauge the importance of features through approaches like Shapley Additive Explanations (SHAP) [25] - a useful

feedback mechanism which we used to guide feature selection [10]. Yet, Random Forests may not be as performant as Gradient Boosted Trees in some use cases, which aim to improve performance over decision trees in another way.

Gradient Boosted Trees ensemble a set of weak learners, or shallow trees, in a bid to improve performance through a reduction of bias. Boosted trees train each weak learner sequentially, with each learner aiming to improve the performance of the ensemble of trees. With boosting, the training of a learner in the ensemble is dependent on the learners trained prior (in contrast with Random Forests). Due to the differences in their approaches for reducing prediction error, we elected to use both architectures in the experiments.

For feature selection, we used a combination of SME knowledge and feature (variable) importance measures [25] to inform our final feature specification. We used both case-level and line-item-level attributes (encoded as features), as well as features which are derived through an earlier exploratory data analysis process. More specifically, we generated a feature which utilizes charging information (the hourly rate and the total) to distinguish groups of line-items by how they are billed (e.g. hourly or as a single invoice for the entire case) through applying a Gaussian Mixture Model (GMM) [8]. Additionally, we utilized the generated feature indicating the number of days since the legal case start (mentioned previously) but applied a log transformation to the feature before use in the models.

4 Experiments

For many anomaly detection systems, the performance of the selected approaches is difficult to assess. System assessments may be performed manually or through direct application to the use case. Fortunately, synthetic data representing anomalies of interest was generated and included in our experiments. Performance metrics can be directly computed by assigning the labels provided through synthetic data generation.

As described earlier, we used Litigation type cases in our experimental process. We conducted a grid search to identify the best possible parameters for each model and subsequently trained them. To provide a means of assessing performance, we classified each observation between two classes: *anomalous* and *normal*. Performance metrics were then calculated based on the consistency between the labels and the model's predictions.

4.1 Setup

For a set of data containing one or more anomalous invoice line-items, we evaluated the last 20% of the data - deemed a validation set - that is unseen by trained models. A model is trained for each model architecture according to a set of parameters identified as part of a grid search and a set of predictions is generated against the validation set. The performance of the model is assessed

according to the agreement between the labels and predicted values provided by each trained model.

For each model architecture, the predicted classes for each line-item in the validation set was evaluated against the labels provided by the synthetic data generation according to the following rules:

1. A **true positive** is recorded if the true label y is equal to the predicted value $\hat{y}$.
2. A **false negative** is recorded if the true label y is anomalous but the predicted value $\hat{y}$ is estimated to be normal.
3. A **false positive** is recorded if the true label y is normal but the predicted value $\hat{y}$ is estimated to be anomalous.

We utilized the Python programming language for our experiments due to the wide variety of open source libraries available dedicated to machine learning and its general popularity with data scientists [5]. Scikit Learn [28] was utilized for dimensionality reduction, clustering, and model training. The libraries Matplotlib [22], seaborn [36], and Plotly [23] were utilized for visualization, with Pandas [27], Numpy [18], and Scipy [35] also being utilized in our experimental pipeline.

4.2 Model Parameters and Evaluation

1,657 legal cases were represented in the originating dataset, with 973 Litigation type cases. The number of principal components for use in SVD was specified as 5, capturing approximately 97.91% of the total amount of variation in the data. When applying T-SNE, the number of components was specified as 2, with the perplexity, learning rate, number of iterations, and the embedding initialization specified as 25, 50, 10000, and Principal Components Analysis (PCA), respectively. When applying DBSCAN, the maximum distance between two samples was set to 4.5 and the minimum number of samples in a neighborhood for a point to be considered a core point was set to 10.

A threshold must be set that identifies global anomalies in a dataset from all others. When generating the synthetic anomalies used for model training, we utilized a threshold of 25, with any combination of feature values occurring less than the threshold considered to be global anomalies. We utilized a total of 5 features for this task related to the case category and line-item UTBMS code, type, and other information. In order to provide a suitable dataset for model training, we specified that 5% of the newly-generated dataset contain local lifecycle anomalies.

For each model architecture, we utilized a common set of both categorical and numerical features that provide both case-level and line-item attributes. Some features included in the modeling process were the line-item-level UTBMS code, the activity, and the days since the case was opened. Case-level features included the case category and the number of unique UTBMS codes utilized. We evaluated the models according to the following performance metrics:

1. **Precision:** The number of true positives tp over the number of false positives fp and true positives tp, $\frac{tp}{tp+fp}$.
2. **Recall:** The number of true positives tp over the number of false negatives fn and true positives tp, $\frac{tp}{fn+fp}$.
3. **F1 Score:** The harmonic mean of the precision and recall, defined as $2 * \frac{precision*recall}{precision+recall}$.
4. **Accuracy:** The number of true positives tp over the total number of samples in the validation set n.
5. **Coverage:** The number of samples which fall above a specified prediction confidence threshold (set to 90%) over the total number of samples in the validation set n.

The best-performing set of parameters for each model were identified and selected through a grid search to maximize the F1 score. The best set of parameters for each model architecture were recorded and applied to subsequent steps.

The modeling data is split into training, testing, and validation sets representing 60%, 20%, and 20% of the data, respectively. More specifically, the training set represents the first 60% of values in the data, with the validation set covering the last 20%. Once the data was split into three distinct sets, we used time series cross-validation to train the model on the training data and assess its performance on the test data before generating our final set of predictions for evaluation on the validation set. We used the set of performance metrics defined above to evaluate and report the performance of each model.

4.3 Results and Discussion

The data selection process returned a total of 340 legal cases for experimentation. With the parameter settings for data selection set as previously defined, a total of 5 clusters (groups) of legal cases were identified (see Table 5 detailing the number of observations in each cluster and Figure 4 for a graphical representation). Once the clusters of legal cases were identified based on their charging patterns, cluster 3 was selected as suitable for the modeling process based on a manual verification of the charging patterns for the cases in the cluster. This cluster provided a set of cases whose charging behavior is well distributed across the case lifecycle based on manual inspection of the line-items in the cluster.

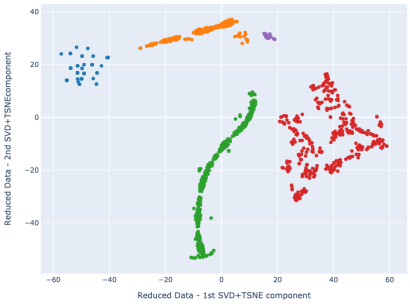


Fig. 4. Cluster 3, shown in red (farthest right), was identified as suitable for modeling.

The synthetic data generation process provided a dataset for model training and evaluation that included 1,967 anomalous line-items, with 77,305 line-items

considered "normal". Figure 6 illustrates the distributions of the log-transformed number of days since case start variable between the anomalous (class 1) and normal classes (class 0) in the dataset. As would be the case in a real-world scenario, the number of days since the case start values are significantly lower for line-items in the anomalous class versus those in the normal class. This follows the intuition that line-items which are anomalous with respect to the legal case lifecycle will exhibit regular (expected) feature values and be out of place only with respect to the chronology of when in the lifecycle they are charged. There is a clear and distinct separation between the two classes of observations (line-items) however, which can be easily detected by most machine learning algorithms. In future work, it may be suitable to explore a synthetic data generation strategy which results in a less distinct separation between the anomalous and normal classes. Models performing robustly under that scenario are likely to perform better overall.

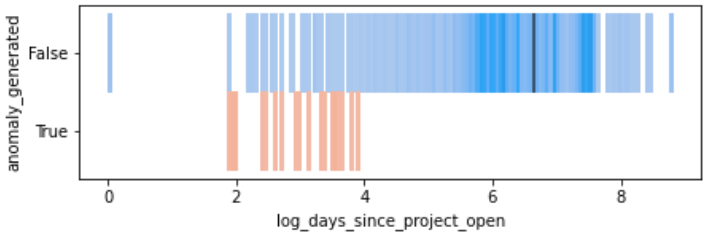


Fig. 6. The distribution of the log transform of the number of days since the case was opened across both non-anomalous (raw) data and anomalous (synthetic) data. The two distributions differ significantly.

43,230 observations (line-items) were utilized in model training, with 14,410 and 14,411 observations belonging to the test and validation sets, respectively. As mentioned, time series cross-validation was used for model training and testing. This approach was selected due to its suitability in the context of how these models would be applied.

Cluster	Observations
0	294
1	107
2	218
3	340

Fig. 5. The number of observations assigned to each cluster.

Specifically, models would be periodically retrained on historical data and then employed for a specified amount of time to predict the classes of any new line-items in the system. Any data available up until the model is trained would be included as part of a pipeline, similar to the experimental process outlined in this paper. After that, features included in the model training process would be generated for any incoming line-items. These line-items would then be assessed for their anomalousness based on the earlier data.

Validation Metrics on the Anomalous Class			
Metric Name	Random Forest	Gradient Boosted Tree	Support Vector Machine
Precision	91.16%	100%	74.35%
Recall	73.28%	56.28%	98.30%
F1-Score	83.17%	72.12%	**84.67%**
Accuracy	97.82%	96.79%	97.38%
Coverage	97.29%	95.41%	93.76%

Table 1. A comparison between the performance of various model architectures on the experimental data. The best-performing model was used for each model architecture, identified through a grid search of possible parameters. It is important to recall that only 5% of the data represents anomalies and that the F1-Score should be the most heavily considered metric.

The performance metrics related to each model are shown in Table 1. Random Forest and SVM show comparable performance when assessed on the F1-Score yet differ in behavior. SVM provides more false positives but captures more anomalies (high recall, low precision) while Random Forest returns a lower number of false positives at the expense of false negatives. SVM resulted in the highest overall F1-score at 84.67% and also outperformed the other models in regards to recall (98.30%). Gradient Boosted Tree showed the highest precision (100%). While the tree-based methods were expected to perform relatively well based on their ability to capture non-linear relationships, their performance when compared to SVM is comparable and perhaps complementary if used in an ensemble. A model architecture could be selected by weighing the trade-off between higher false positive or false negative rates according to the specific anomaly detection problem. A high false positive rate could have significant adverse effects on end-user satisfaction [2].

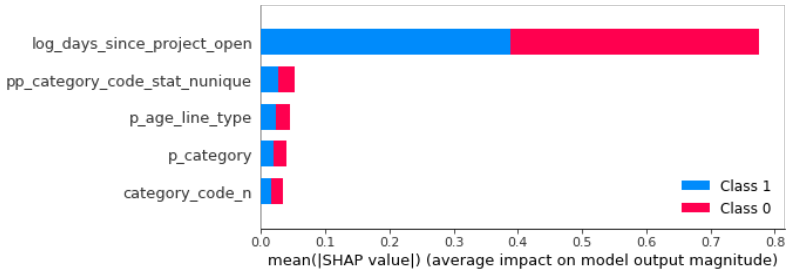


Fig. 7. The Shapley Additive Explanations (SHAP) values from the Random Forest model for the top five most highly-weighted features. Note that the feature importances are weighed evenly across both the anomalous (class 1) and normal (class 0) classes.

We believe that there is value for future work in exploring additional feature engineering and model architectures which may yield improved performance beyond the results in this work, including exploring deep learning models for tabular data that utilize categorical feature embeddings. Feature selection was driven through a combination of SME and interpretation of Shapley Additive Expla-

nations (SHAP) values [25] provided as a result of modeling the Random Forest and Gradient Boosted Tree models. The feature importance values provided the feedback necessary to refine our earlier work and arrive at the feature set utilized in this paper. The top five feature importance values from the trained Random Forest model are shown in Figure 7. It is no surprise that the number of days since the case opened is weighed heavily by the model, as that is how this specific anomaly type (a line-item out of place in the legal case lifecycle) is defined.

5 Conclusion

We provide an intelligent anomaly detection solution which works to address the challenges legal professionals face because of basic e-billing systems and misuse of UTBMS codes, raising costs and affecting overall efficiency. First, we highlight our approach to selecting data that is suitable for the experimental process and for lifecycle-type anomalies utilizing dimensionality reduction and clustering. Then, we provide a mechanism for applying supervised learning to anomaly detection problems in the legal industry when the behavior of potential anomalies is well-known. We highlight our approach towards generating a labeled dataset through manipulation of existing data in alignment with the concept of a local anomaly, providing a basis for training supervised models such as Random Forest, Gradient Boosted Tree, and SVM. The performance metrics for each of these model types against the synthetic data generation are provided and compared. An interpretation of the results and how they would inform application in the legal industry is then offered. By selecting a model architecture which minimizes false positives, this work could help further the flagging capabilities of e-billing software by minimizing erroneous alerts which reduce productivity. This work can improve the performance of automated alerts for e-billing systems in the legal industry.

6 Acknowledgements

This effort was supported by InfiniGlobe LLC located in Newport Beach, California, USA. The authors would like to thank Macon Mclean, Mike Russell (Expedia Group), Bobby Jahanbani (Exterro), Julie Richer (American Electric Power), Monika Schreiner, and Dawn Kabiri for their feedback and support.

References

1. Litigation code set, https://www.americanbar.org/groups/litigation/resources/uniform_task_based_management_system/litigation_code_set
2. A survey of in house attorneys, https://reducespending.legalbillreview.com/a-survey-of-in-house-attorneys-pw/executive-summary/

3. How big is the us legal services market? (2016), http://www.legalexecutiveinstitute.com/wp-content/uploads/2016/01/How-Big-is-the-US-Legal-Services-Market.pdf
4. 2019 global legal department benchmarking report (06 2019), https://www.acc.com/sites/default/files/2019-06/ACC_Benchmark_062019.pdf
5. 2019 python developer survey (2019), https://www.jetbrains.com/lp/python-developers-survey-2019/
6. Uniform task based management system (2020), https://utbms.com/
7. Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system: A survey. Journal of Network and Computer Applications **68**, 90 – 113 (2016). https://doi.org/https://doi.org/10.1016/j.jnca.2016.04.007, http://www.sciencedirect.com/science/article/pii/S1084804516300571
8. Amendola, C., Faugere, J.C., Sturmfels, B.: Moment varieties of gaussian mixtures. Journal of Algebraic Statistics **7**(1) (Jul 2016). https://doi.org/10.18409/jas.v7i1.42, https://doi.org/10.18409/jas.v7i1.42
9. Breiman, L.: Arcing the edge (1997)
10. Breiman, L.: Machine Learning **45**(1), 5–32 (2001). https://doi.org/10.1023/a:1010933404324, https://doi.org/10.1023/a:1010933404324
11. Cheeger, J.: A lower bound for the smallest eigenvalue of the laplacian (1969)
12. Chen, T., Tang, L., Sun, Y., Chen, Z., Zhang, K.: Entity embedding-based anomaly detection for heterogeneous categorical events. CoRR **abs/1608.07502** (2016), http://arxiv.org/abs/1608.07502
13. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (Sep 1995). https://doi.org/10.1007/bf00994018, https://doi.org/10.1007/bf00994018
14. Das, K., Schneider, J.: Detecting anomalous records in categorical datasets. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 07. ACM Press (2007). https://doi.org/10.1145/1281192.1281219, https://doi.org/10.1145/1281192.1281219
15. Donath, W., Hoffman, A.: Algorithms for partitioning graphs and computer logic based on eigenvectors of connection matrices. IBM Technical Disclosure Bulletin **15**(3), 938–944 (1972)
16. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. pp. 226–231. AAAI Press (1996)
17. Fiedler, M.: Algebraic connectivity of graphs. Czechoslovak Mathematical Journal **23**(2), 298–305 (1973). https://doi.org/10.21136/cmj.1973.101168, https://doi.org/10.21136/cmj.1973.101168
18. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del R'ıo, J.F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
19. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc., New York, NY, USA (2001)

20. Hestenes, M.R.: Inversion of matrices by biorthogonalization and related results. Journal of the Society for Industrial and Applied Mathematics **6**(1), 51–90 (Mar 1958). https://doi.org/10.1137/0106005, https://doi.org/10.1137/0106005
21. Hinton, G., Roweis, S.: Stochastic neighbor embedding. In: Proceedings of the 15th International Conference on Neural Information Processing Systems. p. 857–864. NIPS'02, MIT Press, Cambridge, MA, USA (2002)
22. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
23. Inc., P.T.: Collaborative data science (2015), https://plot.ly
24. Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489
25. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
26. MacQueen, J.: Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967). (1967)
27. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference. pp. 56 – 61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a
28. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
29. de Roux, D., Perez, B., Moreno, A., del Pilar Villamil, M., Figueroa, C.: Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM (Jul 2018). https://doi.org/10.1145/3219819.3219878, https://doi.org/10.1145/3219819.3219878
30. Schreyer, M., Sattarov, T., Borth, D., Dengel, A., Reimer, B.: Detection of anomalies in large scale accounting data using deep autoencoder networks. CoRR **abs/1709.05254** (2017), http://arxiv.org/abs/1709.05254
31. Schultz, M., Tropmann-Frick, M.: Autoencoder neural networks versus external auditors: Detecting unusual journal entries in financial statement audits. In: Proceedings of the 53rd Hawaii International Conference on System Sciences. Hawaii International Conference on System Sciences (2020). https://doi.org/10.24251/hicss.2020.666, https://doi.org/10.24251/hicss.2020.666
32. Sparrow, M.K., Chiles, L.: License to Steal. Routledge (Mar 2019). https://doi.org/10.4324/9780429039577, https://doi.org/10.4324/9780429039577
33. Steinhaus, H.: Sur la division des corps matériels en parties. Bull. Acad. Pol. Sci., Cl. III **4**, 801–804 (1957)
34. Travaille, P., Mueller, R., Thornton, D., Hillegersberg, J.: Electronic fraud detection in the u.s. medicaid healthcare program: Lessons learned from other industries. (01 2011)

35. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, l., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S...: Scipy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods (2020)
36. Waskom, M., the seaborn development team: mwaskom/seaborn (Sep 2020). https://doi.org/10.5281/zenodo.592845, `https://doi.org/10.5281/zenodo.592845`

Rymon Trees and Binary Data Biclusters

Joshua Yee, Haoyu Wang, and Dan A. Simovici

University of Massachusetts Boston, Boston MA 02126, USA
Dan.Simovici@umb.edu

Abstract. We present a combinatorial approach to the identification of large biclusters in binary data that makes use of Rymon trees, a technique for systematic enumeration of subsets of sets. We introduce the notions of pre-clusters and bicluster using a polarities defined by binary relations. Experimental work that demostrates the performance of our approach accompanies our algorithm.

Keywords: Rymon trees · biclusters · polarity

1 Introduction

The DNA Microarray technology allows the simultaneous analysis of relationships between groups of genes. Informally, a *bicluster* is a subset of genes that show similar activity patterns under a subset of conditions.

The study of biclusters was initiated with Hartigan's work [5], in which the way of dividing a matrix in submatrices with the minimum variance was studied. In that approach the perfect bicluster is a submatrix formed by constant values, i.e., with variance equal to zero. Hartigan's algorithm, named *direct clustering*, divides the data matrix into a certain number of biclusters, with the minimum variance value, so the fact of finding a number of sub-matrices equal to the number of elements of the matrix is avoided.

An alternative technique for finding biclusters is to measure the coherence between their genes and conditions. Biclustering based on local nearness developed by Cheng and Church [3] introduced a measure, the mean squared residue (MSR), that computes the similarity among the expression values within the bicluster. The ideas of Cheng and Church were further developed by Yang [14, 13] who dealt with missing values in the matrices. As a result of this approach an algorithm named FLOC was designed.

Other works [12] are based in a quality value as well, calculated using the expression values of biclusters, so to measure their coherence. Alternatives in the searching for biclusters have been studied in the last years. We might also consider that a value in the data matrix is the sum of the contributions of different biclusters. Lazzeroni et al. [7] introduced the *plaid model*, in which the data matrix is described as a linear function of layers corresponding to its biclusters and shows how to estimate a model using an iterative maximization process.

Other authors [11] proposes a new method to obtain biclusters based on a combination of graph theoretical and statistical modelling of data. In this model, a gene responds to a condition if its expression level changes significantly at that condition witch respect to its normal level.

In a more recent work [8], a generalization of OPSM model, introduced in [1], is presented. The OPSM model is based on the search of biclusters in which a subset of genes induce a similar linear ordering along a subset of conditions. Some techniques search for specific structures in data matrix to find biclusters. In [6] method for clustering genes and conditions simultaneously based on the search of checkerboard patterns in matrices of gene expression data is proposed. Previously, the data is processed by normalization in a spectral framework model (several schemes all built around the idea of putting the genes on the same scale so that they have the same average level of expression across conditions, and the same for the conditions). Evolutionary computation techniques have also been used in this research area. These techniques use aspects from natural selection within computer science, including genetic algorithms, genetic programming and evolutionary strategies.

In [4] an evolutionary technique, based on the search of biclusters following a sequential covering strategy and measuring the mean squared residue, is used.

We adopt the binary model of data considered in [9]. This model has the advantage of simplicity and avoids dealing with complex optimization problems, which for most part and NP-complete.

We introduce a method of producing biclusters suitable for use on genome data with large sets of genes and limited number of columns. The performance of the algorithm is evaluated on the size of the produced bicluster and the time required to find it, with size referring to the number of elements within the bicluster. Our algorithm will produce locally maximal (in size) biclusters based on a user determined parameter referred to as a threshold.

Note that a brute-force approach could be used to find the largest possible bicluster present in a binary matrix A. One could consider every possible choice of column groupings and the resulting biclusters, noting the size of each result. The largest bicluster would then be the maximal bicluster within A. However the number of column groups grows exponentially in the number of columns of A, thus such an approach is generally unsuitable for most real world data sets.

We begin by presenting Rymon trees (introduced in [10]) in Section 2 where we formally define biclusters, and propose and algorithm for producing biclusters from binary matrices. The experimental results of the algorithm are discussed in Section 3. An extension of the initial algorithm designed for increased performance on larger matrices is presented in Section 4. Finally, in Section 5 we summarize the observations from both algorithms and suggest avenues for future development of this approach.

2 Rymon Trees, Polarities, and Biclusters

The tree of sets approach (TSA) makes use of a Rymon tree [10], which is a systematic way of enumerating the subsets of a finite set S.

Let $\mathcal{P}(S)$ be the set of subsets of a set S. We assume that a total order relation "$<$" exists on the elements of S. The *Rymon tree* for a finite set S is a tree $\mathcal{T}(S)$ whose vertices are the subsets in $\mathcal{P}(S)$. The root of three is the empty set $\emptyset$. If U is a node in $\mathcal{T}(S)$, the descendents of U are the sets of the form $U \cup \{i\}$, where $i > \max U$.

For $S = \{1, 2, 3, 4\}$ the Rymon Tree is shown in Figure 2.

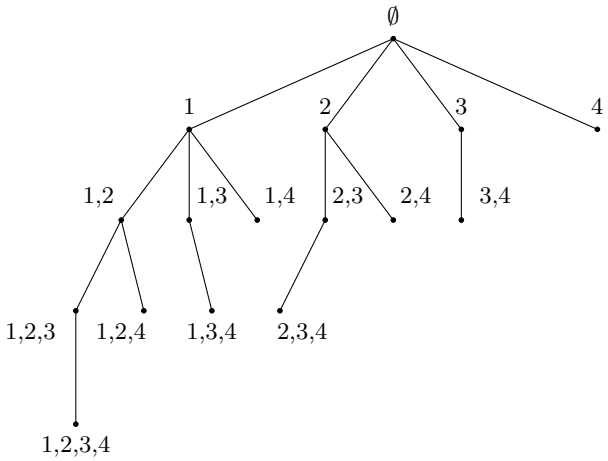


Fig. 1: Rymon Tree on Set $\{1, 2, 3, 4\}$.

Let R be the set of rows of the matrix $A \in \{0, 1\}^{m \times n}$ and let C be its set of columns. Define the relation $\theta \subseteq R \times C$ as

$$\theta = \{(r, c) \mid a_{rc} = 1\}.$$

The mappings $\kappa : \mathcal{P}(R) \longrightarrow \mathcal{P}(C)$ and $\rho : \mathcal{P}(C) \longrightarrow \mathcal{P}(R)$ are defined by

$$\kappa(P) = \{c \in C \mid a_{rc} = 1 \text{ for all } r \in P\},$$
$$\rho(K) = \{r \in R \mid a_{rc} = 1 \text{ for all } c \in K\},$$

for any set of rows P and any set of columns K. The pair (κ, ρ) is a *polarity* [2] defined by the relation θ.

The following properties of the pair of mappings (κ, ρ) are immediate consequences of their definition:

1. if $P_1 \subseteq P_2$, then $\kappa(P_2) \subseteq \kappa(P_1)$,
2. if $K_1 \subseteq K_2$, then $\rho(K_2) \subseteq \rho(K_1)$,

3. $K \subseteq \kappa(\rho(K))$ and $P \subseteq \rho(\kappa(P))$,

for all sets of rows P_1, P_2, P and all sets of columns K_1, K_2 and K.

Also, we have: $\rho(K) = \rho(\kappa(\rho(K)))$ and $\kappa(P) = \kappa(\rho(\kappa(P)))$.

In general, in the data sets that we consider in this paper, the number of rows outnumber by far the number of columns. Therefore, the notions that we are about to introduce have an asymmetric character, in the sense that they begin with sets of columns.

Definition 1. *A* pre-bicluster *in the matrix* $A \in \{0,1\}^{m \times n}$ *is a pair of sets* (P, K)*, where* $P = \rho(K)$*, where* P *is a set of rows and* K *is a set of columns.*

A pair of sets of the form (P, K) *is a* bicluster *if* $P = \rho(K)$ *and* $K = \kappa(P)$*.*

Example 1. For the data set

	C_1	C_2	C_3	C_4
r_1	1	0	1	1
r_2	1	0	1	0
r_3	1	0	0	1
r_4	1	0	0	0
r_5	1	1	1	1
r_6	1	1	1	0
r_7	1	1	0	1
r_8	1	1	0	1

	C_1	C_2	C_3	C_4
r_9	1	1	1	0
r_{10}	0	1	1	1
r_{11}	0	1	0	0
r_{12}	0	1	1	1
r_{13}	0	0	1	0
r_{14}	0	0	1	1
r_{15}	0	0	0	1
r_{16}	0	0	0	1

the Rymon tree annotated with the size of the corresponding pre-biclusters is shown next:

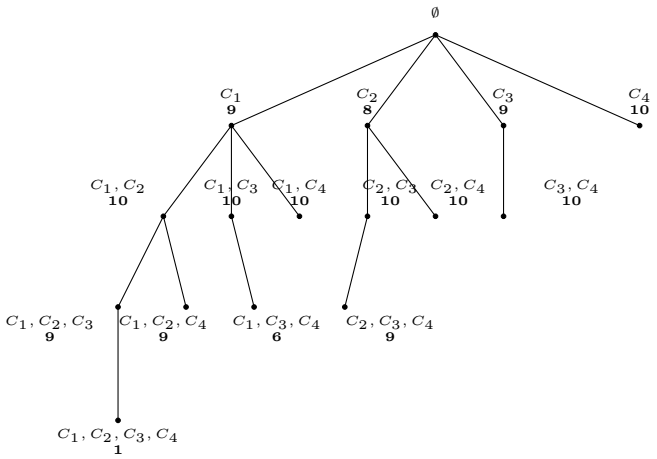


Fig. 2: Rymon Tree on Set $\{1, 2, 3, 4\}$.

Theorem 1. *Every pre-bicluster* (P, K) *is included in a bicluster.*

Proof: Since (P, K) be a pre-bicluster, we have $P = \rho(K)$. Therefore, $\kappa(P) = \kappa(\rho(K)) \supseteq K$. Thus, if $K_1 = \kappa(P)$ we have $K \subseteq K_1$.

We claim that (P, K_1) is a cluster. Obviously, $K_1 = \kappa(P)$; also, $\rho(K_1) = \rho(\kappa(P)) = \rho(\kappa(\rho(K)) = \rho(K) = P$, which shows that (P, K_1) is a bicluster that contains the pre-bicluster (P, K). This concludes the proof.

Note that in figure 2, each pre-bicluster is already a bicluster.

The algorithm accepts a parameter t representing the desired number of elements in the final bicluster result. We define t as a fraction α of the expected number of 1s in matrix A, $s(A) = mnp$, where $p \in [0, 1]$ is the probability that an entry of the matrix A equals 1, that is, $t = \alpha s(A)$. We used $\alpha = 0.1$ in our experiments.

The algorithm functions by examining a set of columns K and its corresponding pre-bicluster (P, K) as follows: first compute the largest bicluster (P, K') containing (P, K). If $|\rho(K)| \cdot |K'| \geqslant t$, then the pair $(\rho(K), K')$ is returned. Otherwise, examine all supersets L of K with $|L| = |K| + 1$, that is all descendants of K in the Rymon tree. For any such subset we have $\rho(L) \geqslant \rho(K)$, so $|\rho(L)| \times |L|$ may increase and, thus, exceed the threshold. The algorithm initially examines the root of the Rymon tree, amounting to a depth first search with early termination on the tree.

Note that while the algorithm examines pre-biclusters, it always checks the size of the corresponding bicluster against the threshold t. This allows the algorithm to potentially skip nodes within the tree to further reduce the total number of nodes that must be expanded and to improve the size of a pre-bicluster that itself exceeds t in size.

Example 2. Consider the matrix A whose Rymon tree is explored with threshold $t = 6$:

$$\begin{pmatrix} 1\,0\,1\,1\,0 \\ 1\,0\,1\,1\,0 \\ 1\,1\,0\,0\,0 \\ 0\,1\,0\,1\,0 \end{pmatrix}$$

Because of the particular choice of threshold, if the algorithm only considers the size of pre-biclusters against t then it would have to traverse a minimum of four nodes of the Rymon tree before finding the only suitable bicluster consisting of columns 1, 3, 4 and rows 1, 2. The actual number of nodes may be higher depending on the order in which the algorithm explores the descendants of a given node. Note that when the algorithm considers the pre-bicluster for columns 1 and 3 together, the corresponding row set consists of rows 1, 2. By instead considering the bicluster corresponding to this column group, consisting of columns 1, 3, and 4, we arrive at the only viable bicluster in a minimum of three node expansions.

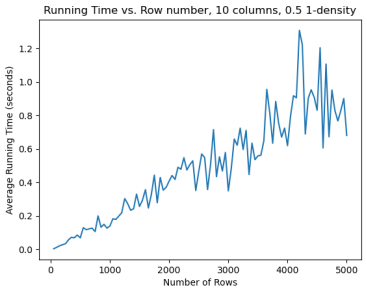
A formal description of the algorithm is as follows:

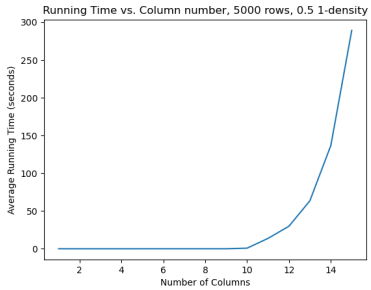
Algorithm 1: Tree Method

Input: A threshold integer t and a stack S representing the rymon tree for a binary matrix A, initially holding the root of the tree
Output: Lists of rows and columns from A specifying a bicluster
while S *is not empty* **do**
  $K = \text{pop}(S)$;
  $P = \rho(K)$;
  $K' = \kappa(P)$;
  $size = \text{sizeof}(P, K')$;
  if $size \geqslant t$ **then**
    return (P, K');
  else
    foreach *child* c *of* K **do** $S.\text{push}(c)$;
  end
end

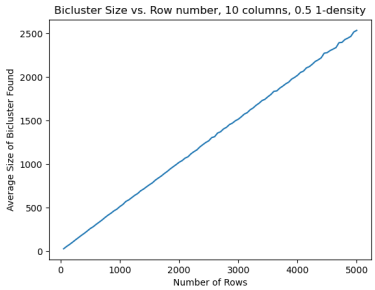
3 Experimental Results

To examine the performance of our algorithm, three parameters were considered: the number of rows, the number of columns, and the 1-density defined as the probability that an entry in a binary matrix is 1. Each parameter was varied in isolation with a number of trials at each parameter value. In each trial, the algorithm is applied to a randomly generated binary matrix A of size n by m with 1-density p using the threshold $t = \alpha nmp$ with $\alpha = .1$. The produced bicluster size and algorithm running time are both recorded. At the end of each set of trials, the average running time and results size are recorded and plotted against the parameter value. The results of the experiments are summarized in the following graphs, all with 10 trials for each parameter value:



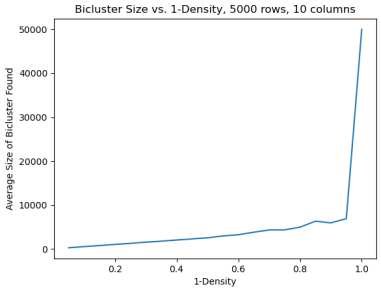


(a) Tree completion time vs. number of columns

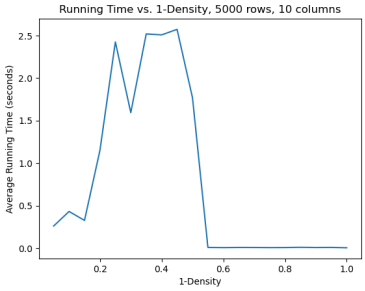


(b) Size of trees vs. number of rows

Fig. 3: Dependency of Rymon Trees on density.



(a) Bicluster size vs. density.



(b) Construction Time vs. density

Fig. 4: Dependeny of Biclusters on density.

4 An Extension of the Proposed Algorithm

Given the results above, a natural evolution of the tree algorithm is to modify it to work efficiently on matrices with large numbers of columns. In order to accomplish this, we develop a process consisting of three steps. First, the columns are clustered into reasonably small groups, with each group consisting of columns whose row values are similar. Then, the tree algorithm is performed on each group of columns in order to obtain biclusters for each column group. Finally, each of the biclusters are combined in such a way as to produce a single large bicluster with high one-density. Formal descriptions of these steps follow below.

For a binary matrix A, with column and row sets represented by C and R. Let $c \in C$ be a column, then define $c_r = a_{rc}$. We define a distance $d : C \times C- > \mathbf{R}$ by $d(c1, c2) = \sum_{r \in R} |c1_r - c2_r|$. Using this distance we perform hierarchical clustering on C yielding a clustering of columns $C_N = \{C_1, C_2, \ldots, C_n\}$ where each C_i is a column grouping produced by hierarchical clustering. The tree algorithm is then run on each $C_i \in C_N$ to obtain a set of biclusters $B_N = \{B_1, B_2, \ldots, B_n\}$ where each bicluster consists of a set of rows and columns i.e. $B_i = \{P_i, K_i\}$.

We obtain a set of biclusters B_N, and because the tree method produces pure biclusters, each $B_i \in B_N$ consists only of 1s. The next step is to combine the B_i in such a way as to maintain a high concentration of 1s in the final bicluster result. To do so, we define the column group for the final bicluster $K_T = \cup K_i$, the union of all columns from the B_i. Then define a treshold $0 \leqslant \tau \leqslant 1$, and then define the final row set as

$$K_T = \{r \in R || \{B_i | r \in K_i\}| >= \tau * |B_N|\}.$$

In other words, we include a particular row in K_T if that row is present in a sufficient number of the individual biclusters. By choosing a sufficently large τ, we can ensure a higher one-density in the final bicluster $B_T = \{P_T, K_T\}$ as the inclusion requirement for a row is more strict. Conversely, low values of τ allow for larger final biclusters at the cost of one-density.

To verify the effectiveness of the algorithm modifications on matrices with high column number, we run a number of experiments noting the average running times, hierarchical clustering cut heights, result sizes, and result one-densities of the multi-column algorithm while varying a number of parameters. Averages were taken across ten trials for each set of parameters, running the algorithm on randomly generated binary matrices with each entry being one with probability .6. The threshold used in the biclustering of the individual column groups is the same as used in the previous set of experiments. The results are summarized in the table below:

rows	columns	τ	max cols. per group	cluster size	density	time (s)
5000	50	.667	5	8559	.78	25.44
5000	50	.8	5	626	.87	25.26
5000	50	.5	5	58495	.67	14.71
5000	100	.667	5	4837	.76	117.60
5000	150	.667	5	1102	.76	264.05
5000	50	.667	7	6082	.81	12.62
5000	50	.667	10	5925	.83	5.55

Let $A \in \{0,1\}^{n \times m}$ be a data matrix. In the worst case, the algorithm visits each node of the Rymon tree. For a node representing a column subset of size k, the algorithm must iterate through each column in the subset and examine each row to determine which of the rows should be included in the current column subset's pre-bicluster. Thus for a column subset of size k, the algorithm goes through nk iterations. The algorithm then performs the completion step on the resulting pre-bicluster, which runs on the $m-k$ remaining columns, and at worst examines all n rows of the matrix to see if the new columns should be included. To both examine the column subset and perform completion on it, the algorithm undergoes nm iterations in total.

There are $\binom{m}{k}$ subsets of size k in the Rymon tree. To explore the entirety of the Rymon tree, the necessary number of iterations is given by $\sum_{k=0}^{m} \binom{m}{k}(nm) = nm \sum_{k=0}^{m} \binom{m}{k} = nm2^m$. The worst case occurs when the algorithm explores the entirety of the Rymon tree without finding a suitably large bicluster, and where every row is included in every column subset's pre-bicluster, in which case the algorithm has running time $\Theta(nm2^m)$. In practical terms, the algorithm terminates long before exploring the entire Rymon tree depending on the choice of the threshold parameter t.

5 Conclusions and Future Work

The tree algorithm result size increases linearly with row number, and the running time increases approximately linearly. Note that the variation in the values is caused by the variation in the portion of the Rymon tree that the algorithm must explore before finding a result. The algorithm could stumble on a viable column group quickly in a lucky run and slowly in another. Its running time appears to increase exponentially in the number of columns. Both density results follow intuition; when there are more 1s present in the matrix the algorithm is able to find larger results and do so faster. Of particular note is that the algorithm is able to function on matrices with 15 columns and 5000 rows in about five minutes, and scales linarly with row number. This means that the algorithm is well suited for gene expression data, which typically has large number of rows and relatively few columns.

The multi-column algorithm fulfills its purpose in allowing the tree algorithm to function better on matrices with larger numbers of columns. This algorithm

is able to produce results on matrices with over 150 columns in roughly four and a half minutes. Varying the τ parameter allows control over the size and one-density of the results, and altering the height at which the hierarchical column clustering is cut at allows for a trade-off between result size and speed. Of note, with sufficiently high cut height, the multi-column algorithm is able to produce results on a matrix with 50 columns in about 5.5 seconds with high one-density in the output.

The choice of the parameters of the algorithm will be further investigated. The threshold used here scales with the column number, row number, and 1-density of the binary matrix but it does so linearly in all respects. It may be that the sizes of biclusters do not scale linearly in all regards, such as in 1-density. Thus a different thresholding choice could allow for larger biclusters to be found.

References

1. Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the sixth annual international conference on Computational biology*, pages 49–57, 2002.
2. G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, RI, third edition, 1995.
3. Y. Cheng and G.M. Church. Biclustering of expression data. In *ISMB*, volume 8, pages 93–103, 2000.
4. Federico Divina and Jesus S Aguilar-Ruiz. Biclustering of expression data with evolutionary computation. *IEEE transactions on knowledge and data engineering*, 18(5):590–602, 2006.
5. John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.
6. Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.
7. Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica sinica*, pages 61–86, 2002.
8. J. Liu, J. Yang, and W. Wang. Biclustering in gene expression data by tendency. In *Proceedings. 2004 IEEE Computational Systems Bioinformatics Conference, 2004. CSB 2004.*, pages 182–193. IEEE, 2004.
9. A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.
10. Ron Rymon. Search through systematic set enumeration. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, USA, October 25-29, 1992*, pages 539–550. Morgan Kaufmann, 1992.
11. Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–S144, 2002.
12. H. Wang, W. Wang, J. Yang, and P. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 394–405, 2002.

13. J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Third IEEE Symposium on Bioinformatics and Bioengineering, 2003. Proceedings.*, pages 321–327. IEEE, 2003.
14. J. Yang, W. Wang, H. Wang, and P. Yu. Improving performance of bicluster discovery in a large data set. In *6th ACM International Conference on Research in Computational Molecular Biology, Poster*. Citeseer, 2002.

Two Strategies for the Classification of the Quality of Red Wine

Brian Benson, Warwick Graco, Hari Koesmarno, Ed Lewis, Garry Mitchell,

Tony Nolan, Peter Phillips

Analytics Shed, Urana Street, Jindera, New South Wales 2642, Australia
warwick.graco@analyticsshed.com

Abstract. This paper reports the results of research into methods for classification of the quality of red wine from its constituent chemicals. Two strategies are employed i.e. an unsupervised and supervised one. In terms of unsupervised learning, a taxometric method was applied to identify the chemicals that distinguish two classes in the red wine dataset to see if any of the chemicals are related to the quality of the wine. The scores for these chemicals were partitioned using normal mixtures clustering to see if clusters that were coupled to certain rating of the quality of the wines could be identified and to determine the chemicals that underly their quality. In terms of supervised learning, a 'Classification by Counting' (ClassC) algorithm was applied to the red wine dataset to identify those conjunctive and disjunctive rules for different pairs of chemicals that could predict the quality of wines. It was found that higher quantities of certain chemicals increase, while higher quantities of the remaining chemicals decrease, the ratings of the quality of the wines. It was also found that certain pairs of chemicals were highly predictive of the quality of the wines. The implications of the results and issues requiring further research are discussed in the paper.

Keywords: Red wine, Taxometric methods, Normal mixtures clustering, Classification, ClassC, Conjunctive rules, Disjunctive rules, Parallel coordinate plots

1 Introduction

There are several methods for identifying the composition of products and how the constituents determine their quality. This paper covers two strategies - unsupervised and supervised - for determining the chemicals that contribute to the quality of wine. There are two motivations behind this research.

The first motivation was to see if using unsupervised learning can identify the chemicals that contribute to the quality of the wines. It was expected that some chemicals will enhance, while other chemicals will diminish the quality of the wines.

This was performed by firstly, identifying the chemicals that define classes in a red wine dataset to see if any of the chemicals are related to the quality of the wine. It was carried out secondly by clustering the wines using the chemicals that define the classes

to see if there are clusters that are coupled with a specific rating of the quality of the wines. The mean chemical scores for the clusters that were coupled to a specific rating were examined to see if they revealed any chemicals that contribute to the quality of the wines. Parallel coordinate plots (see Figures 2(a) to 2(d) for examples) were employed to show wines having similar profiles and their couplings to rated wines.

The second motivation was to use a supervised learning by applying conjunctive and disjunctive rules to see if there are different pairs of chemicals that predict the quality of wine. An algorithm called 'Classification by Counting' (ClassC) was developed to identify these rules that classified high and low-quality wines.

The research reported in this paper is a work in progress. Some of the issues requiring further research are highlighted in the discussion section at the end of this paper.

The remaining parts of the paper cover the dataset used in the research, how class structure can be identified in data, the other analyses that were performed, and the results obtained. The paper concludes with a discussion of the implications of the research and issues requiring further attention.

This paper is primarily aimed at those who apply analytics to industry issues particularly those who work in the wine industry. The findings in this paper may also interest those who produce and those who consume wines.

In Section 2 we explain the data set. The identification of class structure in data using taxometric attribute analysis is described in Section 3. Section 4 explains the identification of clusters in the data while Section 5 describes the identification of different pairs of chemicals through classification by counting. Section 6 gives results. These results are discussed in Section 7. Finally, we give conclusions in Section 7.

2 Dataset

The red wine data set used to do the research was downloaded from the machine learning repository at the University of California Irvine campus. This dataset was used by Cortez et al [1] to identify the best regression model from those evaluated to predict the quality of wine.

The red wine set consisted of 1599 red wines and used eleven chemicals as input variables and had one output variable of the rated quality of the wine. The input variables are described in Table 1.

The quality of the wine was assessed by three wine tasters and was rated between 0 (poor) and 10 (excellent). Details of this dataset, including the statistics of the variables listed in Table 1, are found in Cortez et al [1]. This dataset was selected because it is medium sized in that it has more than a thousand observations and more than ten variables to explore the issues researched in this study.

Table 1. Chemicals Found in the Wines of the Red Wine Dataset

CHEMICAL	DESCRIPTION
FIXED ACIDITY	Most acids involved with wine are fixed or non-volatile (do not evaporate readily)
VOLATILE ACIDITY	Amount of acetic acid in wine which at too high of levels can lead to an unpleasant, vinegar taste
CITRIC ACID	Used in small quantities to add 'freshness' and flavour to wines
RESIDUAL SUGAR	Amount of sugar remaining after fermentation is completed. It is rare to find wines with less than 1 gram of residual sugar per litre and wines with greater than 45 grams per litre are considered sweet
CHLORIDES	Amount of salt in the wine
FREE SULFUR DIOXIDE	The free form of sulfur dioxide (i.e. SO_2) in equilibrium between molecular SO_2 (as a dissolved gas) and bisulfite ion. It prevents microbial growth and the oxidation of wine
TOTAL SULFUR DIOXIDE	Amount of free and bound forms of SO_2 in wine. SO_2 is mostly undetectable in wine. However, free SO_2 concentrations over 50 ppm becomes evident in the smell and taste of wine
DENSITY	The density of wine is close to that of water depending on the percent of alcohol and the sugar content
pH	Measure of how acidic or alkaline a wine is on a scale from 0 (very acidic) to 14 (very alkaline). Most wines are between 3-4 on this scale
SULPHATES	Wine additive that can contribute to sulfur dioxide gas (SO_2) levels, which acts as an antimicrobial and antioxidant
ALCOHOL	Percent of alcohol content of the wine

3 Identification of Class Structure in Data using Taxometric Attribute Analysis

3.1 Discriminatory Variables

One requirement with clustering is to use discriminatory variables that distinguish clusters in data. Members of the same cluster should have profiles that are like each other. If noise variables are included the variables used to cluster cases, it will make it harder to find cases with similar profiles as illustrated in Figure 1. It can be seen in this figure that variables 3 and 4 are noise ones and basically have random patterns of scores, while the scores for variables 1, 2, 5 and 6 show a distinct pattern indicative of a cluster. If there are too many noise variables in the data, it is unlikely that the patterns in the data will be recovered as distinct clusters. One way of ensuring that discriminatory variables are employed in the clustering is to use those that distinguish classes in data.

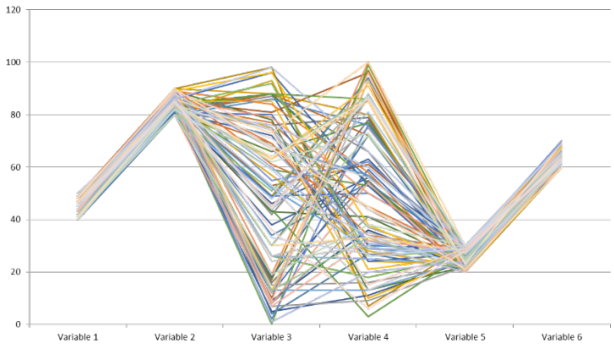


Fig. 1. Examples of Discriminatory and Noise Variables of a Pattern in Data

3.2 Class as a Concept

A class is a distinct category of entities found in the world. Some are very concrete and easy to see, such as different classes of fruit, with apples, oranges and pears being examples. Others are more abstract and not so easily identified, such as the different classes of patients that use health services.

Each class has a population. All populations can have sub classes or subpopulations. For example, apples are a population of fruit and consist of different sub classes such as Gala, Delicious and Fuji apples.

Each class has a set of defining attributes that distinguish it from other classes – in their presence or their absence. For example, the defining attributes of different classes of fruit can include their chemical properties, time of year the fruit is harvested, colour, shape, size, crustiness of the skin, texture, and flavour as examples.

3.3 Defining attributes of a Class

Factor analytic methods [2-3] are employed to discover the underlying dimensions of a population. For example, introversion-extraversion [4-6] is one of the 'big five' personality traits that has been discerned in the population of human beings using factor-analytic methods. Introverts and extraverts are not separate classes of human beings. They are opposite ends of the population distribution of a personality facet for the class of entities called 'homo sapiens' or 'human beings'.

In contrast, taxometric techniques are applied to discover the defining attributes of classes in data, such as the defining attributes of apples. A number of taxometric techniques [7-9] have been developed. The one used to identify class structure in the red wine dataset was previously called 'outlier table analysis' but is now called 'taxometric attribute analysis' (TAA). It is so called because each class has a defining signature or a distinguishing profile that best describes members of the class. The signature is made up of attributes and scores with some being unique to the class and others being shared with other classes.

How does TAA discriminate the defining attribute of classes? This procedure is described in another paper

[10] but essentially it is a non-parametric, multivariate statistical technique for identifying the frequencies of high and low variable scores in data. It does frequency counts of the times one variable, called an 'anchor variable', has a score above a high specified cut-off and the scores for other variables in the dataset are also above the same cut-off. There are also frequency counts done when one variable score is above a high specified cut-off and the scores for other variables in the dataset are below a low specified cut-off.

This process is repeated until all variables in the dataset have been employed as an anchor variable. This provides two matrices of frequency counts of what are labelled 'high-high' frequency counts and what are called 'high-low' frequency counts.

If a variable is non-discriminatory (i.e. is a noise variable), it will have high frequencies both above and below the specified cut-off scores because there are no patterns in the scores with the variable as illustrated with variable 4 in Figure 1. If the frequency count below the low cut-off is subtracted from the other above the high cut-off, the difference in frequency counts will be small and, in some cases, will be zero. If it is discriminatory, it will have a high frequency count either above or below the specified cut-off but not both. Therefore, if one frequency count is subtracted from the other, the difference in frequency counts will be high. From this perspective, if the matrix of high-low frequency counts is subtracted from the matrix of high-high frequency counts it will furnish a difference matrix. This matrix will consist of frequency count differences for the variables in each column of the difference matrix with those having high difference frequencies being the discriminatory features of a class.

Some columns in the difference frequency matrix will have similar patterns of frequency count differences because the high-count entries are the variables that define a class. Since the focus of TAA is on identifying the generic classes in the data, the columns of the difference matrix can be correlated using a non-parametric correlation coefficient such as Spearman Rho [11] and the resulting correlation matrix can have principal components [12-13] extracted to identify those variables that are indicative of one or more classes in the data. This is illustrated with the class structure found in the red wine dataset in Table 2.

3.4 Technical Details

Before any analysis was performed with the red wine dataset, the scores for the chemicals were converted to percentile ranks [14]. The high cut-off scores for the chemicals were above the 50^{th} percentile rank and the low cut-off scores were equal to or below the 50^{th} percentile rank. If variables with different score ranges are used in taxometric analysis, this will distort the results obtained and will provide misleading outcomes. The algorithm used to do TAA was developed over two decades ago and it was found to work well by using percentile ranks. The algorithm was coded in Base SAS [15].

3.5 Initial Determination of Variables

There are a few lessons to be heeded with the use of taxometric techniques. The first lesson is that, regardless of which methods are employed to identify class structure in data, it is up to those doing the research to consult with domain experts to determine

which variables to use in the analysis. This consultation is often an iterative process, in that it takes time to work out which variables are suitable to use and which ones are not by applying them in the taxometric analysis of data to see what results emerge.

Those doing taxometric analysis should also consult with domain experts to see if innovative features can be found to use as inputs to the analyses, such as subtle ones that will identify medical practitioners who over service patients to maximize revenue and try to disguise their over servicing. The important requirement here is that taxometric techniques will identify the defining variables of classes from the pool of variables selected for the analysis, but it cannot determine the variables that should be included in the starting pool. This determination requires human judgment - preferably expert judgment.

The second lesson is that if one or two important variables are left out of the starting pool, it can make the difference between identifying a meaningful class and not identifying one in data. It therefore pays to be over inclusive in the potential features in the starting pool to ensure classes that exist are identified.

From this perspective, two of the authors of this paper tested how many variables could be analysed taxometrically using TAA on a mid-range server and found that at least 350 could be used in the analysis, realising that this meant taking principal components of a 350 by 350 correlation matrix. It was also discovered that some classes have a many defining variables and that these class variables could further be subdivided into component variables by taking principal components [12-13] of the class variables. This analysis gives a better understanding of the constituents of a class.

The third lesson is that it pays to use different high and low percentile rank cuts-offs such as 70/30, 60/40, 50/50 and 40/40 to ensure consistency of the results. A rule of thumb learned from applying TAA is to use 50/50 with small datasets (i.e. less than a few thousand observations) while between 90/10 through to 60/40 can be used with large populations.

4 Identification of Clusters in the Data

Clusters were extracted from the red wine dataset using the normal mixtures method [16-18]. This method was preferred because it was found to give more informative results than other clustering techniques that were examined when it came to clustering profiles and understanding why entities such as wines are in certain clusters.

The normal mixtures method is an iterative technique, but rather than being a clustering method to place like rows together, it is more of an estimation procedure to determine cluster membership. It estimates the probability that a row of case scores is in each cluster. The normal mixtures approach to clustering predicts the proportion of responses expected within each cluster. The assumption is that the joint probability distribution of the measurement columns can be approximated using a mixture of multivariate normal distributions, which represent different clusters. The distributions have mean vectors and covariance matrices for each cluster. An expectation-maximization strategy is employed to optimize cases placed in each cluster.

SAS JMP [19] version of normal mixtures was applied to produce the cluster means for all variables in the red wine dataset. A 10-cluster solution was found to give an

informative understanding of the clusters in this dataset. The same software was employed to produce the parallel coordinate plots of the clusters.

The decision was taken to show the cluster means for all variables for three clusters that were coupled respectively with wine quality ratings of 5, 6 and 7. This included the cluster means for all variable for Clusters 1 (coupled to quality rating 6), Cluster 3 (coupled to quality rating 5) and Cluster 10 (coupled to quality rating 7) are shown in Table 4.. The parallel coordinate plots for these clusters are shown in Figure 2. This includes the plot for another cluster that was coupled to more than one quality rating to illustrate the results obtained. The results in this table show the trends with the scores of each chemical in relation to the rated quality of the red wines. The decision was also taken to use all the chemicals in Table 1 as they distinguish one or both classes in Table 2.

The scores for the chemicals were converted to peer relativity scores using the following maximum-minimization normalization score formula so that they had the same range from 0 to 100:

$$\text{Peer Relativity Score} = \left[\frac{Variable\ Score - Minimum\ Variable\ Score}{Maximum\ Variable\ Score - Minimum\ Variable\ Score}\right] * 100 \qquad (1)$$

Peer relativity scores were used for the clustering of the wines because they had been used in other studies of the partitioning of data where other clustering techniques were tried, and they were found to give results that enabled the techniques to be compared to see which ones were giving the best outcomes.

5 Identification of Different Pairs of Chemicals through Classification by Counting

The steps above cover the use of unsupervised learning for discovering class structure and clusters in the red wine dataset. This part of the paper describes a supervised learning solution for identifying different pairs of chemicals that contribute to the quality of wines in the red wine dataset. As indicated earlier, this method is called 'Classification by Counting' (ClassC).

ClassC uses data formed from a morphological matrix of the chemical attributes. Each column in the matrix is a chemical attribute. The rows in the matrix are the levels of measurement of the attributes, where the levels are transformations of the measures into 'bins' of equal intervals, using relative ratings. In effect, these bins are ordinal levels of measurement.

This paper compares the results of using six bins for the attributes and three bins for the quality rating (that is, high, medium and low) or three bins (that is, high, medium and low) for both input attributes and quality.

ClassC forms the combinations of the levels of the attributes, taken two at a time. These pairs of levels are the 'fundamental' or intersection areas of overlapping circles in the Venn diagram for the attributes.

They are also conjunctive clauses within a Disjunctive Normal Form (DNF) of the rule for classifying the quality of wine. In Boolean logic, a DNF is a canonical normal form of a logical formula consisting of a disjunction of conjunctions [20]. For example, a clause could be Alcohol 6 and Fixed Acid 3 when using six bins or High Alcohol and Low Fixed Acid when using the three-bin transformation. One or more clauses are in DNF if they are joined by an inclusive disjunction relationship; for example, High Alcohol and Low Fixed Acid or High Residual Sugar and Medium Free sulfur.

The algorithm for ClassC is:

1. Determine the morphology of the data: forming dimensions of performance (i.e. chemicals) and levels of measurement within each dimension
2. Transform the measures of the attributes into ordinal levels.
3. Describe each exemplar or 'instance' of the 1599 wines as a profile of levels on the 11 chemical attributes given in Table 1. For example, an instance of wine could be one that has High Fixed Acidity, Low Citric Acid, Medium Residual Sugar, Low Chlorides, High Free Sulfur dioxide, High Total Sulfur Dioxide, Low Density High Sulphates and High Alcohol.
4. Group the 1599 instances into teaching (or training) and testing sets, according to a predetermined ratio (2/1 and 3/1 in this case). For example, if using 2/1 ratio then ClassC takes every third instance and puts that into the testing set and then analyses the remaining two thirds of the instances.
5. If an exemplar or 'instance' (a particular wine with the same chemical profile) appeared with more than one output level then 'quieten' (i.e. 'dampen') the data by considering only the most often seen output level. For example, if the same profile appeared in 300 teaching instances, 200 times with a rating of High Quality, 80 times with Medium Quality, and 20 times with Low Quality then only the 200 High Quality instances will be analysed further.
6. Form pairs of levels (conjunctive clauses).
7. Count the number of times over the teaching set that each observed clause appears with the different levels of output (High, Medium or Low quality of wine).
8. Considering only the High and Low output levels, determine if a pair of attribute levels occurs wholly with one level or the other. That is, over all the instances containing that pair in its description, does it always have a rating of High or always a rating of Low.
9. If a pair does have a pure assignment of an output level that add that pair to the DNF describing the rule for that output level.
10. Validate the rule by applying it to the instances in the test set, counting how many of the clauses in the rule are satisfied for each output level, and predicting the output level from the relative number of clauses that are met.
11. Produce a confusion matrix for the correct and incorrect predictions of each output level.
12. Use the accuracy (percentage of correctly classified cases) and the Matthews Correlation Coefficient [21-22] of the confusion level to see if the rule assigns the instances well enough to the output levels.

6 Results

6.1 Classes in the Red Wine Dataset found using TAA

The classes found in the red wine dataset using TAA are shown in Tables 2 and 3 below.

Table 2. Pattern Loadings for Each Class for the Red Wine Dataset

CHEMICAL VARIABLES	CLASS 1	CLASS 2
FIXED ACIDITY	0.46085	**0.88684**
VOLATILE ACIDITY	**-0.92354**	-0.36678
CITRIC ACID	**0.74888**	**0.65894**
SUGAR	0.03726	**0.97896**
CHLORIDES	**-0.59108**	**0.80548**
FREE SULFUR DIOXIDES	**-0.74156**	**-0.63875**
TOTAL SULFUR DIOXIDES	**-0.97238**	0.14271
DENSITY	-0.22273	**0.97246**
pH	-0.24955	**-0.96608**
SULPHATES	**0.94235**	0.31633
ALCOHOL	**0.95646**	-0.28332
QUALITY	**0.99823**	-0.02019

Table 3. Variance Explained by Each Class for the Red Wine Dataset

CLASS	PERCENT	CUMULATIVE PERCENTAGE	EIGENVALUES
1	**59.79**	**59.79**	**7.174**
2	**38.83**	**98.62**	**4.66**

The defining variables of the Class 1 wines are highlighted in bold in Table 2 and include the quality of the wine. This result suggests that wines low in volatile acidity; high in citric acid; low in chlorides, free sulfur dioxides and total sulfur dioxides; and high in sulphates and alcohol are associated with higher quality wines.

The results in Table 2 indicate there is a second class of wines that have high fixed acidity, high citric acid, high sugars, high chlorides, low free sulfur dioxides, high density and low pH as their defining features but these measures do not appear to have a bearing on the quality of the wines.

Table 3 results show that the Class 1 wines account for nearly 60 percent of the variance while the Class 2 for nearly 39 percent of the variance in the red wine dataset. These are two very distinct and sizeable classes in the dataset.

It is normal practice to give each class a label based on its defining attributes. This is difficult to do with Class 1 and 2 wines because it is not clear what the underlying concept the defining chemicals are measuring with each class. One possible

interpretation is that Class 1 chemicals are indicative of a high-quality wine class and those of Class 2 are indicative of a flavour class where the wines can have flavours ranging from bitter to sweet depending upon the acidity, citrus, salt, sugar and free sulfur dioxides present in the wines.

6.2 Clusters in the Red Wine Dataset

The results of the normal mixtures clustering are shown in Table 4 below. The mean scores are listed in the order of cluster 3, 1 and 10 in Table 4, as this shows the wines listed in order of the rated quality categories of 5, 6 and 7.

Table 4. The Mean Peer Relativity Scores for the three Biggest Clusters found in the Red Wine Dataset

CLUSTER	**3**	**1**	**10**
SIZE	516	512	124
%	32.2	31.8	7.7
FIXED ACIDITY	31.86	33.63	42.48
VOLATILE ACIDITY	30.82	25.34	15.07
CITRIC ACID	22.2	23.32	19.63
SUGAR	24.11	24.5	25.54
CHLORIDES	14.38	13.78	12.98
FREE SO2	59.48	57.8	52.02
TOTAL SO2	54.4	44.7	36.3
DENSITY	50.88	50.88	45.74
PH	44.88	45.67	39.37
SULPHATES	21.94	27.98	35.13
ALCOHOL	26.42	40.02	52.7
QUALITY RATING	5	6	7

Four parallel coordinate plots are shown in Figure 2 for these three largest clusters. The profile pattern for cluster 5 is also shown to illustrate the trend of the wines in this cluster being coupled to more than one quality rating.

The results in Table 4 show that the clusters 1 (see Figure 2(a)), 3 (see Figure 2(b)) and 10 (see Figure 2(c)) are coupled to the respective quality ratings of 6, 5 and 7 of the wines. It is important to note that the quality ratings were included as inputs to the clustering.

An examination of the mean profile for each cluster in Table 4 show that the higher the quality of the wine, the higher the scores for fixed acidity, sugar, alcohol and sulphates and the lower the scores for volatile acidity, citric acid, chlorides, free sulfur

dioxides, total sulfur dioxides, density and pH. There are no distinct clusters for wines rated 3, 4 and 8 in quality.

It is also highlighted that the smaller clusters had patterns like those seen in Figure 2(d). They are coupled to more than one quality rating of the wines.

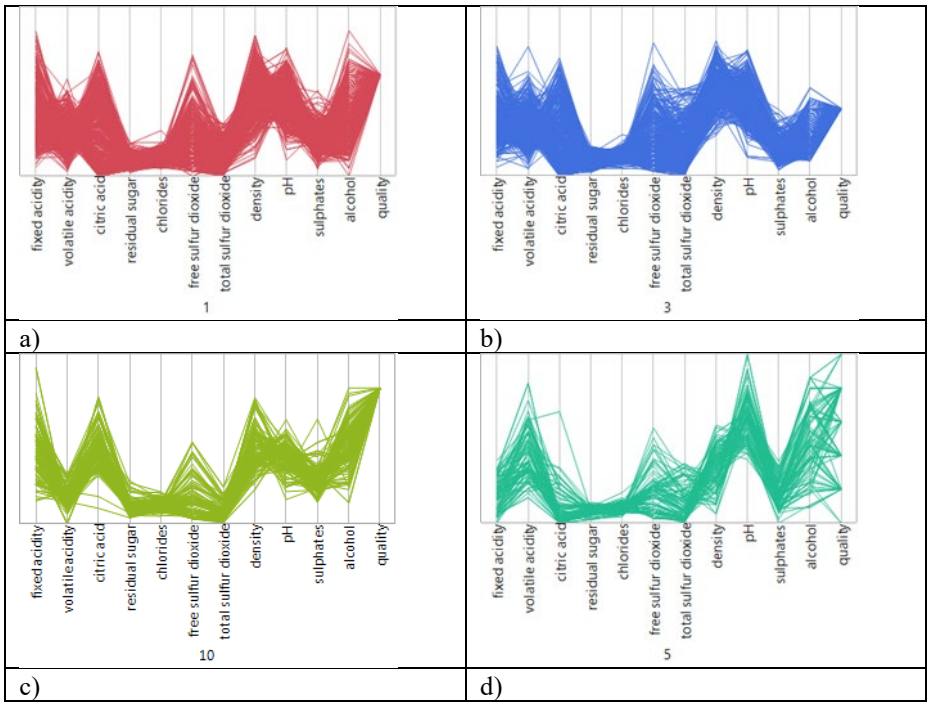


Fig. 2. Parallel Coordinate Plots for Four Clusters

6.3 Combinations of Chemicals that Predict Quality of Wine

The results of using the ClassC method for the red wine data are shown below. This method was used for both the six-bin and three-bin solutions, using teaching (or training)/testing ratios of 2/1 and 3/1. Exploratory runs, using intermediate analysis not described here, determined which of the chemical attributes were the most useful in the derivation of the conjunctive and disjunctive rules. The best results, as shown by the confusion matrices and Matthews Correlation Coefficients, are described next.

Six-Bin Solution

The results for the Six-Bin Solution are shown in Tables 5 and 6. This solution had a teaching/ testing ratio of 3/1 with 41 high quality wines in the teaching set and 13 in the test set and 426 low quality wines in the teaching set and 142 in the test set. Pairs of attributes for free sulfur dioxide, alcohol, residual sugar, citric acid, fixed acidity were investigated as these were found to give the best results in predicting the quality of the wines with a six-bin solution.

Table 5. Confusion Table Test Results Using for the Six Bin Solution

Actual/Predicted	High quality	Low quality	Ratio
High quality	10	3	0.70
Low quality	0	142	1.00
Ratio	1.00	0.98	0.98

Table 6. Disjunctive Normal Form of Rules for the Six Bin Solution

High quality	High free sulfur dioxide and Medium citric acid; High free sulfur dioxide and High alcohol; High free sulfur dioxide and High residual sugar; High free sulfur dioxide and High citric acid; High free sulfur dioxide and Medium fixed acidity; High alcohol and High residual sugar; High alcohol and High citric acid; High alcohol and High fixed acidity; High alcohol and Medium fixed acidity; High residual sugar and High citric acid; High residual sugar and Medium fixed acidity; Medium residual sugar and High citric acid; High citric acid and High fixed acidity; High citric acid and Medium fixed acidity
Low quality	not High free sulfur dioxide and not High alcohol; not High free sulfur dioxide and not High residual sugar; not High free sulfur dioxide and not High citric acid; not High free sulfur dioxide and not High citric acid; not High alcohol and not High residual sugar; not High alcohol and not High residual sugar; High alcohol and not High citric acid; not High alcohol and not High citric acid; not High alcohol and not High fixed acidity; not High residual sugar and not High citric acid; not High citric acid and not High fixed acidity

This matrix gave a MCC = 0.87 and no unclassified instances. The accuracy rate was 91.6 percent for the test set.

The following DNF (with ';' representing the inclusive disjunction connective) were applied as rules to predict the quality of the red wines in the test set.

For high-quality wine, high levels of alcohol (rating above 5) appear in many clauses, as does high free sulfur dioxide. Residual sugar appears as High or Medium with many other attributes. Fixed acidity appears less often, forming only a few clauses.

The rules for low-quality wine are mostly the contraries of the rules for the high-quality wine. This mirror relationship adds confidence to the suggestion that these levels of attributes do discriminate successfully between the levels of quality of wine.

Three-Bin Solution

The results for the Three-Bin Solution are shown in Tables 7 and 8. The 'best' three-bin solution had a teaching/ testing ratio 2/1 with 36 high-quality wines in the teaching set and 18 in the test set and 379 low-quality wines in the teaching set and 189 in the test set. Pairs of attributes for free sulfur dioxide, alcohol and fixed acidity were investigated, as these were found to give the best results in predicting the quality of the wines with a three-bin solution.

Table 7. Confusion Table Test Results for the Three Bin Solution

Actual/Predicted	High quality	Low quality	Ratio
High quality	15	2	0.87
Low quality	0	189	1.00
Ratio	1.00	0.99	0.99

This result gave a MCC = 0.93 but with one unclassified instance. The accuracy rate was 99% for the test set.

The following DNF were applied as rules to predict the quality of the red wines in the test set.

Table 8. Disjunctive Normal Form for Rules fort he Three Bin Solution

High quality	High free sulfur dioxide and High alcohol; High free sulfur dioxide and High fixed acidity; High free sulfur dioxide and Medium fixed acidity; High alcohol and High fixed acidity; High alcohol and Medium fixed acidity
Low quality	not High free sulfur dioxide and not High alcohol; not High free sulfur dioxide and not High fixed acidity; not High alcohol and High fixed acidity

Using a coarser binning gave more robust results, with fewer misclassified instances because there were fewer rare occurrences of some fundamental areas in the training set. It also required fewer DNF clauses to classify the teaching set.

The addition of extra attributes to the profile removed the unclassified instance but added to the misclassifications.

The DNF are similar in both cases. They both show that high-quality wines require high levels of free sulfur dioxide, high levels of alcohol, and high levels of fixed acid. It is possible that high levels of residual sugar and of citric acid could contribute but there is not robust evidence to support this expectation.

7 Discussion

There are two sets of results provided for unsupervised learning and two for supervised learning. The unsupervised learning results are found in Table 2 for Class 1 and 2 wines and the mean profiles for clusters 1, 3 and 10 are found in Table 4. These results indicate that the levels of certain chemicals have a bearing on the quality of the wines. Trends in the two tables suggest that increasing levels of fixed acidity, sugar, sulphates and alcohol while decreasing levels of volatile acidity, citric acid, chlorides, free sulfur dioxides, free sulfur dioxides, density and pH boost the quality of the wines.

There is one caveat with the cluster results in Table 4: these are filtered results. That is, they consist of wines that have similar profiles. There were a few rated wines in the 5, 6 and 7 categories that had outlying profiles and were found in other smaller clusters. These wines have different profiles compared to those seen with the parallel coordinate

plots for the wines shown in the clusters in Figure 2. A visual inspection of these wines showed that they had high or low scores on certain chemicals such as sugar, chlorides, density and pH.

The question can be asked why similar trends were not found with the wines rated 3, 4 and 8. One possible reason is that the numbers in these rated categories are too small to discern unique patterns.

It is also expected that the wines rated 3 or 4 in quality would also have too much of certain chemicals and not enough of others. A visual inspection of the chemical scores for these wines revealed that one wine had the maximum score for volatile acidity, a second for citric acid, a third for chlorides and a fourth for sulphates. One had the minimum score for alcoholic content. In contrast, wines rated 8 were prominent for having high alcoholic content. There are, of course, many examples that do not have extreme scores, but they follow the trends for wines found in the 5, 6 and 7 rated categories of having too much of certain chemicals that decrease the quality of wine or low quantities of those that increase the quality of this product.

Turning to the supervised learning results, the use of the ClassC method found for the six-bin results that the presence of high levels of alcohol and high free sulfur dioxide were predictive of high-quality wines. High and medium fixed acidity, high and medium residual sugar, and high and medium citric acid was found to be predictive of high-quality wines in combination of other attributes.

The three-bin results showed that high levels of free sulfur dioxide, high levels of alcohol, and high and medium levels of fixed acid were predictive of high-quality wines.

The unsupervised results are descriptive of the contents of the wines coupled to certain wine ratings, while those of the supervised learning discriminate chemicals that are predictive of the quality of the wines. Generally, the results for both unsupervised and supervised learning are in agreement about what chemicals contribute to the quality of the red wines. The one exception is free sulfur dioxide. The clustering results suggest that lower levels of this chemical are associated with higher quality wine while the results of the ClassC analysis indicates the opposite that higher levels of this chemical contribute to a higher quality wine. It is not clear what is causing this difference. It could be that free sulfur dioxide compensates for the presence of either higher or lower levels of other chemicals in particular wines.

An examination of the correlation matrix (not given here) between the chemicals reveals that free sulfur dioxide has low-to-zero correlations with all other chemicals except total sulfur dioxide. While this issue has not been researched, it is known that meaningful score configurations can be found in data that have low-to-zero correlations between variables [23] and are coupled to different outputs such as the quality ratings of wines.

Two solutions that could be explored in future research include affinity mining, also called frequent pattern mining [24], where the different combinations of chemicals and their levels that are linked to low-quality, medium-quality, and high-quality wines are discerned. To give a few simple hypothetical examples, frequent distinct patterns in the red wine data could include:

IF High Fixed Acid. Medium Free Sulfur Dioxide. High Alcohol **THEN** High-Quality Wine

IF Medium Citric Acid. High Chlorides. Low Sugar **THEN** Low-Quality Wine

IF Low Sulphates. High Alcohol. Low Volatile Acidity **THEN** Medium-Quality Wine.

The above examples show triplets. There could be quadruples, centuples and higher combinations of the chemicals that have meaningful associations with the quality of the wines. It is to be noted that there are triplets in the DNF given by ClassC and it can be extended to analyse them more fully.

Configural frequency analysis [25-26] is another technique that could be applied to find frequent binary score patterns of high and low chemicals that are linked to high, medium and low-quality wines. This method, along with affinity mining, requires further research. These analyses could also identify other interesting combinations of chemicals linked to quality wines.

A few observations are offered with interpreting the results of this paper. The first is that ClassC results are the end-result of a filtering process to find which pairs of chemicals and their levels best predict the quality of wine. This filtering process is expected to leave out chemicals that can influence the quality of certain wines. For example, the presence of chemicals A and B but the absence of G may result in wines being given a high-quality rating whereas another with A, B and C present might lead to wines being given a low rating for quality. There are many possibilities with eleven chemicals and their having zero, low, medium and high levels. There are 11^4 or more than fourteen and half thousand alternative patterns with these chemicals and levels. The number of combinations depends upon the chemicals selected to be tested such as pairs, triplets, and higher. It is expected that the ClassC can analyze data for triplets as it shows that alcohol and free sulfur and not low fixed acidity almost implies high quality.

There were combinations of chemicals and levels that were not found in the red wine dataset while others had noticeable frequencies. For example, the triplet of High Alcohol, Low Volatile Acidity, Low Sugar is not found in the red wine dataset, while there are 896 instances of Low Volatile Acidity, Low Sugar, Low pH in the dataset. These many missing triplets could be worth testing to see what effects they have on the quality of the wines.

The classification accuracies obtained by Cortez et al [1] were 62.4 percent for a support vector model, and 59.1 percent for a neural network and a multiple regression model. The accuracy rates for the six-bin solution were 91.6 percent and for the three-bin solution of 99.0 percent with the test sets. These are high accuracy rates thus showing that the level of certain chemical pairs performs well in discriminating the quality of the wines in the red wine dataset.

One must be careful in comparing the results of the of the Cortez study compared to the ClassC study. Cortez et al used 10-fold cross validation on the whole red wine dataset while the ClassC results are based on the wines included in the test dataset. A fair comparison would have all developed models tested on a suitable independent dataset.

The predictive results achieved with the ClassC algorithm might also be attributable to the fact that it looked for disjunctive relationships between the chemicals where it was not only the chemicals that were present that counted, but also the ones that were absent. This method requires further research to see to what extent that ingredients that are absent contribute to the quality of a product. For example, absence of alcohol is a strong indicator of a low-quality wine. This approach may also highlight one of the limitations of regression modelling methods - that they do not account for what is missing in a mixture of ingredients that either add or subtract to the quality of a product.

From a broader perspective, the solutions explored in this paper are examples of compositional analysis that focuses on identifying the different ideal combinations of ingredients that produce quality products such as different quality foods and materials. This approach is important in the fields of food informatics [27] and other sciences that deal with the composition of materials, such as those used to produce advanced strong and durable composites [28-29] that play a role, for example, in the manufacture of aircraft.

This finding of predictions of the quality of wine from combinations of chemical attributes might be considered a heresy by those who are wine connoisseurs, but the day might come where, instead of fermenting wines in vats and kegs in vineyards the way they have been produced historically, they will be factory produced using artificial intelligence to come up with different mixtures of ingredients to produce different classes of high-quality wines. The wines will be customized to the different tastes of consumers. The results of the analyses reported in this paper indicate that the factory production of wines is technically feasible. It is just a question of using the right ingredients and applying the right climatic conditions to produce wines of quality.

8 Conclusion

The results of this study have demonstrated how one taxometric method was applied to identify the defining attributes of two classes of wine and how normal mixtures clustering was used to partition the wines using the chemicals found in the wines. Normal mixtures clustering was able to find couplings between certain clusters of the red wines and ratings of their quality.

The ClassC algorithm demonstrated how it is possible to winnow down all the alternative pairs of chemicals to a small set that predicts the quality of the red wines using conjunctive and disjunctive rules. The solutions explored in this paper are examples of different ways to identify which inputs contribute to an output.

It was also indicated that a composition analysis can be employed to find the different combinations of ingredients that produce a scrumptious cake, nice flavour of honey, a delightful jam, a tasty pie, a yummy biscuit, and similar. This method can assist food producers to provide quality products that meet consumer needs and tastes. The same can be done with advanced materials used to construct buildings, aircraft, and ships as examples.

References

1. Cortez, P; Cerdeira A; Almeida F; Matos T; Reis, J.: Modeling Wine Preferences by Data Mining from Physicochemical Properties. In Decision Support Systems, Elsevier, **47**(4), 547-553. ISSN: 0167-9236 (2009)
2. Child, D.: (2006). The Essentials of Factor Analysis, 3rd edition. Bloomsbury Academic Press, London (2006)
3. Mulaik, S.A.: Foundations of factor Analysis. Chapman and Hall, Boca Raton Florida (2010)
4. Digman, J.M.: Five Robust Trait Dimensions: Development, Stability, and Utility. Journal of Personality. **57** (2), 195–214 (1989)
5. McCrae, R.R; Costa, P.T.: Validation of the Five-Factor Model of Personality across Instruments and Observers. Journal of Personality and Social Psychology. **52** (1), 81–90 (1987)
6. McCrae, R.R; John, O.P.: An Introduction to the Five-Factor Model and its Applications. Journal of Personality. **60** (2), 175-215 (1992)
7. Waller, N. G; Meehl, P. E.: Multivariate Taxometric Procedures: Distinguishing Types from Continua. Sage, Newbury Park, CA (1998)
8. Beauchaine, T.P.: A Brief Taxometrics Primer. Journal of Clinical Child and Adolescent Psychology. **36**(4), 654-676 (2007)
9. Ruscio, J; Ruscio, A.M; Carney, L.M.: Performing Taxometric Analysis to Distinguish Categorical and Dimensional Variables. Journal of Experimental Psychopathology. **2**, 170-196 (2011)
10. Graco, W.J; Koesmarno, H.: Configurations and Couplings: An Exploratory Study. 9th International Conference ICDM New York USA (2013)
11. Daniel, W. W.: Spearman Rank Coefficient. Applied Nonparametric Statistics, 2nd Edition. PWS-Kent, Boston. P358-365 (1990)
12. Jackson, J.E.: A User's Guide to Principal Components. Wiley, Hoboken NJ (1991)
13. Jolliffe, I. T.: Principal Components Analysis. Springer Series in Statistics. New York: Springer-Verlag (2002)
14. Base SAS Software See
15. Percentiles, Percentile Rank & Percentile Range: Definition & Examples. See https://www.statisticshowto.com/probability-and-statistics/percentiles-rank-range/
16. McLachlan, G.J; Chang, S.U.: Mixtures Modelling for Cluster Analysis. Statistical Methods in Medical Research. **13**: 347-361 (2004)
17. McLachlan, G. J; Peel, D.: Finite Mixture Models. Wiley. Hoboken NJ (2000)
18. McLachlan, G. J; Krishnan, T.: The EM Algorithm and Extensions. Wiley, Hoboken NJ (1997)
19. SAS JMP Overview of Platforms for Clustering Observations. See https://www.jmp.com/support/help/en/15.2/index.shtml#page/jmp/overview-of-platforms-for-clustering-observations-3.shtml#
20. Disjunctive Normal Form. Encyclopedia of Mathematics. EMS Press, Berlin Germany (2001) https://encyclopediaofmath.org/index.php?title=Disjunctive_normal_form
21. Chicco, D; G. Juman, G.: The Advantages of the Matthews Correlation Coefficient (MCC) over F1 score and Accuracy in Binary Classification Evaluation. BMC Genomics, 21:6 (2020)doi.org/10.1.1186/s12864-019-6413-7, retrieved 11 Feb 21
22. Tharwat, A.: Classification assessment methods. Applied Computing and Informatics, www.emerald.com/insight/2210-8327.htm (2018), retrieved 11 Feb 21
23. Meehl, P. E.: Configural scoring. Journal of Consulting Psychology. **14**, 165-171 (1950)
24. Han, J; Kamber, M; Pei, J.: Data Mining Concepts and Techniques. Morgan Kaufmann, Burlington MA. Chapter 7 (2012)

25. Schrepp, M.: The use of configural frequency analysis for exploratory data analysis. British Journal of Mathematical and Statistical Psychology. 591(1), 59-73 (2006)
26. Von Eye, A.: Configural Frequency Analysis. Methods, Models and Applications. Routledge, New York. (2012)
27. Pau, P.K; Aithal, P.S; Bhuimali, A.: Food Informatics and its Challenges and Opportunities- A Review. International Journal on Recent Researches in Science, Engineering and Technology. 5 (9), 46-53 (2017)
28. Williams, J.: The science and technology of composite materials. https://www.science.org.au/curious/technology-future/composite-materials
29. Gay, D.: Composite Materials. CRC Press, Boca Raton Fl (2015)

Author Index