A Framework for the Classification and Exploration of Semi-Structured Data

Louis Burger and Jan van Vuuren

July 26, 2024

A framework for the classification and exploration of semi-structured data

Louis Willem Burger[0009−0009−9719−6735]
and Jan van Vuuren[0000−−0003−4757−5832]

Stellenbosch Unit for Operations Research in Engineering, Department of Industrial Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa
`louiswillemburger@icloud.com, vuuren@sun.ac.za`

Abstract. In recent years, advances in *natural language processing* (NLP) have broadened the traditional boundaries of proficiency in artificial intelligence considerably, particularly with respect to tasks demanding high levels of cognitive sophistication. While the evolution of NLP has been impressive, a significant gap remains in the efficacy of NLP with respect to the classification and exploration of semi-structured data. These data comprise a blend of structured and unstructured features, and hold considerable potential, especially when traditional NLP classification tasks are complemented with structured meta-data.
A generic framework, called the *Classification and Exploration of Semi-structured Data* (CESD) framework, is proposed in this article for enhancing the efficacy of classification tasks based on unstructured data when including insights gleaned from accompanying structured data. The versatility of the framework empowers users to modify it or to add framework components, as per their specific requirements, with the overarching goal of equipping users with a holistic understanding of the best-performing classification models and to elucidate the inherent characteristics of semi-structured input data sets based on their structured and unstructured features.
A computerised instantiation of the CESD framework is validated by applying it to a real-world data set. The case study data pertains to the severity of software error logs in a production environment, and comprises both structured and unstructured data describing these errors.

Keywords: NLP classification · Clustering · Semi-structured data.

1 Introduction

In the past, it was widely acknowledged that artificial intelligence surpassed human capabilities in data-driven decision-making processes. When it came to tasks requiring cognitive sophistication and creativity, however, the general consensus was that artificial intelligence seemingly lagged behind human competency [13]. Yet, over the past few years, there has been a remarkable evolution in language-based artificial intelligence systems [18]. Significant progress has specifically been observed in the realm of *natural language processing* (NLP), a sub-discipline of artificial intelligence aimed at enabling computers to interpret and generate language in a manner akin to human cognition. Such advancements have necessitated a re-evaluation of the widely held view that artificial intelligence leaves much to be desired in automated technologies based on cognition. Today it is recognised that the rise of NLP has undoubtedly marked a pivoting point in the development of computational linguistics in particular, and artificial intelligence in general.

During the 1990s, there was a notable convergence in the research community towards empiricism and the application of probabilistic language models. This shift allowed researchers to evaluate empirically the assertions made by Chomsky and others [12] that machines were incapable of the level of cognitive understanding of grammar achievable by humans, proving that many of these earlier arguments, although persuasive on paper, did not hold up to empirical scrutiny [48]. As a result, models based on probabilistic and statistical approaches dominated NLP during the 1990s. Rather than NLP being considered a rule-based, deterministic domain, neural networks and deep learning techniques today dominate the field, which has enjoyed substantial growth of late [26]. The advent of tools and techniques in NLP has not only expanded the boundaries of how we perceive language understanding, but it has also been instrumental in devising novel applications that serve real-world purposes.

One notable manifestation of the aforementioned growth is the ability of NLP to classify unstructured text data. This capability is considered a cornerstone achievement, providing solutions in areas ranging from sentiment analysis [43] to document clustering and topic modelling [8]. In each case, the essence lies in comprehending the inherent characteristics of textual information and deciphering the underlying patterns, sentiments, or themes.

In parallel with these advances, structured data classification and exploration remains a well-established discipline, often characterised by its strict adherence to relational data structures and schemas [25]. Structured data sets, which predominantly contain categorical or numerical features, are considered easy to process and analyse due to their inherently organised and predictable characteristics.

As our environment transitions to a data-driven world, a conspicuous gap has, however, emerged. Although there are numerous techniques available in the literature for various classification and exploration tasks within the realms of structured data and unstructured data, few methodologies prevail for similar tasks in the context of *semi-structured* data[1] [2]. The significance of this gap in automated competency becomes apparent when one considers real-world applications in which tasks requiring NLP classification are often accompanied

[1] When reference is made to the amalgamation of structured and unstructured features in a data set, the term *semi-structured data* is used to describe the result, conforming to the definition of Marr [37], where semi-structured data may take the form of a relational table, containing separate structured and unstructured features (with the former often enriching the latter as meta-data). This choice of nomenclature is not to be confused with another recognised definition according to which semi-structured data are structured data in a non-relational unstructured format [54],

by structured meta-data. If leveraged correctly, these meta-data may well provide valuable insight, potentially enhancing the classification process. In the medical domain, for instance, a combination of the NLP classification of electronic health record free-text data accompanied by structured electronic health record data has shown potential improvements in the diagnosis of conditions such as non-valvular atrial fibrillation [16]. Similar benefits may be enjoyed in consumer science, where product reviews, although primarily textual, may be equipped with meta-data such as timestamps, user ratings, or geographic tags. These meta-data may yield valuable insight into the characteristics of different reviews, potentially changing the manner in which classification is performed. In a similar vein, spam classification tasks related to the content of emails might be greatly enhanced when informed by meta-data such as hosting addresses, file sizes, and the geographical origins of emails.

These compelling applications underscore the potential utility of tasks that require NLP classification in which an analyst has access to structured meta-data. The premise in such an application not only pertains to the amalgamation of structured and unstructured data forms, but also to the synthesis of insights that may be drawn from structured attributes and their potential to improve upon the classification accuracies achievable when the analyst only has access to textual data elements.

The aim in this article is to explore the juncture between the contexts of structured and unstructured data when performing classification tasks. More specifically, a framework is put forward for the *classification and exploration of semi-structured data* (CESD), hereafter referred to as the CESD framework. The objective of the framework is to guide a user towards enhancing the accuracy of classifying data objects exhibiting unstructured text features into similarity classes by incorporating insight gleaned from additional, structured (numerical and/or categorical) data features associated with the data.

A pivotal facet of the framework is that it facilitates a dynamic process during which the user can choose between two different data streams — in isolation or in combination — one for unstructured data and geared towards NLP classification, and the other aimed at clustering structured data into similarity classes. The framework is generic in the sense that the constituent components along any of the aforementioned data streams may be populated with a diverse repertoire of models, based on the subjective preferences of the user and tailored to the particulars of the underlying empirical study at hand.

The framework accepts as input raw, unstructured textual data in conjunction with structured data from any application domain. It offers guidance throughout the entirety of the analysis process, starting with data preprocessing, systematically proceeding along any or both of the two data-streams, and ending with an appropriate aggregation and synthesis of the results obtained. The ultimate aim is to provide the user with a comprehensive and concise overview as to which classification model or combination of models performs the best in

such as *extensible markup language* (XML) and *javascript object notation* (JSON) files.

respect of classifying the semi-structured input data into similarity classes and to provide the user with inherent characteristics of these data based on their structured and unstructured features. It thereby enables the user to extract pertinent and actionable insights from the unstructured input data.

2 Literature review

Although no work directly related to the proposed framework exists according to the authors' best knowledge, there are a myriad of publications exploring the relationship between structured and unstructured data. In some of these sources, it is argued that there may indeed be valuable information in the synergy between structured and unstructured data features which, when extracted appropriately, might enhance both the accuracy and reliability of NLP classification models [34]. A case in point is the work of Liu [34] in which improved accuracy and reliability of sentiment classification was demonstrated when the textual data set was complemented by structured meta-data. Similarly, Tang *et al.* [56] explored improvements in lung nodule classification when fusing structured and unstructured (images in this case) data. Khaleghi et al. [30], on the other hand, proposed a tree-based approach for classifying semi-structured data. Furthermore, King [31] proposed ensembling the outcomes of different classification model streams handling structured and unstructured data, respectively.

The CESD framework employs pertinent algorithms from both the domains of traditional machine learning and NLP. Various notions related to these algorithms are discussed in the remainder of this section.

2.1 Preprocessing of structured data

Gathering data and subsequently converting them into an appropriate format, suitable for either predictive or descriptive analyses, often proves to be a demanding task [9]. Following initial data collection, it is important to scrutinise and attempt to enhance the data quality before any analytic procedures are applied — a process commonly known as *data cleaning*. Techniques for this process are typically focused on recognising and adapting to outliers, rectifying erroneous data entries, and imputing missing values in the data [41].

2.2 Data partitioning

In the context of machine learning, the data used to provide a model with experience is commonly referred to as the *sample set*. This sample set is typically partitioned into three distinct subsets during model development, namely a *training set*, a *validation set*, and a *test set*. These data subsets collectively serve the purpose of both training and evaluating machine learning models under consideration for deployment [53]. While the training set facilitates the construction of a model, the validation and test sets are utilised to gauge its efficacy [32].

2.3 Hyperparameter tuning

The performance of a machine learning model is strongly influenced by the selection of values for its hyperparameters. The often arduous process of choosing appropriate values for these hyperparameters is referred to as hyperparameter *tuning*. There are a wide variety of techniques available in the literature for hyperparameter tuning, each with its own complexities, advantages and disadvantages. While state-of-the-art techniques may employ metaheuristics or probabilistic methods for this purpose [6], traditional techniques typically include *manual search*, *grid search*, and *random search*.

2.4 Evaluation metrics

A vast number of evaluation metrics are available in the literature for evaluating the efficacy of classification and clustering tasks. The most popular metrics for classification tasks are *precision*, *recall*, the *F1-score*, *accuracy*, the *area under the receiver operating curve* (AUROC), and the *area under the precision recall curve* (AUPRC) [17, 19, 33]. As for evaluating clustering quality, the most common metrics for gauging separation quality include the *silhouette* coefficient [49], the Caliński-Harabasz index [11], and the Davies-Bouldin index [14], whereas the *adjusted rand index* (ARI) [24] proves to be a popular metric for gauging similarity between clustering partitions.

2.5 Supervised and unsupervised machine learning algorithms

The algorithms suggested for incorporation into the CESD framework may be partitioned into three categories, pertaining to classification, clustering, and dimensionality reduction tasks.

In the realm of supervised learning, classification is a machine learning domain in which the objective is to assign each data record in a data set to one of k pre-defined categories. The process involves the estimation of a function mapping the data records to category labels or to a category membership probability distribution during a training process based on labelled (domain, range)-examples. In the latter case, the probabilistic function outputs may be converted to categorical assignments by a process of thresholding. The CESD framework is capable of accommodating traditional, yet effective, classification algorithms, such as decision trees [10], bagging decision trees [20], boosting decision trees [50], logistic regression [7], and naïve Bayes classifiers [44].

In the domain of unsupervised machine learning, clustering plays an important role and involves discerning specific features or patterns in a data set, which can subsequently facilitate the formation of meaningful groups or *clusters* [3]. The CESD framework requires distinct clustering configurations suitable for distance-based clustering of mixed data — similar to those discussed by Van de Velden *et al.* [58]. Mixed-data clustering techniques may adopt a variety of distance metrics and typically incorporate some variation of the k-means algorithm. Five quintessential clustering methods for mixed data clustering include the k-means algorithm [17], the k-medoids algorithm [28], the k-modes algorithm [22], the k-prototypes algorithm [23], and the Gaussian mixture algorithm [40].

Dimensionality reduction techniques play a pivotal role in cluster visualisation and, in some cases, serve as a preprocessing technique to mixed-data clustering combinations. Prevalent techniques for dimensionality reduction in the realm of clustering include *principal component analysis* (PCA) [1] for numerical data and *factor analysis for mixed data* (FAMD) [42] along with *uniform manifold approximation and projection* (UMAP) [38] for mixed data. These techniques are frequently applied within the realm of clustering algorithms to facilitate a visual representation of high-dimensional data clusters in two- or three-dimensional spaces.

2.6 Natural language processing

The CESD framework facilitates the adoption of a very traditional NLP processing pipeline combined with both traditional machine learning classification algorithms, as well as transformer-based models such as BERT.

A number of preprocessing measures typically transpire in NLP applications prior to performing tokenisation. Such measures encompass, though are not limited to, linguistic translation and the mapping of contractions — each bearing its own set of implications for downstream NLP tasks [26].

Prior to conducting any form of computational analysis of a document, a procedure must be in place for converting unstructured text data into a format that is computationally intelligible. Each document essentially consists of an unstructured sequence of sentences, varying in length. While humans might find it relatively straightforward to interpret such unstructured text, computers are only capable of recognising character strings — rendering the need for text to be converted into a syntactic form that a computer can "comprehend." Tokenisation serves as one of the most pivotal steps in NLP tasks and involves the decomposition of a text corpus into smaller pieces, commonly referred to as *tokens*, which may be as rudimentary as words or as complex as subwords or phrases [26].

The data preprocessing stage subsequent to tokenisation in NLP tasks often involves a series of operations aimed at the *normalisation* and *filtering* of tokens. These preprocessing steps are pivotal for enhancing the effectiveness of downstream NLP tasks [4]. Extensive libraries exist for performing these steps on English text, which emphasises the benefits of performing translation as a pre-tokenisation processing step.

Normalisation encompasses a variety of sub-processes concerned with the standardisation of text. This may include the conversion of text to lowercase, rectifying spelling errors, and the application of *stemming* and *lemmatisation* techniques [29] — all of which should be executed with care. The conversion of tokens to lowercase might, for example, be deemed unnecessary for certain model architectures, whereas the correction of spelling errors might lead to malapropisms. Filtering generally refers to the elimination of irrelevant characters or tokens, which may include special characters, numbers, or even entire phrases that do not contribute to meaning in the task at hand [51].

Vectorisation methods, responsible for converting textual data into a numerical form, may be classified as being either *context-dependent* or *context-independent*. There is, however, a strong correlation between context-independent methods and count-based techniques (commonly exemplified by *bag-of-words* (BOW)-like approaches), and between context-dependent methods and embedding-based techniques (which include various types of word embeddings).

Context-independent text vectorisation methods provide a simple approach to converting textual data into a numerical form, largely ignoring the syntactic and semantic relationships between words. In the BOW model, for example, documents are represented as unordered sets of their words, stripped of any grammatical considerations [52]. Similarly, the *term frequency-inverse document frequency* (TF-IDF) which, while capturing the relative importance of a term within a corpus, remains oblivious to the surrounding textual context [47]. Techniques such as *word hashing* [36] and the *hashing vectoriser* [27] also fall within this scope, prized for their computational efficiency, but critiqued for the loss of semantic complexity [59].

Context-dependent methods represent a more advanced type of text vectorisation, designed to capture some of the complex relational dynamics between words — typically by employing sophisticated machine learning architectures. Although most context-dependent methods utilise embedding-based approaches, variations to count/presence based methods may introduce some degree of context by considering higher-order n-grams, rather than singular terms (unigrams), as document features. These n-grams represent sequences of adjacent words in their original order [51]. Typically, most context-dependent methods are, however, embedding-based. An embedding is a vectorisation technique for converting individual tokens within a corpus into vectors of real numbers. In this transformation, each dimension of the embedding vector corresponds to a latent feature inherent to the respective token [60]. GloVe [45] and Word2Vec [39] are well-known embedding algorithms. The pinnacle of context-dependent embeddings, however, is considered to lie in transformer-based models such as BERT and *generative pre-trained transformers* (GPT). These models have intricate architectures, reminiscent of attention mechanisms, and are capable of producing embeddings that are sensitive to both syntactic and semantic nuances [15, 46]. The BERT model, developed by Devlin *et al.* [15], has undeniably come to be recognised as a monumental advancement in the NLP landscape, particularly for text representation.

BERT employs a deep transformer architecture, originally conceived by Vaswani *et al.* [57], for pre-training language representations. Unlike traditional word embeddings, BERT captures contextual information from both left to right and right to left, thus being truly *bidirectional* as the name suggests. BERT has been pre-trained on large *Wikipedia* and *Bookcorpus* data sets invoking two primary tasks, namely *masked language modelling* (MLM) and *next sentence prediction* (NSP). BERT consists of multiple layers of transformer encoders stacked on top of one another, with the MLM being a cornerstone of its pre-training process — randomly masking a fraction of the input tokens and then predicting these masked tokens based on the surrounding context. This bidirectional training allows for the training of a unified model for a wide range of NLP tasks.

The *BERT for sequence classification* model is a fine-tuned version of the original BERT model, specifically tailored for tasks that involve classifying entire sequences of text into predefined categories by adding a classification layer on top of the transformer encoder. Unlike other fine-tuned models, the BERT for sequence classification model is trained to understand not only individual tokens, but also the relationships and dependencies between these tokens in each sequence — rendering it effective for tasks such as sentiment analysis, topic categorisation, and document classification [55].

3 The CESD framework

The framework proposed in this section is intended to guide a user during the process of classifying data records that exhibit both structured and unstructured free-text attributes into similarity classes. Although there are numerous techniques available in the literature for the various classification tasks within the realm of structured data or unstructured data separately, combining the two is not common practice.

Figure 1 contains a high-level overview illustration of the CESD framework, which comprises the following main components: A graphical user interface, a central processing component, and a database. By interacting with the framework through the graphical user interface, a user can provide input data, configure framework settings, and make decisions about the system that is being developed. The purpose of the database is to save pertinent data during the system development and deployment processes when executing the framework. The primary body of the framework is represented by its central processing unit, depicted in Figure 1 by the shaded region. This unit is responsible for all of the framework's computing, modelling, deployment, and analysis tasks.

There is an arrow between the graphical user interface and the central processing component, labelled *raw data*, which represents the user providing the raw input data to which the classification system is to be applied. Although it is assumed that the input data contain unstructured data (free-form text) as well as structured data (representing potentially both continuous and categorical attributes), no further assumptions are made in respect of the nature or domain of the data. Along with the raw structured and unstructured input data, the user is expected to provide the necessary labels as some of the components utilise supervised learning algorithms.

Communication between the user and the various framework components during the classification system development process is represented by the arrows between the graphical user interface and the central processing unit, labelled *configurations & decisions*. These arrows are drawn as solid lines, signifying that user involvement is *necessary* rather than optional for the framework to function properly during the system development process. The domain and type of the input data, the model architecture, and the feature set utilised during model

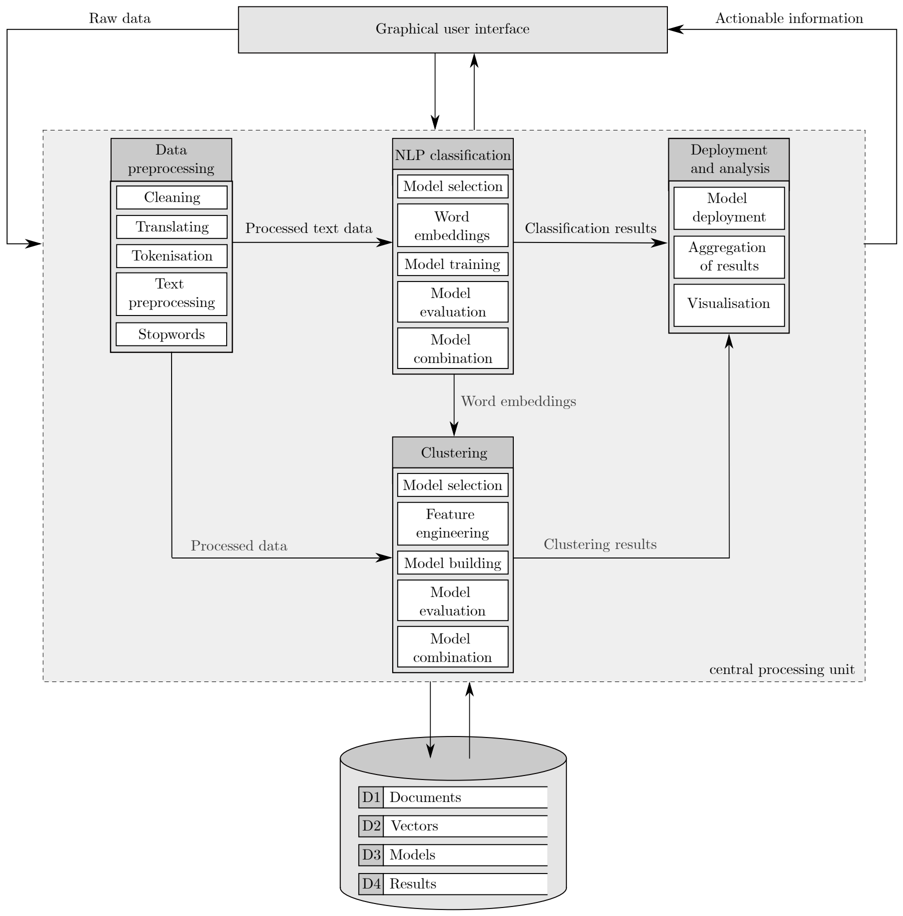


Fig. 1: A high-level overview of the CESD framework for classifying mixed data.

training all affect how well a machine learning algorithm performs in terms of performing classification tasks. Since there is no "one-size-fits-all" strategy that could conceivably be applied to all data sets, the objective of the framework is to *guide* the user through the process of constructing a classification system rather than to fully *automate* the process. Therefore, user involvement is required during the system design process.

The storage and retrieval of data and other resources during the system development process are represented by the arrows between the database and the central processing unit, which are labelled *data & resources*. In addition to preserving trained models for future use, this enables any component within the central processing unit to access the data and the results produced by other framework components.

The final classification results produced by the system are presented to the user *via* the arrow labelled *information* which leads from the central processing unit to the graphical user interface. These results ought to be in the form of *actionable information* that can be used by the system user to guide organisational or personal decision-making. That is, the system output results ought to be presented in a clear and understandable manner, such as in the form of a summary or visualisation. The remaining arrows between the various framework components represent the transfer of data and results between them.

Four components make up the framework's central processing unit. The first of these is called *data preprocessing*, and entails translating, vectorising and preparing the unstructured text data for use as input data to the subsequent modelling components. Furthermore, the structured data also have to undergo preprocessing, which includes dealing with missing data values and imbalanced data classes. The framework's model construction phase is collectively represented by two components — its *natural language processing classification* component and its structured *clustering* component. The framework provides two separate model development streams, which are illustrated by the pathways represented by the two arrows leaving the preprocessing component and later reuniting at the analysis component. The purpose of these two streams, which relate to the establishment of structured and unstructured data models, respectively, is to provide the user with freedom to select a model formulation that is best suited to his or her specific use-case. Due to the fundamentally different modelling approaches embodied in the two model development streams, the model development processes are further partitioned. The use of models specifically employed for NLP classification of the unstructured portion of the data constitutes the first stream, feeding into the second component. The NLP component also generates vector features based on the unstructured text, which are then sent to a third component along the second stream of structured data. This third component takes the structured data as input and applies different clustering techniques to these data according to an exploratory descriptive approach, as opposed to performing a classification task.

The two streams described above converge at a final component, called *deployment and analysis*. This component serves the purpose of guiding the user as to how the models established in earlier components should be applied to new, unlabelled data, and how the results produced by these models should subsequently be aggregated into a comprehensive summary or result visualisation which can be viewed at various levels of detail by the user.

4 Case study

This section is devoted to a discussion on the application of the computerised instantiation of the proposed framework to case study data pertaining to a production environment, which have been anonymised for the purpose of this article. The data pertains to the logging of errors in this environment, after which domain specialists categorise errors into different classes of severity — an indication of how time-sensitive the processes should be of resolving a particular error.

4.1 The data preprocessing component

The first step of the case study involved invoking the preprocessing component of the CESD framework, tasked with the ingestion of the raw data and their subsequent transformation into a usable data structure. The data subset utilised in this case study contained 43 features and approximately 415 000 records. In this data set, the feature of particular significance (the target feature) is *error severity.* This feature is categorical in nature, and may assume one of seven ordinal values that serve as an indication of the severity levels associated with reported errors. Moreover, the data set incorporates two unstructured text features, the *name* and the *description* of each error, which are intended for utilisation in the NLP classification component of the framework. As a result of many optional data fields in the logging of each error, a considerable portion of the features exhibited substantial quantities of missing values.

During the initial stage of preprocessing, the focus was directed towards the target feature which, as mentioned, denotes severity across seven ordinal categories, namely *01-safety relevant*, *02-breakdown occurs*, *03-permanent unsatisfactory*, *04-deficient*, *05-unsatisfactory*, *06-customer irritated*, and *07-customer noticed.* These seven classes were very unbalanced in terms of their occurrence in the data subset.

The objective was to derive a binary feature from these categorical values, serving as an indicator of *severe* or *non-severe* errors, with the former denoting errors potentially carrying substantial cost implications during production. The intent was to demonstrate the viability of automating the assignment of this binary value to future errors, thereby flagging potentially severe issues for closer scrutiny by a quality specialist. The need for such an approach arises from the fact that the logging of new errors initially involves assigning severity levels, oftentimes inaccurately. This leads to cases of severe errors initially going undetected as these inaccuracies are only later rectified by specialists. As a result, a need was identified to partition the error severity values ordinally into two distinct categories. Discussions with an expert in this domain corroborated our approach of finding a splitting point among the seven error severity values. The eventual decision of where to establish this split was reinforced by the skewed distribution of the classes, with a notable under-representation of more severe errors. A method was devised to categorise errors in such a manner that those in Categories 1–4 were deemed as *severe*, while those in Categories 5–7 were deemed as *non-severe.* This classification rationale aligns with the modified class distribution illustrated graphically in Figure 2.

Given the ordinal nature of this categorisation, it was considered the most balanced, meaningful binary representation attainable as a result of the inherent class imbalances. In order to avoid model bias towards the majority class, undersampling was applied to the majority class in pursuit of a balanced train-

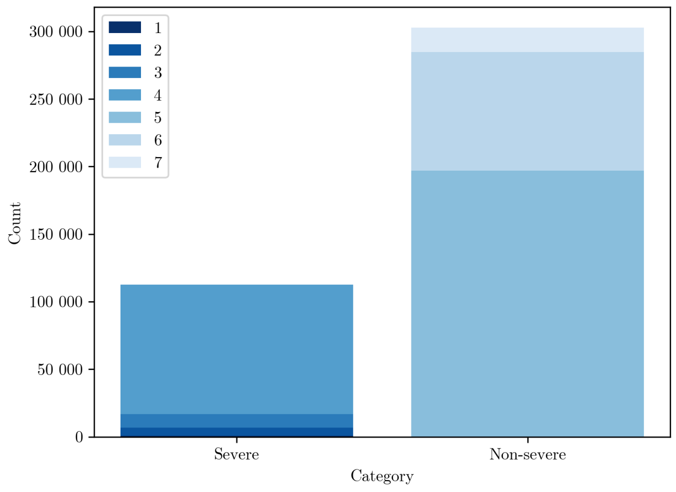


Fig. 2: The class distribution of the modified error severity feature in the data subset, showcasing the composition of smaller classes.

ing subset. This was coupled with stratified sampling to ensure the inclusion of representative proportions from the original classes.

The second phase of data processing pertained to the treatment of the unstructured data. Table 5 in the appendix contains a summary of eleven vectorisation combinations employed in respect of the case study data within the NLP classification component of the CESD framework. Due to the presence of two distinct and identifiable text features in the data set — the *name* and *description* of each error — all model permutations were applied individually to both cases. These combinations were undertaken with the aim of achieving optimal classification outcomes. These algorithms are combined with word vectorisation techniques originating from both the rudimentary count-based domain, as well as the more sophisticated BERT embedding approach. The last combination employs the transformer-based model BERT, accompanied by its distinctive vectorisation technique.

As for data set integrity, it is noteworthy that no instances of absent *name* or *description* features were observed. In accordance with the CESD framework, the initial stage of preprocessing (before tokenisation) is aimed at achieving translation. Approximately 30% of the ticket *name* and *description* features were in languages other than English (also employing space delimiters). It was therefore decided to translate all text to English due to extensive preprocessing libraries existing for English [21, 35]. As for tokenisation, the NLTK Python library was used to undertake word-level tokenisation on sentences for use by the first five combinations in Table 5. For the manipulation of text in Combinations 6 through 11, *Bert-base-uncased* (BERT's pre-trained base model trained on lowercase text) was employed.

Word level tokens necessitated additional preprocessing. Word clouds of unprocessed tokens revealed many application-specific acronyms, with the rationale for their inclusion being somewhat unclear. Some of these acronyms denoted the names of product components and processes involved in an error, therefore holding semantic relevance. The incorporation of these acronyms in the data could be up for debate, but the possibility that specific acronyms pertained to certain severities of errors instilled confidence in the potential of their inclusion manifesting discriminatory power. The word clouds also revealed the need for stopword removal and filtering. After translation, contractions were expanded, data underwent normalisation to lowercase characters, the elimination of special characters and punctuation took place, and stop words were eliminated.

The third phase of the data processing pertained to the structured data. A total of 38 features contained structured data and required additional processing (all the features other than the *error severity*, *name*, and *description* features). Initially, 31 features were categorical, six were numerical, two were date features, and two were text features.

In light of the requirement for only categorical and numerical feature use in the clustering component, it was decided to establish a numerical feature denoting the duration over which an error remained unresolved from the difference between two date features. Furthermore, a feature mistakenly categorised as a categorical attribute was engineered to a numerical feature, indicating the maturity of level of the product. Another high-cardinality categorical feature, containing many sub-features separated by commas, was engineered to a numerical count feature.

A summary of the transformed features may be found in Table 6 in the appendix.

The issue of missing values was addressed next. Given the modest proportion of missing values for most features, two rather rudimentary (but effective) methodologies were employed. For categorical features, the mode was used to impute missing values, while k-nearest neighbour imputation was utilised for numerical features. Furthermore, the high-cardinality feature, *Feature 14*, did not exhibit characteristics enabling the numerical encoding thereof. The feature was therefore excluded from the data stream feeding into the clustering component.

4.2 The NLP classification component

For the first ten classification combinations in Table 5, the embeddings derived by performing both TF-IDF and BERT vectorisation were utilised to train of the five distinct algorithms in the table. The task of hyperparameter tuning was performed, carrying out a grid search in conjunction with 5-fold cross validation. The hyperparameter values subjected to the grid search are outlined in Table 7 in the appendix. Default values were used during the tuning process for all other hyperparameters not expressly delineated in the table.

Not all hyperparameter combinations of the logistic regression model were feasible due to inherent constraints. The process of conducting the grid search involved 5-fold cross-validation, as mentioned. According to this approach, the training data were partitioned into varying subsets aimed at enabling iterative training in respect of varying selections of these subsets while validating in terms of the remaining data. This methodology renders it unnecessary to have a separate validation set.

The need, however, arose in the BERT model for a training and validation subset. During the validation process, the objective was to evaluate the model's performance on unseen data, rather than tuning its weights. It is important to underscore the reason why it was still necessary to have a further separate test set. This was due to the validation accuracy, although not affecting model weights, ultimately being used to determine appropriate values for parameters such as the optimal training epochs.

The BERT model, implemented in PYTHON as `BertForSequenceClassification`, was instantiated with parameters specifying the number of distinct classes and a dropout probability. Additional parameters included the *AdamW* optimiser and the establishment of a *linear warmup* scheduler. Ultimately, the training and validation cycles transpired across a total span of ten epochs, after which the test data set could be utilised to compute an unbiased accuracy metric.

4.3 The clustering component

The underlying objective of the clustering component is to extract meaningful insight from the data set. The focus therefore shifted to performing clustering on specific error severity classes in the data set. This allowed for discerning nuances such as the optimal number of clusters inherent to a given class. Analytic instruments, such as word clouds, were then be employed to glean insight into, and elucidate, the intrinsic characteristics of these clusters. The decision was made to perform clustering on the structured data in an isolated fashion, before performing clustering on a combination of the structured data and BERT embeddings, thus making it possible to draw comparisons between the different results obtained. The clustering was therefore partitioned into parts, with respective clustering combinations for each part as denoted in Table 1.

Table 1: Different combinations of distance metrics and clustering methods applied to the structured portion and the BERT embeddings of the error data set.

#	**Distance metric processing**	**Clustering**
1	Dimension reduction (FAMD)	k-Means
2	Recode numerical features	k-Modes
3	Dimension reduction (FAMD & PCA)	k-Means

In the context of clustering combinations applied to the structured data, the sheer size of the data set presented unique challenges. Most clustering algorithms employ distance metrics of relatively high time complexity, thereby limiting the number of viable options. Given the computational demands, coupled with data sensitivity concerns rendering the use of cloud-based computing infeasible, only two clustering combinations were implemented (the first two entries of Table 1). Combination 1, which utilised FAMD for dimensionality reduction, rendered the use of k-means computationally feasible. Moreover, Combination 2, the k-modes approach (tailored for the clustering of categorical data) utilised a matching dissimilarity metric of low time complexity. Both these combinations proved to achieve large ARI scores in the work of Van de Velden *et al.* [58].

In order to avoid the so-called cluster masking problem for combination 1, an outcome in which the process of dimension reduction conceals the inherent structure of the clusters, it was imperative to select an appropriate degree of dimensionality reduction. This was performed by utilising the elbow method to evaluate component variance, which indicated that 70% of variance was described by 693 components. As for Combination 2, the numerical features were subjected to discretisation before embarking on the clustering process. This involved binning numerical features into a set of N bins, with categorical labels being assigned based on the bin in which a value lies. The k-means algorithm was used to optimise the bin selection per feature, allowing up to five bins. The elbow method was again utilised to determine the optimal number of k-modes clusters for Combination 2.

In the context of clustering applied to the combination of BERT embeddings and the structured data, the sheer size of both data sets again presented unique challenges. In light of computational constraints, an alteration of Combination 1 was performed — utilising the 693 FAMD features and 50-PCA-reduced BERT description embeddings (attained through elbow selection at which 87% of the variance was described). This clustering method is denoted as Combination 3 in Table 1.

Furthermore, due to the addition of fifty numerical features, the application of Combination 2 was considered computationally infeasible as this would entail the addition of fifty categorical features to the structured data set — notwithstanding the fact that discretising the PCA-reduced BERT embeddings would almost inevitably lead to a significant loss of information.

4.4 The deployment and analysis component

In order to carry out a fair evaluation of the NLP classification models, they were exposed to the unseen test data set. Although the models were not explicitly trained on the validation set, it was used to make decisions about the values of hyperparameters or, in the case of the transformer model, to determine the number of epochs over which the model should be trained. Given the imbalanced nature of the testing set, the AUC score was deemed an appropriate evaluation metric. The corresponding scores achieved by the various model combinations in respect of the *name* and *description* features are summarised in Table 2.

Several observations can be made. First, it is evident that the BERT model exhibited superior classification performance in respect of both the *name* and *de-*

Table 2: Test AUC scores for different classification model combinations in respect of the *name* and *description* features of the processed error data set.

#	Vectorisation	Classification	Name	Description
1	TF-IDF	Extra trees	0.74	0.77
2	TF-IDF	Adaptive boosting	0.68	0.73
3	TF-IDF	Random forest	0.73	0.76
4	TF-IDF	Multinomial naïve Bayes	0.76	0.76
5	TF-IDF	Logistic regression	0.76	0.77
6	BERT-base-uncased	Extra trees	0.68	0.71
7	BERT-base-uncased	Adaptive boosting	0.67	0.70
8	BERT-base-uncased	Random forest	0.67	0.72
9	BERT-base-uncased (normalised)	Multinomial naïve Bayes	0.65	0.63
10	BERT-base-uncased	Logistic regression	0.69	0.72
11	BERT-base-uncased	Bert	**0.78**	**0.80**

scription features. Furthermore, logistic regression emerged as the top performer across both embedding combinations among the conventional models. It is important to underscore that the traditional models suffered a decrease in AUC when combined with BERT embeddings — possibly stemming from the inherent limitations of these models in terms of encapsulating the complexities of BERT embeddings. Upon closer examination of the classification outcomes achieved in respect of the *description* feature, a general trend was observed of a very slight increase in AUC scores for all but Combination 9. This phenomenon can likely be ascribed to the better encapsulation of class labels within the description fields of errors.

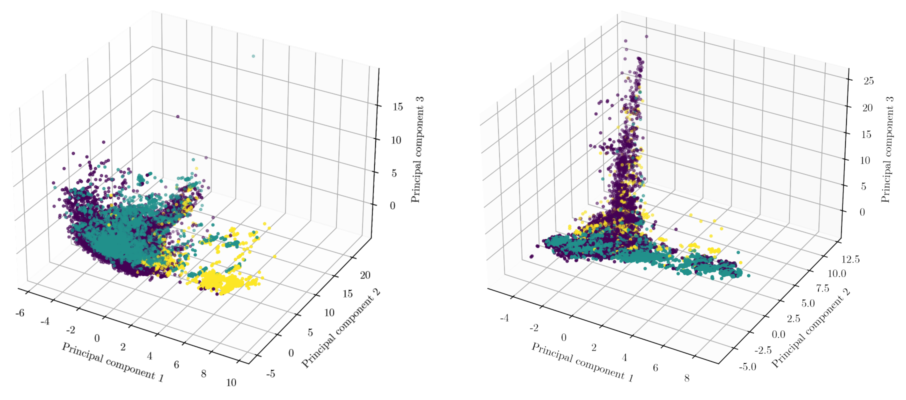
For clustering Combination 1, silhouette scores were used to determine the ideal values for k (the number of clusters). These silhouette scores may be found in Table 3 for the *non-severe* and *severe* error classes, respectively.

Table 3: Silhouette scores for different values of k for clustering Combination 1, and for the non-severe and severe error classes, respectively.

Clusters	2	3	4	5	6
Silhouette scores: Non-severe	0.122	0.119	0.055	−0.169	−0.179
Silhouette scores: Severe	0.157	−0.217	−0.187	−0.184	−0.183

For the *non-severe* error category, the silhouette scores indicated that values of $k = 2$ and $k = 3$ appeared to yield the best numbers of clusters — with $k = 2$ exhibiting marginal superiority. For the *severe* category, on the other hand, $k = 2$ emerged as the singular logical option due to its positive silhouette score. Upon evaluation of the silhouette scores for both categories, an overarching theme is discernable in which the clusters do not exhibit significant separation, particularly when the value of k exceeds 3. The k-modes clusters emanating from Combination 2 were therefore examined.

The silhouette score, which relies on measuring distances between data points and cluster centroids, is not directly compatible with the distance metrics employed by the k-modes algorithm. Therefore, a different approach was adopted. In the context of the k-modes algorithm, the cost denotes the cumulative dissimilarities between objects and their respective cluster modes — quantifying the dissimilarity between data points within a cluster and the central mode of that cluster. Notably, as the value of k increases, the cost tends to decrease. The rate of this cost decrease, however, starts to decrease at a certain juncture, resulting in the formation of an elbow in the graph. This elbow represents a starting point of diminishing returns with respect to the specific value of k. For both the *severe* and *non-severe* error classes, it was observed that the most appropriate number of clusters is $k = 3$. In order to gauge the quality of clusters, one may attempt to visualise them in a three-dimensional space. While a dimensionality reduction technique such as FAMD may not entirely capture the distance metric employed by the k-modes algorithm, it might yield some insight into cluster quality. Figure 3 contains a three-dimensional visualisation of the two pairs of clusters, for the *severe* and *non-severe* error classes, respectively.



(a) *Non-severe errors* (b) *Severe errors*

Fig. 3: Three-dimensional representations for clustering Combination 2, showcasing cluster separation for both classes of error severity.

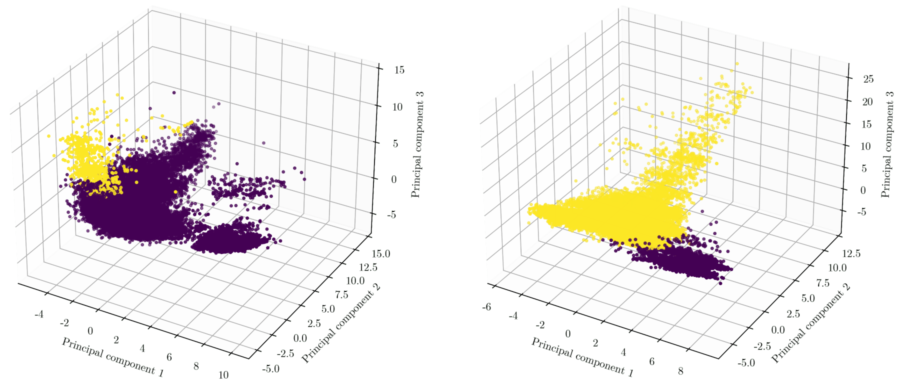
While these clusters do not appear to exhibit a substantial degree of separation, certain discernible distinctions are apparent. The *non-severe* error class seems to exhibit superior separation, with two larger clusters, depicted in purple and turquoise, appearing to be right next to each other. The smaller yellow cluster, on the other hand, seems to lie some distance away from the other clusters. As for the *severe* error clusters, the two larger clusters again exhibit some degree of separation, running in the directions of principal components 1 and 2. The third, smaller yellow cluster, does not appear to exhibit good separation from the other clusters. As previously noted, it is worth considering that the FAMD may not entirely capture the extent of variance accounted for by the k-modes distance metric — hence this depiction should not serve as the sole determinant of good separation. Due to a good amount of literature endorsing the overall effectiveness of the k-modes method both in terms of capturing categorical data clusters and being extended to mixed data, especially considering that a majority of the features were initially categorical, further investigation into these clusters was required [5, 58]. This task was carried out by observing the respective word clouds. The word clouds for the different clusters exhibited a prevalence for certain groups of tokens and acronyms, potentially providing insight to production specialists.

Lastly, for clustering Combination 3, silhouette scores were used to determine the ideal value for k, the number of clusters. These silhouette scores may be found in Table 4 for both the *non-severe* and *severe* classes.

Table 4: Silhouette scores for clustering Combination 3 and for different values of k.

Clusters	**2**	**3**	**4**	**5**	**6**
Silhouette scores: Non-severe	0.542	−0.109	−0.134	0.051	−0.113
Silhouette scores: Severe	0.121	0.023	−0.210	−0.213	−0.193

In both cases, the silhouette scores indicated that a value of $k = 2$ emerged as the singular logical option due to markedly positive corresponding silhouette scores. Upon evaluation of the silhouette scores for both categories, it is possible to discern an overarching theme according to which clusters do not exhibit significant separation when k exceeds 2, similarly to the observation made for the silhouette scores of clustering Combination 1. The reason for this may possibly be due to the complex and diverse information captured by the structured portion of the data. Figure 4 contains a graphical representation of the two sets of k-means clusters in the form of three-dimensional plots obtained upon having performed a further PCA on the 743 component embedding (the amalgamation of the 693 FAMD components and the 50 PCA components). This illustration demonstrates a commendable degree of separation — more than that of Combination 1 — between the clusters of both error classes, both of which exhibit two clusters greatly varying in size — a phenomenon particularly pronounced in the case of the *non-severe* error clusters and potentially shedding light on the large silhouette score obtained in Table 4.



(a) *Non-severe errors* (b) *Severe errors*

Fig. 4: Three-dimensional representations for clustering Combination 3 showcasing cluster separation for both classes of error severity.

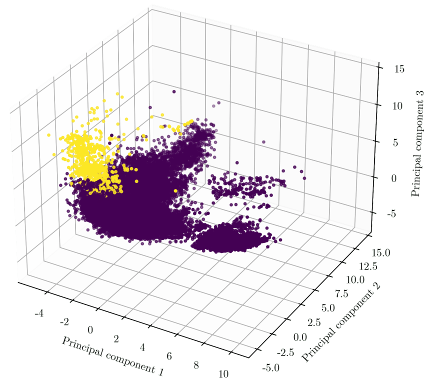
These clusters were again further investigated by observing the respective word clouds, which again revealed interesting trends in terms of clusters containing certain words.

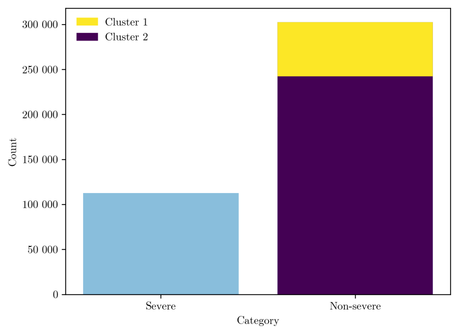
5 Conclusion

Based on the aforementioned analysis, the top-performing classification model, BERT, achieved an impressive AUC test accuracy of 80% for the target variable when applied to the *description* feature. Just short thereof, both the logistic regression and extratrees algorithms achieved AUC scores of 77%. These classification results attest to the efficacy of the CESD framework with respect to implementing NLP classification.

Moreover, the clustering component of the framework revealed valuable insights pertaining to the target variable of the data set. The approach based on clustering a combination of the BERT embeddings and structured data achieved a noticeable increase in silhouette score over the approach based solely on structured clusters — identifying clusters that varied considerably in size.

Although it is stressed that experts within the particular manufacturing domain may be able to interpret the corresponding word clouds appropriately, potentially extracting valuable practical insights from them, an attempt was made to uncover exemplars of these insights. The analyst is afforded considerable freedom when it comes to the task of how (s)he prefers to use the outputs of the clustering component. Based on words appearing only in specific clusters for the respective classes of severity, for example, an analyst might glean insight affecting the preprocessing steps pertaining to the NLP classification component. It may be argued that tokens which appear prominently in both classes be removed from the text corpus, as it might confuse NLP models. Alternatively, the analyst might choose to use the clusters as some sort of filtering criteria for model training, perhaps even determining stratified sampling weights within error severity classes (larger clusters might contain words that are proportionately more important). This suggestion is illustrated in Figure 5, in which the training undersampling process has been replaced with stratified sampling of the cluster ratios (from Combination 3) for the larger *non-severe* error class. The large silhouette score of the non-severe clusters emanating from Combination 3, being upward of 0.5, indicates that these clusters are very well separated, and may perhaps improve upon model generalisation if incorporated into the training set ratios.





(a) *Non-severe* clusters from Combination 3 (b) Stratified undersampling proposal

Fig. 5: Cluster-weighted stratified sampling proposal for training data.

References

1. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics **2**(4), 433–459 (2010)
2. Abiteboul, S., Buneman, P., Suciu, D.: Data on the web: From relations to semistructured data and XML. Morgan Kaufmann Publishers, Burlington (MA) (2000)
3. Aggarwal, C.C., Reddy, C.K.: Data clustering: Algorithms and applications. CRC Press, Boca Raton (FL) (2014)
4. Aggarwal, C.C., Zhai, C.X.: Mining text data. Springer, New York (NY) (2012)
5. Ahmad, A., Dey, L.: A k-mean clustering algorithm for mixed numeric and categorical data. Data and Knowledge Engineering **63**(2), 503–527 (2007)

6. Bibaeva, V.: Using metaheuristics for hyper-parameter optimization of convolutional neural networks. In: Proceedings of the 28th IEEE International Workshop on Machine Learning for Signal Processing. pp. 1–6. Aalborg (2018)
7. Bishop, C.M.: Pattern recognition and machine learning, vol. 4. Springer Science & Business Media, New York (NY) (2006)
8. Blei, D.M., Jordan, M.I.: Modeling annotated data. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 127–134. Toronto (2003)
9. Bramer, M.: Principles of data mining, vol. 2. Springer, London (2013)
10. Breiman L, Friedman JH, Olshen RA & Stone CJ: Classification and regression trees. Wadsworth International Group, Belmont (CA) (1984)
11. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**(1), 1–27 (1974)
12. Chomsky, N.: Syntactic structures. Mouton & Co, The Hague (1957)
13. Davenport, T., Kirby, J.: Strategies for remaining gainfully employed in an era of very smart machines. https://hbr.org/2015/06/beyond-automation (2015), last accessed 2022/12/01
14. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **1**(2), 224–227 (1979)
15. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4171–4186. Minneapolis (MN) (2019)
16. Elkin, P.L., Mullin, S., Mardekian, J., Crowner, C., Sakilay, S., Sinha, S., Brady, G., Wright, M., Nolen, K., Trainer, J., Koppel, R., Schlegel, D., Kaushik, S., Zhao, J., Song, B., Anand, E.: Using artificial intelligence with natural language processing to combine electronic health record's structured and free text data to identify nonvalvular atrial fibrillation to decrease strokes and death: Evaluation and case-control study. Journal of Medical Internet Research, **23(11)**, Manuscript e28946 (2021)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press, Cambridge (MA) (2016)
18. Gruetzemacher, R.: The power of natural language processing. https://hbr.org/2022/04/the-power-of-natural-language-processing (2022), last accessed 2022/05/06
19. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology **143**(1), 29–36 (1982)
20. Hastie, T., Tibshirani, R., Friedman, J.: An introduction to statistical learning, vol. 1. Springer, New York (NY) (2009)
21. Honnibal, M., Montani, I.: spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io/ (2017), last accessed 2022/06/03
22. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining. Data Mining and Knowledge Discovery **3**(8), 34–39 (1997)
23. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery **2**(3), 283–304 (1998)
24. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**, 193–218 (1985)
25. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)

26. Jurafsky, D., Martin, J.: Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, vol. 2. Pearson Education, Upper Saddle River (NJ) (2008)
27. Kanada, Y.: A vectorization technique of hashing and its application to several sorting algorithms. In: Proceedings of the 1990 International Conference on Databases, Parallel Architectures, and Their Applications. pp. 147–151. Miami Beach (FL) (1990)
28. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: An introduction to cluster analysis. John Wiley & Sons, New York, (NY) (2009)
29. Kazmaier, J.: A framework for evaluating unstructured text data using sentiment analysis. PhD dissertation, Stellenbosch University, Stellenbosch (2020)
30. Khaleghi, T., Murat, A., Arslanturk, S.: A tree based approach for multi-class classification of surgical procedures using structured and unstructured data. BMC Medical Informatics and Decision Making **21**, 1–12 (2021)
31. King, M.A.: Ensemble learning techniques for structured and unstructured data. PhD dissertation, Virginia Polytechnic Institute and State University, Blacksburg (VA) (2015)
32. Kuhn, M., Johnson, K.: Applied predictive modeling. Springer, New York (NY) (2013)
33. Lichtnwalter, R., Chawla, N.V.: Link prediction: Fair and effective evaluation. In: Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. pp. 376–383. Istanbul (2012)
34. Liu, B.: Sentiment analysis and opinion mining. Morgan & Claypool, San Rafael (CA) (2012)
35. Loper, E., Bird, S.: Nltk: The natural language toolkit. In: Proceedings of the Association for Computational Linguistics Interactive Poster and Demonstration Sessions. pp. 214–217. Barcelona (2004)
36. Luhn, H.P.: A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development **1**(4), 309–317 (1957)
37. Marr, B.: What's the difference between structured, semi-structured and unstructured data? https://www.forbes.com/sites/bernardmarr/2019/10/18/whats-the-difference-between-structured-semi-structured-and-unstructured-data/?sh=66574ff52b4d (2019), last accessed 2022/10/25
38. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. https://arxiv.org/abs/1802.03426 (2018), last accessed 2023/03/01
39. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Poster at the 2013 International Conference on Learning Representations, Scottsdale (AZ) (2013)
40. Ng, S., McLachlan, G., Yau, K.K., Lee, A.H.: Modelling the distribution of ischaemic stroke-specific survival time using an em-based mixture approach with random effects adjustment. Statistics in Medicine **23**(17), 2729–2744 (2004)
41. Nisbet, R., Elder, J., Miner, G.D.: Handbook of statistical analysis and data mining applications. Academic Press, Orlando (FL) (2009)
42. Pagès, J.: Analyse factorielle de donnees mixtes: Principe et exemple d'application. Revue de Statistique Appliquée **52**(4), 93–111 (2004)
43. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval **2**(1–2), 1–135 (2008)
44. 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., Édouard Duchesnay: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**(85), 2825–2830 (2011)
45. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543. Doha (2014)
46. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Tech. rep., OpenAI, San Francisco (CA) (2018)
47. Ramos, J.: Using tf-idf to determine word relevance in document queries. In: Proceedings of the 1^{st} Instructional Conference on Machine Learning. vol. 242, pp. 29–48. Los Angeles (CA) (2003)
48. Roberts, E.: Natural language processing: History. https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html (2004), last accessed 2022/12/03
49. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53–65 (1987)
50. Schapire, R.E.: A brief introduction to boosting. In: Proceedings of the 16^{th} International Joint Conference on Artificial Intelligence. pp. 1401–1406. Stockholm (1999)
51. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to information retrieval, vol. 39. Cambridge University Press, New York (NY) (2008)
52. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1), 1–47 (2002)
53. Shah, T.: About train, validation and test sets in machine learning. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (2017), last accessed 2022/05/02
54. Strekalova, Y.A., Bouakkaz, M.: Semi-structured data, pp. 1–3. Springer International Publishing, Cham (2017)
55. Sun, C., Qiu, X., Xu, Y., Huang, X.: How to fine-tune bert for text classification? In: Proceedings of the 18^{th} China National Conference: Chinese Computational Linguistics. pp. 194–206. Kunming (2019)
56. Tang, N., Zhang, R., Wei, Z., Chen, X., Li, G., Song, Q., Yi, D., Wu, Y.: Improving the performance of lung nodule classification by fusing structured and unstructured data. Information Fusion **88**, 161–174 (2022)
57. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31^{st} Conference on Neural Information Processing Systems. vol. 30, pp. 1–11. Long Beach (CA) (2017)
58. Van de Velden, M., Iodice D'Enza, A., Markos, A.: Distance-based clustering of mixed data. Wiley Interdisciplinary Reviews: Computational Statistics, **11(3)**, Manuscript e1456 (2019)
59. Weinberger, K., Dasgupta, A., Langford, J., Smola, A., Attenberg, J.: Feature hashing for large scale multitask learning. In: Proceedings of the 26^{th} annual International Conference on Machine Learning. pp. 1113–1120. Montreal (2009)
60. Zhang, L., Wang, S., Liu, B.: Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), Manuscript e1253 (2018)

A Appendix

This appendix contains tabulated information pertaining to the case study, referenced in §4.

Table 5: Different combinations of vectorisation and classification methods employed.

#	**Vectorisation**	**Classification**
1	TF-IDF	Extra trees
2	TF-IDF	Adaptive boosting
3	TF-IDF	Random forest
4	TF-IDF	Multinomial Naive Bayes
5	TF-IDF	Logistic regression
6	BERT-base-uncased	Extra trees
7	BERT-base-uncased	Adaptive boosting
8	BERT-base-uncased	Random forest
9	BERT-base-uncased (normalised)	Multinomial Naïve Bayes
10	BERT-base-uncased	Logistic regression
11	BERT-base-uncased	BERT

Table 6: Error data set feature information, after the processing of structured features.

Feature	**Mean**	**Std**	**Cardinality**	**Missing %**	**Data type**
Feature 1			689	0.06%	Categorical
Feature 2			3	1.56%	Categorical
Feature 3			5	1.23%	Categorical
Feature 4			11	0.04%	Categorical
Feature 5			5	0.05%	Categorical
Feature 6			604	0.00%	Categorical
Feature 7			9 286	6.36%	Categorical
Feature 8			23	0.03%	Categorical
Feature 9			119	0.03%	Categorical
Feature 10			4	0.71%	Categorical
Feature 11			4	0.00%	Categorical
Feature 12			11	0.00%	Categorical
Feature 13			421	1.73%	Categorical
Feature 14			74 198	0.38%	Categorical
Feature 15			40	0.03%	Categorical
Feature 16			7	4.33%	Categorical
Feature 17			3	14.18%	Categorical
Feature 18			2	0.00%	Binary
Feature 19			2	0.00%	Binary
Feature 20			2	7.10%	Binary
Feature 21			2	0.00%	Binary
Feature 22			2	1.01%	Binary
Feature 23			2	0.00%	Binary
Feature 24			2	7.09%	Binary
Feature 25			2	0.00%	Binary
Feature 26			2	20.98%	Binary
Feature 27			2	0.00%	Binary
Feature 28			2	0.22%	Binary
Feature 29	1.274	19.322		6.06%	Numeric
Feature 30	0.001	0.143		0.00%	Numeric
Feature 31	628.474	250.981		2.26%	Numeric
Feature 32	0.003	0.056		0.00%	Numeric
Feature 33	0.105	1.058		0.00%	Numeric
Feature 34	0.014	0.131		0.00%	Numeric
Feature 35	441.847	72.133		31.58%	Numeric
Feature 36	9.851	11.085		0.00%	Numeric
Feature 37	364.488	266.027		0.00%	Numeric
Name				0.00%	Text
Description				0.00%	Text
Error severity			2	0.00%	Binary

Table 7: The hyperparameter grid values considered for each model during the classification tasks performed on the error data set.

Algorithm	**Hyperparameter**	**Possible Values**
Extra trees	Number of estimators	5, 10, 15
	Maximum depth	None, 5, 10
	Maximum features	sqrt, log2, None
AdaBoost	Maximum depth of base estimator	3, 5, 10
	Number of estimators	5, 10, 15
	Learning rate	0.1, 0.5, 1.0
Random forest	Number of estimators	5, 10, 15
	Maximum depth	None, 5, 10
	Maximum features	sqrt, log2, None
Multinomial naïve bayes	Alpha	0.1, 1.0, 10.0
Logistic regression	C	0.1, 1.0, 10.0
	Penalty	l1, l2, elasticnet
	Solver	newton-cg, lbfgs, saga