THE UNIVERSITY OF CHICAGO

ADVANCED TEXT MINING TECHNIQUES AND THE APPLICATIONS IN CLINICS, CHEMISTRY AND MANUFACTURING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE PRITZKER SCHOOL OF MOLECULAR ENGINEERING
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY
CHAO ZHANG

CHICAGO, ILLINOIS
JUNE 2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Andrey Rzhestky, for guiding me through my scientific journey. Dr. Rzhetsky has introduced me to the fields of medical data science and text mining, and his tremendous support has helped me to accomplish all that I've accomplished during my PhD. He has also given me great freedom to explore a wide range of possibilities during my journey. I would also like to thank all the Rzhetsky lab members for helping me with my projects, especially Hanxin Zhang, Gengjie Jia, Bohdan Khomtchouk, Yanan Long, Rachel Melamed, and Chengjian Shi.

I would also like to thank our collaborators, including Dr. Olufunmilayo I. Olopade, Dr. Pedro Lopes, Dr. Risi Kondor, Dr. Dezheng Huo, and the members in their labs, as well as collaborators from AIMS in Rwanda, Apmis in Nigeria, and float clinics from the Congo. I'm also grateful to my internship advisor, Sthitie Bom at Seagate Technology, who provided invaluable guidance and support during my year-long internship.

I would be remiss to not include special thanks to my University of Chicago friends and colleagues for helping me adapt to the new – sometimes overwhelming, always stimulating - UChicago environment. And, of course, I could not have made it without the support and understanding of my parents. Finally, I would like to thank my wonderful and amazing fiancée Jun Wang, who stuck with me through the happiest and the hardest times. Her companionship and encouragement have been crucial for both my well-being, and that of the project's.

ABSTRACT

In the past decades, computer science has developed with astonishing rapidity, and this has been accompanied by an equally shocking explosion of big data. In response, text mining methods have likewise made similar progress and are being applied to various areas like document classification and spam detection. However, existing text mining methods or models have mostly focused on a single task solution; this is no longer sufficient, as the need for large-scale general applications continues to grow. For example, in the clinical context, there have been many diagnosis models for specific diseases, such as cancers, but very few models for the more general and common diseases, especially in lower-resource areas. Using the latest text mining models, we have developed a general diagnosis assistant system, which we will test and deploy in African countries. Additionally, we have built a simple application for the system, which consumes negligible computing resources and is not dependent on internet availability. Text mining can also create values in the field of chemistry. The language of chemistry strongly affects how chemists think about chemistry. We investigated the possibility of improving standard chemical structure representation using relational learning and text mining methods. The current representation system was developed in 19th century and it no longer fully reflects the properties of chemical compounds that were discovered or developed later. We proposed a new representation method and proved that it does a better job in inferring biochemical properties. At the same time, text ext mining can be further extended to other fields like manufacturing. While the manufacturing sensor data can be transformed into the same format as texts, we applied text mining methods to perform a diagnostic prediction for wafers in Seagate hard drive factories. In summary, we explored multiple text mining techniques and applied them on clinics, chemistry and manufacturing. In the following chapters, we will discuss the details of these techniques and their applications in more detail.

CHAPTER 1

INTRODUCTION

The explosion of big data, in which a large portion is in the form of text, has led researchers to begin developing computer algorithms to deal with text data: Text mining allows researchers to mine useful information from large-scale, free texts [104, 127].

The idea of text mining was first proposed by Hans Peter Luhn in 1958 in the IBM Journal[68]; this is also recognized as one of the earliest artificial intelligence methods. Luhn's paper described a method to extract words from documents and use the frequency statistics to make document classification.

After text mining was proposed, much fruitful research followed. In 1960, Maron developed a technique for indexing literature with a mechanized library system [70]. Later, in 1995, Feldman Ronen et al. developed the first Knowledge Discovery in Text (KDT) model at the first International Conference on Knowledge Discovery and Data Mining [26].

There have been quite a few milestones in text mining technique development in the past decades. Bengio et al. proposed the first neural network language model in 2003 [7], using a neural net with one hidden layer to predict the next word in a sentence. Later in 2008, Collobert and Weston demonstrated that word embedding, trained on a large corpus, contains meaningful information and can be used on multiple downstream tasks [16]. At the time, lack of computational power at that time hindered the speed of these developments; but a breakthrough occurred in 2013, when Mikolov et al. proposed Word2Vec [74], which was an extremely simplified neural network model that proved easily well-executed with contemporary computers of that time. Since then, Word2Vec has been proven to be surprisingly powerful. Its success has primarily been the result of its training strategies. There are two strategies for Word2Vec: 1) Continuous bag-of-word, which uses the surrounding context of the target to predict the target word, with the orders of the context words ignored, and; 2) Skip-gram, which switches the input and output of the continuous bag-of-word strategy. It did not take long after Word2Vec's success for a few alternative models to surface. Released in 2014, GloVe [83] emphasized the geometric relations between the embedded words. Fast-text [51], developed in 2016, represented n-grams within words to include an etymological analysis. Elmo [84], proposed in 2018, instead generated a contextualized embedding in which the word embedding also includes information for the surrounding words.

Transformer [107], proposed in 2017, initiated a new era of deep learning and text mining due to its ground-breaking performance in various tasks such as classification and machine translation [63]. Transformer was first built as a sequence-to-sequence model, especially designed for machine translation. It consisted of an encoder and a decoder module, both of which were made of stacked layers of multi-head self-attention and feed forward networks. Self-attention is a specific type of attention mechanism [5] which describes each word in the input sentence's contribution to other words in the same sentence. Transformer's self-attention mechanism allowed the model to process the entire sentence simultaneously. In contrast, previous models such as recurrent neural network (RNN) and long short-term memory (LSTM) models took the input words one by one and were constrained by parallel training.

Given Transformer's ground-breaking performance and influence, there have been numerous attempts to build on and optimize the original model. Some variant models modify its multi-head self-attention module to make it more efficient or allow it to deal with super long sequences [54, 121]; some variants change the overall architecture to better fit into classification or generation tasks [90]; for instance, BERT [18] uses segment embedding to achieve a deeper word representation, provides large, pre-trained models, and pushes the entire text mining a step forward. Other variants focus on applying these models to other fields, such as audio, images, or videos [19, 101]. For example, Vision Transformer [20] is a break-through in the field of computer vision that outperforms the previously dominant convolution neural network models [60].

Along with such model development, more inspiring research was performed regarding different kinds of text mining tasks [2, 95]. The most common text mining tasks include: 1) text representation and encoding, such as text preprocessing and vector space models; 2) text classification with naive bayes, KNN, decision tree, and SVM; 3) text clustering with Hierarchical Clustering algorithms, k-means, Probabilistic Clustering, and Topic Models, and; 4) information extraction with Name entity recognition (NER), Hidden Markov Models, Conditional Random Fields, and Relation Extraction. These applications are briefly introduced in the following paragraphs.

Text preprocessing and encoding are the earliest text mining applications, while also outlining the first steps for all kinds of advanced text mining tasks, because computational algorithms can only further process the data when the texts are preprocessed into numerical representations. Typical preprocessing methods include: tokenization, which involves breaking a character sequence up into pieces [113], filtering such as stop-words removal [96], Lemmatization for morphological analysis of the words [78], and stemming to obtain the stem (root) of derived words [67]. While these methods can only handle the simplest tasks, they formed the foundations for the entire research field of text mining and natural language processing. Text encoding is another key in the progress towards making it possible to accomplish complex tasks with machine learning algorithms. The first text encoding algorithm is Vector Space Model[94], in which documents are represented with numerical vectors and each word is represented by a number indicating the word's importance in the document.

Text classification is the most widely studied and implemented text mining application. Text classification's strategy is to assign predefined classes to text documents [77]. One example of text classification is email spam filtering. While spam filtering is a binary classification task, there are other types of classification model with multiple labels [34]. Given the broad applications, many algorithms have been developed and applied for text classifications. Typical text classification methods include: Naive Bayes classifiers [71] such as Multi-variate

Bernoulli Model [61] and Multinomial Model [77, 72]; nearest neighbor classifiers which use distance-based measures to perform the classifications such as k-nearest neighbor algorithm [36]; decision tree classifiers [29], which perform the predictions by traversing a tree structure; and support vector machines [17], which maximizes the separation boundaries found in classification tasks.

Text clustering is the task of locating groups of similar texts in a collection of texts and is one of the main techniques used when organizing documents [46]. Clustering differs from text representation in terms of interpretability – the clusters are meaningful and directive. Typical text clustering methods include: hierarchical algorithms, including both top-down and bottom-up architectures [79]; k-means clustering, which partitions documents into k clusters [10] and its derivatives algorithms with higher efficiency [3], and; topic models which create a probabilistic generative model for text documents, such as Probabilistic Latent Semantic Analysis [40] and Latent Dirichlet Allocation [9].

Information extraction (or information retrieval) involves automatically extracting structured information from unstructured or semi-structured text. Information extraction is the predecessor of natural language understanding, but its goal is more specific; information extraction seeks to extract structured and specific target information [41]. There are two main types of information extraction tasks: named entity recognition and relation extraction. Named entity recognition locates named entities such as organization or person names in free texts [62].Relation extraction locates the semantic relations between entities in texts [53]. Typical information extraction methods include: hidden Markov models, which consider neighboring words' predicted labels, comparing them by using traditional probabilistic techniques [89][91], and conditional random fields which use probabilistic models for sequence labeling [59].

Besides these major tasks, text mining techniques are also applied to many other fields, such as ontologies [117] and question answering [4]. To fully utilize the power of these methods, we applied text mining algorithms – including Word2Vec and transformer – to the field of clinics, chemistry, and manufacturing. Solving these text mining problems requires both the techniques and the datasets. Thanks to the support from my advisor and collaborators, we got access to the right datasets for these problems.

In the clinical context, the problem is that lower-resource areas are suffering from severe disease burden. Even if many of the diseases like malaria have disappeared from the United States, they are still among the top killers in African countries. Using the text mining techniques trained on large medical literature and clinical notes, we were trying to find a way to ease such disease burdens – even if only a little bit – and save lives. Additionally, we also developed an absolute breast cancer risk prediction model for Nigerian women based on a dataset including 1,811 breast cancer cases and 2,225 control cases from the Nigerian Breast Cancer Study.

In the chemistry field, we developed a new naming system to better represent the biochemical compound properties. In collaboration with the King Laboratory at the University of Cambridge, we extracted the most common substructures from chemical compounds with AI, and used these substructures to name or represent the compounds. Using text mining, we also identified how current researchers name the chemical compounds from academic literature. By comparing these two naming systems, we proved that AI method did a better job than the current naming system.

In the manufacturing sector, we applied state-of-the-art text mining techniques, including auto-encoder and transformer models, to soft sensor datasets from Seagate Technology, which is one of the largest hard drive manufacturing company in the world. Seagate had a much larger database than any other publicly available soft sensor data. we transformed the sensor data into the same format as embedded texts. In this way, text mining techniques, such as transformers, could be applied with minor modifications. This application not only achieved great performance for soft sensor problems, but also advanced text mining techniques' applicability.

The following chapters provide more in-depth details and discussion about each of the above projects. Chapter 2 examines the process of developing the diagnosis tools for lower-resource areas; Chapter 3 discusses the projects for chemical naming with AI, and Chapter 4 shares the results and the methods for soft sensing in Seagate. Finally, Chapter 5 summarizes my PhD work and the possible future work it enables.

CHAPTER 2

CLINICAL APPLICATIONS

Lower-resource areas in Africa and Asia face a unique set of healthcare challenges: the dual high burden of communicable and non-communicable diseases; a paucity of highly trained primary care physicians, and; a lack of reliable, inexpensive internet connections, which makes many digital systems for recording and storing electronic medical records (EMRs) inapplicable.

To begin to address these healthcare challenges, we determined to leverage the availability of cellular phone networks and implement an open-access, computerized diagnostic system that would operate as a smartphone application in the absence of reliable internet connections. We designed our system to help primary care physicians in lower-resource areas to generate, record, and share accurate diagnoses in real-time using a centralized database. The system would be adaptive, aiming to arrive at diagnoses with a minimum number of questions. As a mobile application, it would be lightweight and use negligible computational resources.

Our application would provide primary care physicians in lower-resource areas with a tool that enabled faster and more accurate diagnoses. This same application could also be leveraged to automatically populate local or national EMR systems.

2.1 Light-weight Diagnosis Assistant – Phase 1

2.1.1 Introduction

Diagnostic errors affect an estimated 12 million Americans each year [97]. There are multiple reasons for this large number; one reason is physician fatigue and burnout. However, a deeper reason lies in the heavy-tailed pattern of disease frequency distribution [52]. Most diseases are in the very low prevalence area of overall disease distribution, thus, physicians who encounter too few instances of a complex pattern are less likely to recognize it. While this problem affects all physicians, it is exacerbated in resource-poor areas, where primary care providers function as generalists due to high demand and shortages of medical personnel.

Electronic Medical Records (EMRs) are rapidly growing in both volume and ubiquity, and this, along with the development of high-performance computing systems, is helping modern computers to outperform humans in many focused, diagnostic tasks. Attempts to develop efficient, computer-aided diagnosis support systems (DSSs) have been around since the 1960s [75, 76, 112]. While experimental DSSs that focus on single diseases have been successful in past decades, DSSs for general disease diagnoses have not flourished. There are a few general DSSs available commercially, like Isabel, DXplain, and GIDEON [6, 8, 106], but their clinical use remains limited due to their poor specificity and adaptiveness [12].

EMRs are considered essential to monitor and improve patient well-being [35]. Epic is the currently dominant clinical software company. Epic offers a clinical software suite that includes a communication portal and a set of specific diagnosis systems. Johnson et al. reviewed Epic's software comprehensively [50] and concluded that Epic provides a high-quality EMR system for collecting and managing accurate medical records. However, it comes with substantial costs: US physicians using Epic tend to complain about the work burdens stemming both from its module design, as well as from the data entry involved to maximize billing, manual data coding, expensive, ongoing vendor support, and difficulty with introducing third-party extensions. This inability of existing systems to meet the needs of lower-resource countries calls for a free, lightweight, network-free application that can perform documentation of patient encounters (creating EMRs) for general health problems and the tropical diseases that are prevalent in lower-resource settings – with both speed and accuracy.

On the other hand, to our knowledge, there is no practicable diagnosis system for most lower-development areas, such as Nigeria. Based on the above information, there are demands, and therefore opportunities, for a universal, efficient, and adaptive diagnosis system. Before laying out plans for designing such a system, we formulated a "desiderata" of such a system's properties as following:

1. The system should help physicians arrive at diagnoses efficiently and accurately, with a minimum number of questions/tests.

2. The system should be easing physicians' work burdens, rather than adding to them. For example, the system should make it easier to record and unify patient-specific data.

3. The system should be able to collect patient data and be adaptive (learning from data overtime).

4. The system should take into account event frequencies and make use of probabilistic data.

5. The system should be able to learn the location-specific properties of a patient population and the accompanying disease distribution.

6. The system should be able to assist physicians with patient status follow up (for example, checking if a patient still has a fever and alerting the physician, if required).

7. The system should be able to operate in resource-poor settings, such as Nigeria, where internet access is limited. Thus, text messages are the most reliable main mechanism of communication between the medical record server and physician/patient.

8. The system should provide an obvious benefit to patients such as diagnosis accuracy, efficiency, and friendly following-up.

Our goal is to develop a system that would combine machine learning, computerized adaptive diagnosis [32, 33], mobile application software, and EMRs, allowing physicians to accurately and easily capture patient-specific documentation across a broad spectrum of diseases. The system would incorporate active-learning and functionality without internet connectivity. Paired with a centralized server which would receive messages from mobile terminals and dispense updated information to physicians, this system would provide a new Decision Support System (DSS) paradigm. If successful, this open-source software could be deployed globally.

To this end, we developed a computerized AI physician assistant system to aid primary care physicians in recording symptoms and diagnoses of a broad range of diseases, automatically recording patient encounters in a structured form. Learning from large US medical datasets, and using adaptive learning procedures and online learning algorithms, our system prompts primary care physicians with detailed, yet concise, suggestions of options to record symptoms, follow-up tests, and treatment options. Our light-weight mobile application uses negligible computational resources and is able to transfer data via text messages, as its intended users are primary care physicians in lower-resource areas with unstable internet connectivity.

The following sections discuss our system's design and current operation processes in greater detail.

2.1.2 Materials and Methods

The datasets we use come mainly from three sources: UMLS at NIH, MarketScan from IBM, Stanford University's DeepDive Open Datasets, and a knowledge database of disease-symptom associations generated with data from New York Presbyterian Hospital admitted during 2004[111]. They are three different types of data for different usage:

DeepDive Open Datasets are large text corpora containing natural medical language. We are using one of DeepDive's datasets called BioMed Central, which comes from an STM (Science, Technology, and Medicine) publisher of 274 peer-reviewed open access journals. We used this corpus to discover the correlation between natural language and unified medical terms, thus enabling our system to recognize the natural language inputs.

UMLS is an abbreviation for Unified Medical Language System, affiliated to the US National Library of Medicine (NLM) — National Institutes of Health (NIH). The UMLS is a set of files and software that brings together many health and biomedical vocabularies and standards to enable interoperability between computer systems. The UMLS has an API that enables us to run a query and associate between different codes, such as the International Classification of Diseases (ICD) and the Concept Unique Identifier (CUI) codes for diseases, as well as between codes and natural languages. Also, some medical textbooks are using UMLS codes to record the associations between diseases and symptoms. These textbooks provide information needed for the diagnostic process. Our database of disease-symptom associations is based on UMLS, in which both diseases and symptoms are represented by a unified code.

MarketScan are research databases collected by IBM. These databases include electronic medical records for millions of patients The data are contributed by large employers, managed care organizations, hospitals, EMR providers, as well as Medicare and Medicaid. Through the MarketScan databases, we have patients' age and gender information, and thus, we have distribution information, providing a prior hypothesis for diagnosis. These electronic health records contain not only the correlation between diseases, but also the correlation between diseases and medication. This information can be very useful after the diagnosis process.

Multiple journal reviews have covered a number of computerized diagnosis support systems developed during the past decades [24, 57, 81, 93]. These systems implement many machine learning methods, such as naive Bayes, decision trees, and support vector machines. Their applications vary among different disease categories including heart diseases, diabetes, cancers, and depression. Fatima, et al has summarized the machine learning algorithms used in diagnosis support systems [24], including Bayes network, Support Vector Machine, Naive Bayes, decision tree, deep learning, and others. These systems are used in different disease categories including heart, diabetes, liver, dengue, and hepatitis. Average accuracy is around 80 percent. Naive Bayes and Support Vector Machines appear to be the most accurate and widely-used algorithms.

However, existing models are all trained from labeled patient data, which means that the most accurate machine learning model is based on accurate data points from human physicians. This places much more pressure on physicians to collect and validate the data. Furthermore, machine learning models are constrained within a narrow spectrum due to both the training data's limited size and model complexity. Traditional machine learning diagnosis methods also require a complete set of input information from patients, which is sometimes hard to acquire and prone to errors.

We designed the system with consideration to the disease burden in low-resource areas. Prior to the start of our system's development, we interviewed many primary care physicians in Nigeria, as summarized in Table 2.1. Most of the physicians still utilize hand-written patient records and believe that a mobile application would ease their labor and improve effectiveness. Most of our interviewees preferred the Android platform. To increase user efficiency, our application works in an adaptive manner that requires only the minimum number of questions or tests to give suggestions, all based on natural language processing and decision tree algorithms.

We used Word2Vec [74] to process the medical literature and clinical notes. Word2Vec is a natural language processing method that represents words as numerical vectors and learns the associations between words that appear closely in the corpus. Word2Vec is a combination of two methods: continuous-bag-of-words (CBOW) and skip-gram model.

CBOW is developed to predict the probability of a word given a context. With a shallow neural network, CBOW is trained with a large corpus and learns a relation between words, which are represented by a set of vectors. The input layer can be one or more words, and the output layer is a vector representing a set of words with probabilities. The highest

Table 2.1: Survey results from physicians in Nigeria. Most of the physicians who took our surveys reported using hand-written recordings, which are not well-organized and sometimes even hard to read. They indicated a preference for an Android-based application, if provided. In the meantime, most of them think that a single mobile application would not be enough, reporting that the entire diagnosis and treatment lifecycle requires a full software suite.

Topic	**Options**	**Responses**	**Comments**
Preferred recording method	Hand-written Digital Total	43 6 46	18 doctors complained about hand-written recording
Preferred electronic method, if provided	Android App Digital Total	39 6 46	Some primary care providers already have access to both mobile and web apps
Preference between single mobile application and a suite	Single app Suite of apps More than mobile apps Total	6 10 6 22	This is a follow-up survey and fewer doctors responded

probability corresponds to the most relevant word, as shown in Figure 2.1A. CBOW takes the average of the contexts surrounding a word (as seen above in the calculation of hidden activation). For example, ”Apple” can be both a fruit and a company, but CBOW takes an average of both the contexts and places ”apple” between a cluster for fruits and companies. The skip-gram model, as shown in Figure 2.1B, takes several different target variables and output a set of probability combinations. It allows words to learn from different contexts (such as fruit apple and company apple) but takes more computing resources.

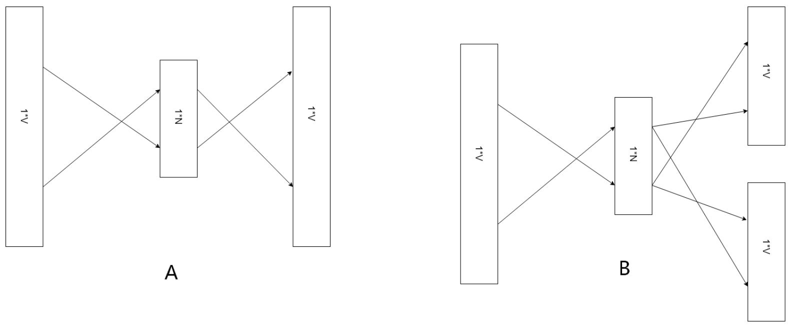


Figure 2.1: Plate A shows a schematic representation of the continuous-bag-of-words method. Plate B shows a schematic representation of the skip-gram model. V represents the total number of distinct words.

We use skip-gram to predict the relationships between natural language and medical terms because it's been shown to perform better when representing words with low frequency. Skip-gram extracts the representing vectors to relate the natural input from patients with our medical codes. Also, it updates the weighted disease-symptom matrix with the relationships between diseases and symptoms.

By automatically learning relationships from large corpora, Word2Vec is able to capture the association between words. We assumed an association between words with a distance of less than 50 words, which is about the length of two sentences. The model remembers and learns the quantified co-existence relationship between medical terms. The advantage of Word2Vec is that it captures the contribution of synonyms. This is important because the disease manifestation and symptoms can vary greatly in the corpus.

As shown in Figure 2.2, we extracted the associations between medical terms and used them to construct a decision tree model [11, 88].

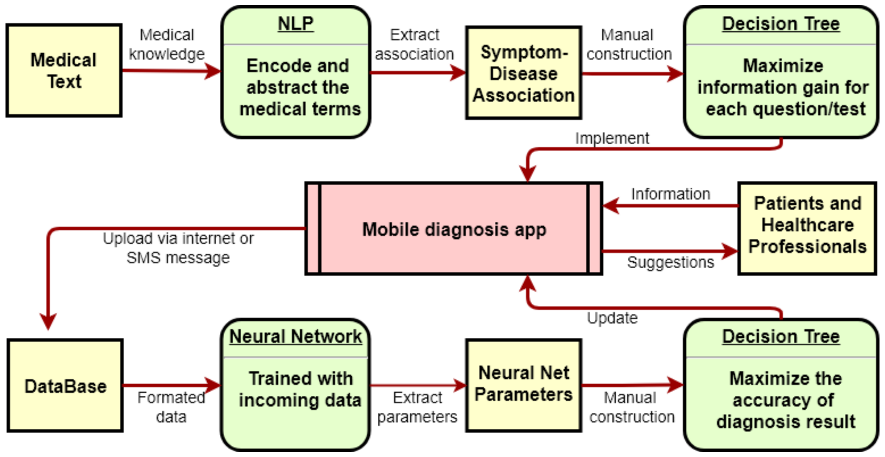

Medical Text
Medical knowledge
NLP
Encode and abstract the medical terms
Extract association
Symptom-Disease Association
Manual construction
Decision Tree
Maximize information gain for each question/test
Implement
Mobile diagnosis app
Information
Patients and Healthcare Professionals
Suggestions
Upload via internet or SMS message
Update
DataBase
Formated data
Neural Network
Trained with incoming data
Extract parameters
Neural Net Parameters
Manual construction
Decision Tree
Maximize the accuracy of diagnosis result

Figure 2.2: DSS development workflow. Word2Vec extracts associations between medical terms (such as symptoms and diseases) from the medical text corpus. These associations are then used to construct a decision tree model for adaptive diagnosis, and the model is implemented into an Android application.

After extracting the associations between symptoms and diseases, we constructed a decision tree to mimic the healthcare visit by generating questions iteratively based on previously collected information. Drawing from a symptom bank, it generates new questions by maximizing the information gain, either after the patient's initial information input or the last question. The system maximizes the information gain by finding the symptom item that separates the candidate diagnosis into two groups with similar size, which minimizes the statistical entropy.

In practice, patients are asked to provide their most concerning symptom as a trigger, which then corresponds to the root of the decision tree. Next, the system calculates the expected information gain for each symptom or lab test, and then adaptively decides what to ask in the next tree node. This process runs through several iterations and, when enough information is gathered, the system is then able to output diagnosis suggestions. This methodology minimizes the number of necessary questions or tests, thus expediting the diagnosis process.

After collecting additional data, the system updates the parameters with a single-layer, artificial neural network [73], which allows the assistant system to further improve as data accumulates. The entire code for the model and the Android application is open-source and can be found at `https://github.com/imechaozhang/Mead-Android-App`.

We developed a mobile application based on this model. Figure 2.3 shows the mobile application's workflow. The application allows physicians either to directly input both patient-specific symptoms and their diagnosis, as well as query the application for suggestions. The application automatically saves everything locally in an SQLite database, which synchronizes with the remote server when internet connectivity is available. To optimize data storage scalability and accessibility, the remote server uses MongoDB, an open-source, document-based database. In order to function in lower-resource areas, where internet connectivity may be unstable or non-existent, the application can communicate with a remote database via SMS messages. When physicians choose to ask the application for diagnostic suggestions, the application takes the patient's basic information and their trigger symptoms and perform an adaptive diagnosis by asking about symptoms or tests with the largest information gain. Within a few steps, the application provides suggestions regarding the diagnosis and prescriptions, as well as further, possibly necessary lab tests. Physicians only need to click on a few buttons to complete the entire process.

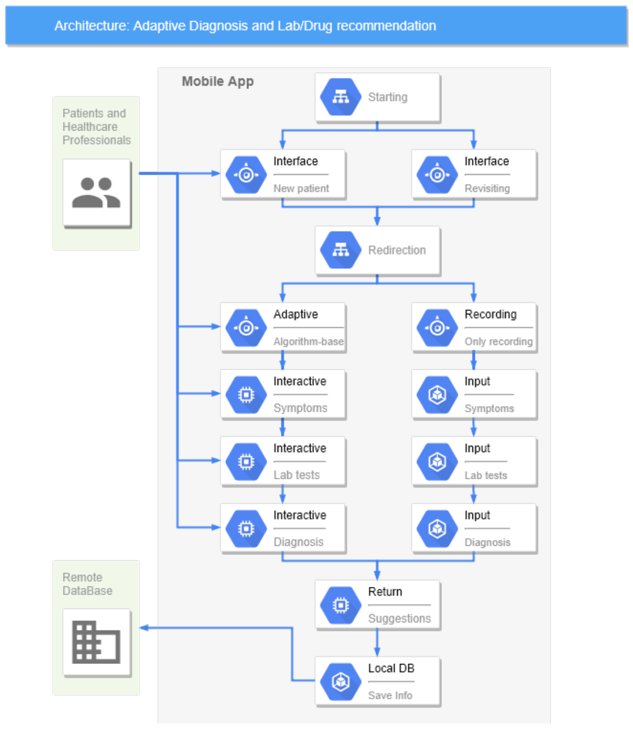


Figure 2.3: The mobile diagnosis application's architecture. The application interacts with both patients and physicians and optimizes the diagnostic process using an adaptive process based on a decision tree algorithm. All patient data are stored both locally and in the Cloud, so revisiting patients do not need to re-provide basic information. The data collected is encrypted before it's sent to remote databases to ensure patient information security.

While our application can make recommendations to physicians from just a few clicks, that's all they are – suggestions. Physicians are under no obligation to accept those suggestions, and indeed, a mechanism exists for them to reject the application's suggestions and type in their own diagnosis. We assume machine learning can never be 100 percent accurate and – importantly – our machine learning algorithm learns from mistakes.

2.1.3 Results

Using the MarketScan datasets, which include billions of medical records from all across the US, we were able to find some useful information. The datasets contain the age, gender, disease, and medication of each patient over time. We extracted information on the age and gender distribution across different diseases and examinations.

To begin, we summarized the MarketScan datasets' population distribution. Figure 2.4 shows the distribution of age/gender for all ICD codes, including both ICD9 and ICD10 codes for disease diagnosis and examinations. Overall, the population of female adults is slightly larger than males; however, `PopulationPyramid.net` shows that, below the age of 45, the male population slightly outnumbers the female. This indicates that the MarketScan datasets are slightly biased. There is a huge drop in the population at the age of 65. This is also due to the bias in the data. At the age of 65, the retirement age in the US, people usually switch their health insurance and the data becomes unavailable.

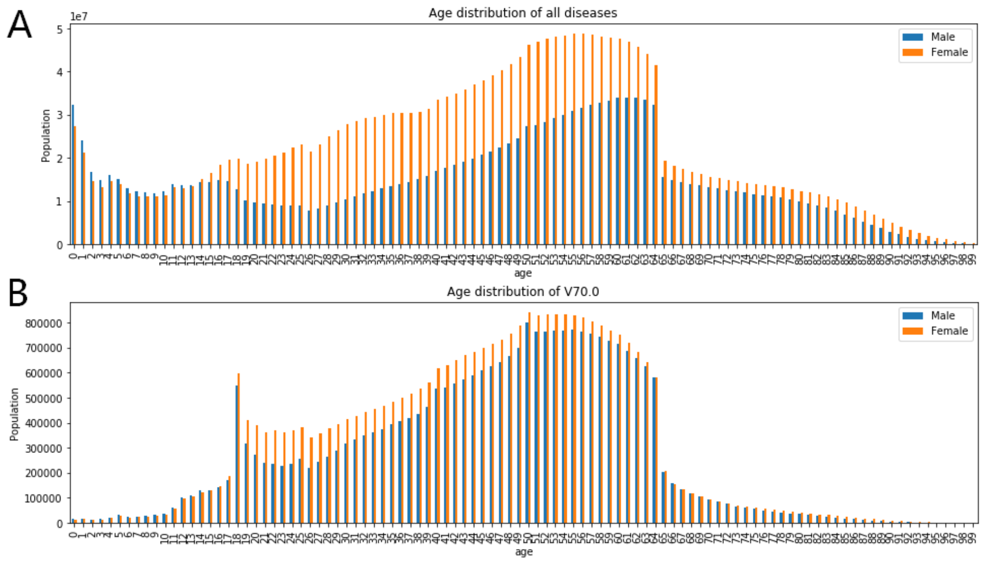


Figure 2.4: Age and gender distribution in the US in 2003-2018. A) The summary of patients for all ICD codes. B) Age/gender distribution for ICD9-CM code V70.0, which represents routine general health examination.

Age and gender are necessary information for diagnosis process because many diseases differ in distribution with respect to age and gender. For example, breast cancer in female patients is far more prevalent than in males. The same is true for age distribution. Figure 2.5 shows two common diseases that illustrate this type of difference: hypertension disease and influenza. We can see that hypertension more commonly effects the older population, while influenza more commonly effects the younger. For these same diseases, gender distribution is also different: Females are more likely to suffer from influenza than males, as opposed to hypertension.

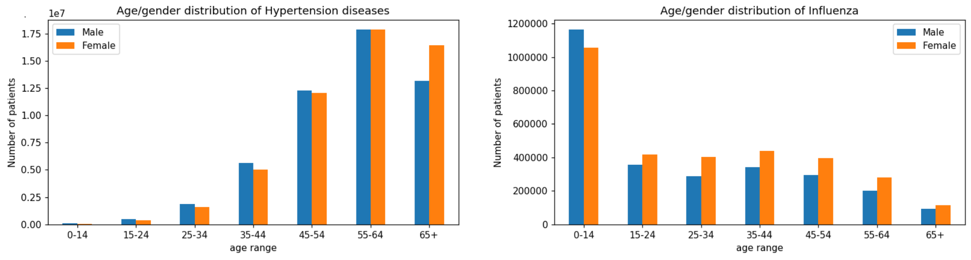


Figure 2.5: Age and gender distribution of hypertension/influenza in the US in 2003-2018.

Age and gender are not the only important factors in disease distribution – region also plays an important role. MarketScan is a dataset in the US, which might not fit to other countries like Nigeria. Based on data from the WHO, Figure 2.6 and Figure 2.7 show two examples of the different population distributions for breast cancer and tuberculosis in both the US and Nigeria. For breast cancer, we only show female patients because the instances of breast cancer in male patients are too few. Both countries have similar age distributions, but the US population is twice more likely to have breast cancer. However, for tuberculosis, the distribution between the two countries are quite different. US patients tends to be older than Nigerian patients, and in Nigeria, the total number is 20 times higher than in the US. Gender differences are also more significant in Nigeria.

Given the huge difference between developed countries (like US) and lower-resource areas (like Nigeria), we believe there is an urgent need for a specially-designed diagnosis system for lower-resource areas. We built, deployed, and tested an Android application intended to fulfill the need for decision-support systems to enhance physicians' diagnostic processes in lower-resource countries.

Figure 2.8 shows the application's key components, accompanied by screenshots showing

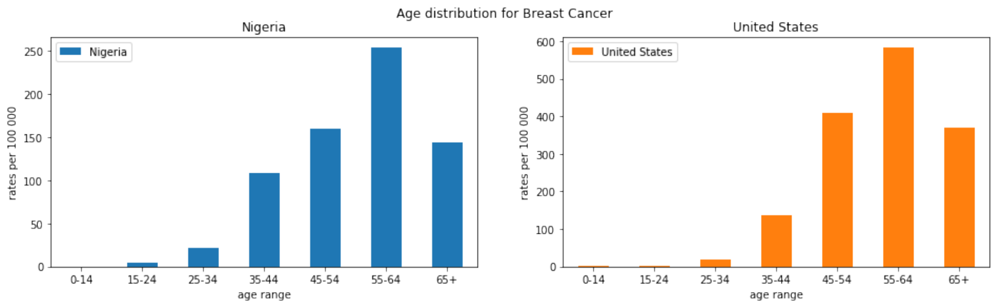


Figure 2.6: Age distribution of breast cancer in the US in 2003-2018, and in Nigeria in 2012 [47]

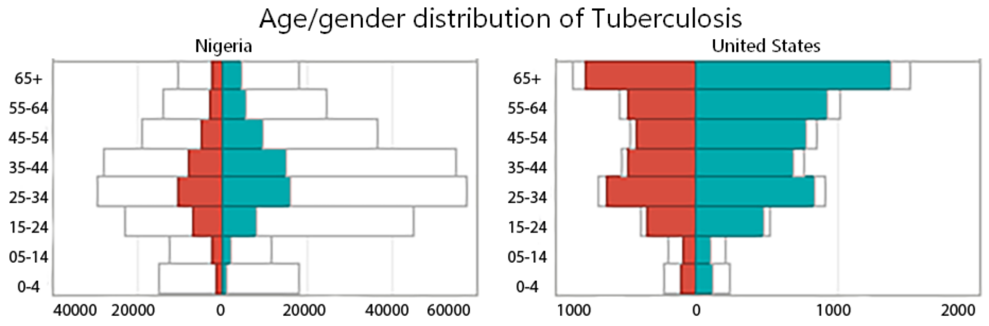


Figure 2.7: Age and gender distribution of tuberculosis in the US and Nigeria in 2018 (from the World Health Organization (WHO)). the y axis is the age range, and the x axis is the number of patients per 10 000 population. Blue stands for male, and red for female. Colored parts stand for treatment coverage.

the application's workflow: basic information collection, symptom input, adaptive question generation, and diagnosis suggestion pages. There are also a few recordings, confirmation, and orientation pages that guide the primary care physicians through the diagnosis process. The application has a few key functionalities: (1) Primary care physicians are able to visit and query the local database to retrieve existing patient information, eliminating the need to input basic patient information each time; (2) The application can work as an EMR system, allowing physicians to simply input and save all the information gathered both on the Android device and in the Cloud (if internet is accessible); (3) The application generates questions regarding symptoms based on the information collected during the previous steps, after which it further generates an adaptive diagnosis suggestion (this is the application's primary function), and; 4) The application automatically detects areas where internet connectivity is either unstable or unavailable by enabling synchronized data via encrypted SMS messages – an important consideration in lower-resource areas.

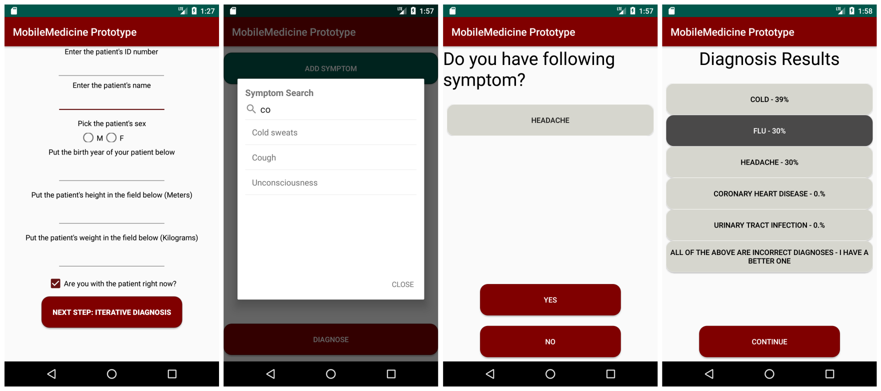

MobileMedicine Prototype
Enter the patient's ID number
Enter the patient's name
Pick the patient's sex
M F
Put the birth year of your patient below
Put the patient's height in the field below (Meters)
Put the patient's weight in the field below (Kilograms)
Are you with the patient right now?
NEXT STEP: ITERATIVE DIAGNOSIS
MobileMedicine Prototype
ADD SYMPTOM
Symptom Search
co
Cold sweats
Cough
Unconsciousness
CLOSE
DIAGNOSE
MobileMedicine Prototype
Do you have following symptom?
HEADACHE
YES
NO
MobileMedicine Prototype
Diagnosis Results
COLD - 39%
FLU - 30%
HEADACHE - 30%
CORONARY HEART DISEASE - 0.%
URINARY TRACT INFECTION - 0.%
ALL OF THE ABOVE ARE INCORRECT DIAGNOSES - I HAVE A BETTER ONE
CONTINUE

Figure 2.8: Example screenshots from the Android application's prototype. 1) The basic information collection page, including the patient's age, gender, weight, and height. These are the most basic properties that are likely to be associated with disease. Other important information, such as family history or travel history, will be included in future versions. 2) The symptom input page, in which the primary care provider can select from the dropdown menu or manually enter patient symptoms. 3) The adaptive diagnosis page, where the application generates related questions about symptoms/tests and then generates new questions based on the answers. 4) The result page, which provides suggestions about diagnoses, tests, and prescriptions.

The application's current iteration relies on a "one-time pad" encryption scheme to protect information transferred over SMS. "One-time pad" is a symmetric encryption algorithm that uses a randomly generated key, known only by both the sender and receiver, to protect information security.

In terms of hardware, we used a Dell Optiplex 780 running Ubuntu 16.04 LTS as the server. Our software is fairly hardware-agnostic, but we recommend that any implementations based on ours use a 64-bit processor, with at least 250GB of storage memory, and at least 8GB of RAM. To test SMS compatibility, we used a Huawei E8372 device, along with an open-source python library to communicate with the device `https://github.com/pablo/huawei-modem-python-api-client`. For application development and testing, we used unlocked Samsung S5 smartphone models.

We beta tested our initial Android application in Pakistan in early 2020 at Quaid-e-Azam International Hospital and Bashir Begum Memorial Welfare Hospital. We collected 113 diagnosis records for 99 individual patients, with 46 males and 53 females. Most patients were aged from 15 to 60, except for two children and two seniors. During our testing, physicians collected information from patients using the adaptive diagnosis process, in which the application sorted the disease list consisting of the 165 most common diseases and suggested five diagnosis results with the highest probability, given the information collected. Regardless of the suggestions from the application, the physicians also made their own diagnoses. We considered the application's prediction to be accurate when the doctor's diagnosis lay within the application's top five suggestions. Figure 2.9 shows the application's diagnostic accuracy. The application suggested five diagnoses each time, and for 77 out of 113 cases, the doctors agreed with the suggestions from the application, and for 27 cases, they agreed with the first suggestion. The top-five accuracy came to 68 percent and the top-one accuracy came to 24 percent. Note that this model's suggestions were built out of the medical literature without any real patient data, so there is much room for improvement.

While these application testing results are preliminary, they prove that our current prototype is functional and has the potential to serve a larger group of primary medical care providers. We will update the algorithms as the application is being used by a wider pool of doctors in lower- resource areas, based on the diagnosis data collected during usage.

2.2 Light-weight Diagnosis Assistant – Phase 2

To verify the algorithm updating policy, we collected and processed about 1000 clinical records from Nigeria. After an optical character recognition (OCR) processing provided by Microsoft Azure, we generated a text corpus with a size of 200kb. While this corpus was much smaller than those previously mentioned, it came from one of the lower-resource areas. We assumed that this corpus is more closely related to our Pakistan experiments

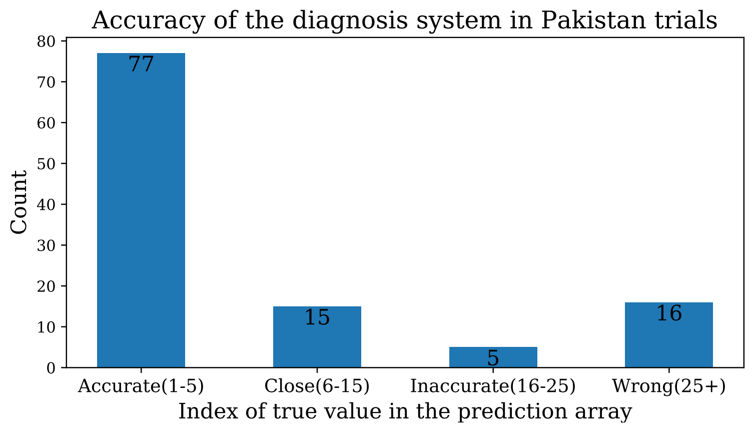


Figure 2.9: Results from the application's beta testing in Pakistani hospitals. In this chart, the y axis is the total case count, while the x axis is the application's accuracy index based on physicians' diagnosis. The accuracy index is defined as following: The application covers the 165 most common diseases in the US and in African countries. When used for diagnostics, we calculated the probability for all 165 diseases based on information collected from patients. The application sorted the disease lists according to the probabilities and suggested the top five diseases with the highest probability to physicians, allowing them to choose from these five suggestions of diagnoses or reject them to make other diagnosis.

than the corpus from the US, so we used this data to inform the updating policy to update the application and once again calculated the accuracy on the beta testing data from the Pakistani hospitals.

We used the new corpus to generate a new association matrix with the same methods described above, but this new association matrix only covered ten percent of diseases and symptoms. We added this new matrix to the original matrix after multiplying it by 0.1. We used the new association matrix as the updated model.

Table 2.2: Performance comparison between the original and updated algorithms, from the Pakistani testing trial

Model	Accurate	Close	Inaccurate	Wrong	AUCAC
Original	77	15	5	16	0.94028
Updated	77	16	4	16	0.94035

We observed a small improvement in our applications predictive power after the update, as shown in Table. 2.2. The result was slightly better than the original results shown in

Figure. 2.9, by having one more 'close' prediction. To better describe the improvement, we introduced a new metric called "normalized area under the cumulative accuracy curve" (AUCAC). First, we calculated a probability prediction for each disease, and we called k the diagnosis index if the k^{th} prediction met with the doctor's diagnosis. Optimally, all the diagnosis indexes should be 1. Next, we defined the top-N cumulative accuracy as the frequency of the diagnosis index k located between 1 to N, and we plotted the cumulative accuracy curve as a function of N, as shown in Figure. 2.2. The AUCAC is the normalized area under the curve, which is the area divided by the number of diseases, so that the optimal score is 1. We observed only a tiny improvement (0.00007) in terms of AUCAC, and this is because the new text corpus was almost 10,000 times smaller than the original corpus. The black-dashed line in Figure. 2.2 indicates the performance of a random guess, corresponding to a AUCAC of 0.5.

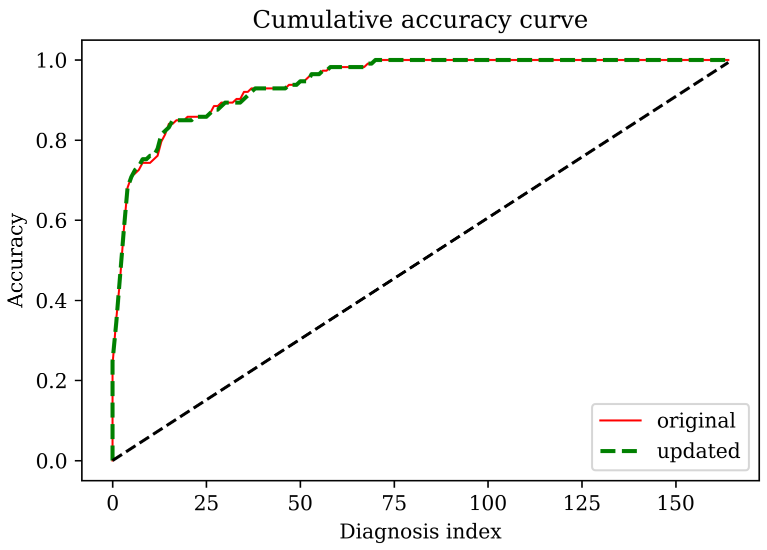


Figure 2.10: The cumulative accuracy curve. The top-N cumulative accuracy as a function of N from 0 to the number of diseases 165. The updated curve improved slightly, compared to the original one.

2.3 Side Project - Breast Cancer Prediction Application

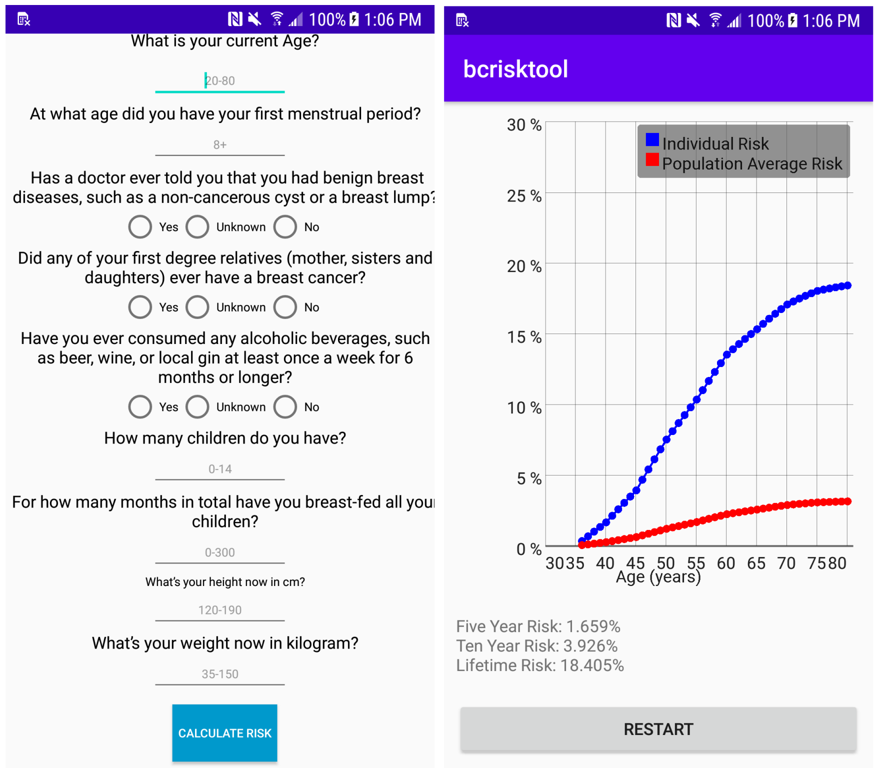
Breast cancer is the most common cancer in women. While risk prediction models for breast cancers are widely used, there is no breast cancer screening program generally offered in Sub-Saharan Africa (SSA). The Huo group at the University of Chicago has developed a model to predict the absolute breast cancer risk, especially for Nigerian woman. This work was published on Cancer Epidemiology and Prevention Biomarkers [109].

The Huo group utilized a dataset that covers thousands of breast cancer cases and control cases. Based on nine factors such as age, parity, number of children, breast feeding, and alcohol consumption, they developed an absolute breast cancer risk prediction model with multi-variable logistic regression. The model, evaluated with the area under the ROC curve, was [AUC]=0.703, outperformed any existing models for the Black population. More importantly, this model is the first of its kind in Africa.

The Huo group's model is accurate and has the potential to benefit all Nigerian women. However, they lack a deployment tool. They developed a web application for their model, but as mentioned in previous sections, internet accessibility is a challenge in Nigeria. It turns out that an Android application would work the best in this situation. Thus, we developed an Android application for the Huo group's breast cancer prediction model based on our diagnosis assistant application, outlined in Section 2.1, and shown in Fig. 2.11. This application ran much faster than the original web application and worked perfectly in Nigeria.

2.4 Discussion and conclusions

Lower-resource countries are suffering from severe disease burdens and lack of well-trained physicians. Current diagnosis systems are either not accurate enough or require too many resources, rendering them unusable in these markets. We are collaborating with the African


What is your current Age?

20-80

At what age did you have your first menstrual period?

8+

Has a doctor ever told you that you had benign breast diseases, such as a non-cancerous cyst or a breast lump?

Yes Unknown No

Did any of your first degree relatives (mother, sisters and daughters) ever have a breast cancer?

Yes Unknown No

Have you ever consumed any alcoholic beverages, such as beer, wine, or local gin at least once a week for 6 months or longer?

Yes Unknown No

How many children do you have?

0-14

For how many months in total have you breast-fed all your children?

0-300

What's your height now in cm?

120-190

What's your weight now in kilogram?

35-150

Five Year Risk: 1.659%
Ten Year Risk: 3.926%
Lifetime Risk: 18.405%

Figure 2.11: Example screenshots from the breast cancer prediction model Android application. Left: The input page for information collection, covering nine factors that influence risk. Right: The output page, showing the predicted risk as a function of age.

Institute of Mathematical Science and the University of Ibadan to design and deploy this electronic adaptive diagnosis system in order to help physicians in Nigeria, as well as other parts of the world, such as Rwanda and Pakistan. This system is a lightweight, adaptive diagnosis assistant system designed to help physicians arrive at diagnoses faster and with less errors and to assist with efficient patient record keeping. The adaptive diagnosis assistant system is based on online learning algorithms and will improve by itself as more patient data accumulates.

We developed this Android application prototype for use in clinics, and beta tested it with five doctors in Pakistan. The performance and feedback were mostly positive, with a 68 percent diagnostic accuracy, and the doctors reported that the application was very user-friendly, and they liked the idea that it mimics a physicians' decision-making process.

As to be expected, the doctor-to-patient ratio is extremely small in developing countries – doctors in both rural and urban areas of Pakistan have huge patient loads with which they struggle. It is therefore important to provide a tool that can help improve their diagnostic efficiency and hasten their workflow, enabling them to potentially attend to more patients per time unit.

Our diagnostic application is specifically designed for lower-resource settings. While the application targets lower-resource settings, such as rural Nigeria, the US is not devoid of lower-resource areas. In the US, annually, over 12 million adults who seek outpatient medical care receive a misdiagnosis [100], and there nearly 30 million uninsured individuals (ten percent of the population). Our application may be able to help these populations as well. In lower-resource settings, particularly in countries ranking low on the Human Development Index, obtaining accurate and timely medical data is extraordinarily difficult. Without robust local datasets, the function and evaluation of our diagnosis system is limited in its present form. Although we have utilized large datasets from the US and experimented in Pakistan, it is unlikely that these data can properly represent rural African areas. We are in the process of deploying the application to countries like Nigeria to conduct a first trial experiment to collect more localized data, while recognizing that there is vast heterogeneity across the African continent - and also within countries, especially between urban and rural contexts.

In the meantime, based on the feedback of doctors who tested the assistant system, it would be ideal if medical data could include more fields such as the duration and evolution of symptoms or patient medical histories. These are important factors that would help doctors arrive at better diagnoses and treatment plans. However, in our attempt to develop this ideal system, we are restricted by both insufficient data sources and technological limitations. The application requires a large amount of related data in order to add the requested features or medical input fields to the system, as well as different algorithms to handle the time-series information. We are planning to upgrade the system in this direction after the next trial in Nigeria.

Network accessibility may also be a problem when deploying the diagnosis system. To the best of our knowledge, cellular networks in many African rural areas are slow and brittle, and some rural areas have no cellular coverage at all. However, we did our best to build the lightest and most reliable diagnostic system possible.

Consequently, the current version of the system only requires users to send SMS messages of dozens of characters for each visit. Theoretically, this matches what we have observed in the field thus far. However, we have not tested the system while connected to an extremely unstable network; it is possible that data synchronization may fail. As we continue to develop our AI assistant system, adding additional facets of patient health descriptors, the required messages will grow larger, creating logistic difficulties in areas with low network capacity. We are reaching out to local Nigerian universities and companies in search of testing support for these issues.

CHAPTER 3
CHEMICAL APPLICATIONS

3.1 AI for a Chemical Naming System

3.1.1 Introduction

The project in this chapter involved an attempt to change the linguistic structure that has been used historically to represent organic chemical compounds. We theorized that these linguistic taxonomies define – and therefore necessarily limit – the way researchers conceive of, develop, and execute their research.

Have modern technological developments created a context that might enable us to improve upon these conceptually based research limitations? Is it possible that AI methods could do a better job than human scientists when identifying useful chemical sub-structures?

We undertook to answer these questions in collaboration with the University of Cambridge's King Laboratory, in their Department of Chemical Engineering and Biotechnology.

The commonly-used naming system for organic chemical compounds was developed in 19th century Germany [14], and was based on the compounds' sub-structures. Given that the names themselves describe related chemical properties, the language itself strongly influences how researchers think about chemistry.

We are investing the possibility to improve on the standard way of representing chemical structures, which would affect the way how people think about them. Specifically, we are trying to determine whether AI methods can do a better job in identifying useful chemical sub-structures than have human scientists.

We represented chemical compounds as labelled graphs through their atoms (graph nodes) and bonds (graph edges), and employed relational learning and quantitative structure activity relationship learning to extract frequent patterns of sub-structures for compounds. The performance of the extracted patterns had outperformed the standard structures when comparing the performance in ML. However, the performance was optimized during the learning process, so we needed a more objective evaluation. To achieve this evaluation, we found a way to determine the most commonly-used chemical sub-structures with BLAST (basic local alignment search tool), which is a biological tool used to compare DNA or RNA sequences. BLAST's high rate of efficiency allowed us to compare word and character sequences in chemical literature to find the most commonly-used chemical sub-structures.

While the performance of this new chemical compound representation is still under evaluation, BLAST has proven effective in helping us extract the common sub-structures from the literature. The following sections will describe the datasets and methods we used in this project and show the results.

3.1.2 Data

We used two datasets used in this project: PubChem, the world's largest collection of freely-accessible chemical information and terabytes of Elsevier literature which have been collected by the Argonne National Laboratory.

We used the PubChem data to extract the full list of chemical sub-structures, and we searched for these sub-structures in the literature to find those most commonly-used. Due to the vague nature of natural text, in which people can use different formats of the same words (such as plural or connecting symbols), a fuzzy match is required to find these sub-structures' occurrences.

3.1.3 Method – BLAST

We used BLAST to perform a fuzzy match for both the sub-structures and the literature text. As a biotechnology tool, BLAST is usually used for DNA/RNA or protein sequence comparison. A typical use case is to identify whether a specific gene is contained in a DNA sequence. We found this use case to be very similar to fuzzy match for sub-structures in literature, so we transformed the sub-structures and literature texts to DNA sequences and applied BLAST for the search.

The first step of the experiment was to locate all possible sub-structures. We fully traversed the IUPAC names in PubChem database and enumerated the sub-structures. IUPAC is the most widely used chemical nomenclature developed by the International Union of Pure and Applied Chemistry. This method allowed us to identify all the sub-structures used in the standard naming system.

The second step was to encode the sub-structures and the literature texts into the mRNA format with nucleotide symbols 'TACG'. Because we were only concerned with the names of the sub-structures, we encoded all the non-alphabetic characters with the single symbol 'T' to save space. We used the other three symbols ('ACG') to encode the 26 English letters, and transformed all upper-case letters into lower-case. Using this encoding, a perfect match will always get the highest score, and fuzzy matches with punctuation will get a higher score than matches with a difference in letters.

The third step was to search for the matches between the encoded sub-structures and literature and then identify those most commonly-used. BLAST uses statistical theory to produce a bit score and expect value (E-value) for each alignment pair. The bit score gives an indication of the alignment's quality, and the E-value indicates the statistical significance of the pairwise alignments. A lower E-value reflects a more significant alignment result. We took each sub-structure as a query and each sentence in the literature text as a sequence, and used BLAST to check the alignment. We regarded the high-score alignment counts as the occurrence frequency of the sub-structures.

We calculated the BLAST scores by counting the matched and mismatched symbols in the query and sequence segments, and we evaluated the significance based on the Gumbel extreme value distribution. The probability (p) of observing a score (S) equal to or greater than x is given by the following equations:

$$P(S \geq x) = 1 - exp(-e^{-\lambda(x-\mu)})$$

$$\mu = \frac{log(Km'n')}{\lambda}$$

in which λ and K are fitted parameters based on the distribution of alignment scores between the query and shuffled sequence segments, and m' and n' are the effective lengths of the query and encoded text sequences. The E-value is defined as the expectation of the match assuming a fully random process:

$$E \approx 1 - exp(-p(S \geq x)D) \approx pD$$

3.1.4 Results and Discussion

From the PubChem database, we obtained 240k sub-structures in total, after dropping words shorter than three letters (mostly the element symbols). EWe encoded each of these sub-structures into a query and searched through the Elsevier literature corpus to find matches with BLAST. For each query (sub-structure), BLAST generated a list of matches with an E-value. We considered only the top two E-values as high-score matches because they are likely to be exact matches and fuzzy matches with connecting symbols, and other E-values correspond to some mismatch in the letters.

A sub-structure's occurrence frequency is counted as the number of sentences in which we found a high-score match. If a sub-structure appears in the same sentence multiple times, we counted only one. Figure 3.1 shows the results of the sub-structure frequencies in a word cloud. The larger the word, the more frequent it has been used in literature.

As shown in Figure 3.1, while some of the sub-structure words were the obvious ones, such as "acid" and "amino," and many others provide the necessary information for our proposed

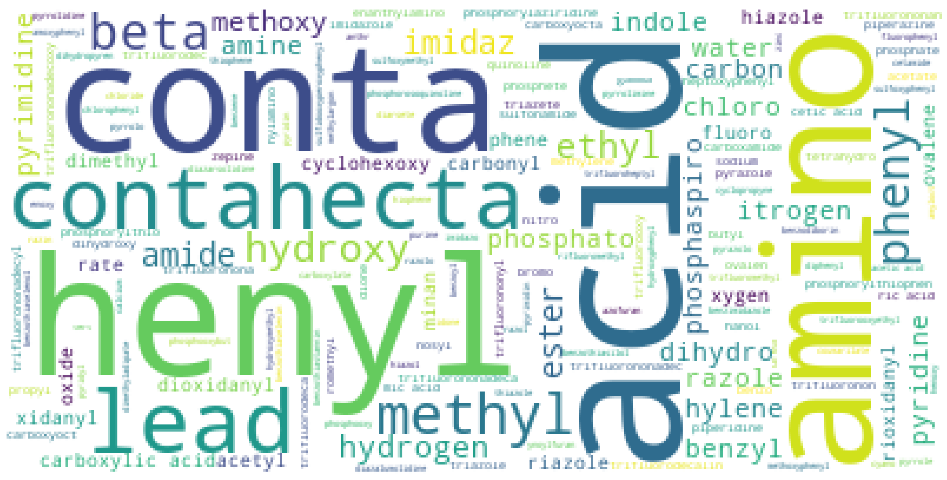


Figure 3.1: Word cloud for chemical sub-structures. The words are extracted from PubChem database, and the frequency is counted based on the Elsevier literature corpus.

naming system. However, there are also some sub-structure words that are biased due to the nature of the datasets. For example, 'lead' has different meanings in chemistry as compared to general literature, and a large portion of the counts come from the general meaning of "guiding". Besides the defects, BLAST has done a great job here finding the patterns of most commonly-used chemical sub-structures and significantly helped us to develop our new chemical naming system.

CHAPTER 4
MANUFACTURING APPLICATIONS

With the proliferation of Internet of Things (IoT) devices, the distributed control systems are now capturing and processing more sensors operating at higher frequency than ever before. These new data, due to their volume and novelty, cannot be effectively consumed without the help of data-driven techniques. Deep learning is emerging as a promising technique to analyze these data, particularly in soft sensor modeling (or soft sensing). From an architectural perspective, the strong representational capabilities on complex data and the flexibility it offers from an architectural perspective make it an active topic of applied research in industrial settings.

However, despite its apparent promise, successful applications of deep learning in soft sensor modeling have not yet been widely integrated in factory control systems – largely due to data-modeling limitations; soft sensing does not yet have access to large-scale industrial data, which are typically varied, noisy, and incomplete. The results published in most research papers are therefore not easily reproduced when applied to the variety of data in industrial settings. In this chapter, we provide manufacturing datasets that are much larger and more complex than any public open soft sensor data. Moreover, the datasets we have used are from Seagate Technology factories on active service with only the necessary anonymization, which means they reflect the complex and noisy nature of real-world data. We introduced a variance-weighted multi-headed quality-driven autoencoder classification model that fit well into the high-dimensional and highly imbalanced data. In addition to the use of weighting or sampling methods to handle the highly imbalanced data, the model also simultaneously predicts multiple outputs by exploiting output-supervised representation learning and multi-task weighting.

There have been many studies exploring deep learning models in soft sensor area. However, the models have become more and more complex – yet, their datasets remain limited;

researchers have been fitting million-parameter models with hundreds of data samples, which is insufficient to exercise the effectiveness of their models. Our study has attempted to solve this persistent problem, by providing large-scale high-dimensional time series manufacturing sensor data from Seagate Technology to the public.

Our project aims to demonstrate the challenges and efficacy of modeling industrial big data by using a soft sensing transformer model on these datasets. We chose to use Transformer because it has outperformed state-of-the-art techniques in natural language processing, and since then has also performed well in the direct application to computer vision without requiring the introduction of image-specific inductive biases [107, 20]. We first observed the similarity of a sentence structure to the sensor readings and processed the multi-variable sensor readings in a similar manner to that of sentences in natural language. We then formatted the high-dimensional time series data into the same shape of embedded sentences and fed them into the transformer model. The results showed that the transformer model outperformed the benchmark models in the soft sensing field based on autoencoder and long short-term memory (LSTM) metrics. To the best of our knowledge, we are the first team in academia or private industry to benchmark the performance of an original transformer model with large-scale numerical soft sensing data. Additionally, in contrast to the natural language processing or computer vision tasks in which human-level performances have been regarded as golden standards, our large-scale soft sensing study is an example that the transformer is able to interpret high-dimensional numerical data which humans have been unable to interpret.

4.1 Autoencoder-based Model for High-dimensional Imbalanced Industrial Data

4.1.1 Introduction

(This work has been published on International Conference on Neural Information Processing [123]©Springer.)

Data-driven soft sensing models in the past few years have generated active research and investment due to their efficient handling of complex data relationships and reduced reliance on first principles. They offer reliable and economical alternatives to rule-based, first principles-driven expensive measuring sensors, thereby enabling effective feedback, and control strategies for process monitoring [82].

A variety of deep learning techniques have been proposed over the last few years in soft sensor application, including stacked autoencoders[23, 118, 119, 37, 92], gated units[102], and flavors of recurrent [120] and convolutional neural nets [42]. This interest in deep learning techniques has been motivated by the promising empirical results of these deep learning techniques, which show improved representation ability, the ability to exploit unlabeled data to improve performance, and the potential for non-linear feature extraction.

While the body of research around soft sensing has been non-trivial and has contributed substantially to the interest and increased research in data-driven soft sensing, we still see a gap between laboratory outcomes and industrial deployments of these soft sensors [48].

One of the reasons for this gap is the lack of real-world data upon which these models can sufficiently be exercised. For example, most of the soft sensors-related work cited in this article use datasets such as the benchmark TE (Tennessee Eastman) and the Debutanizer column [28]. These have relatively low volumes and dimensions (with a number of samples of up to $30k$ and dimensions up to 52). On the other hand, a real, distributed control system in an industrial plant monitors thousands of operations, generating tens of thousands of sensors from hundreds of pieces of equipment capturing millions of observations per day. Additionally, there is an inherent data imbalance problem that none of these studies capture or address. From a practical implementation perspective, we believe that there is also a need for an architecture that can not only learn the interactions between all these sensors within a tool, but also be able to simultaneously predict operations down the manufacturing line, in order to establish proactive process monitoring and controls.

In this section, we describe our attempt at building a scalable soft sensor application that is motivated by the above-mentioned current gap in the variety of process data for research and the speed of deployment and generalizability for broad coverage in industrial process control. We proposed a variance-weighted multi-headed classification model that simultaneously predicts multiple outputs by exploiting output-supervised representation learning and multi-task weighting.

Our contributions are:

1. A generalized imbalance-weighted algorithm that uses sensor data captured from the process tools to predict multiple outcomes measured by multiple measurement tools

2. The open release of the preprocessed real-world data to advance further research in soft sensing using a diverse family of semiconductor process manufacturing tools.

In this section's following subsections of this section, we describe the proposed variance-weighted multi-headed classification model and present a case study on wafer manufacturing data from Seagate Technology, in order to demonstrate the performance of our proposed model.

4.1.2 Related Studies

Over the last decades, most deep learning-based soft sensor modeling has used variants of autoencoders as a pre-training approach. This was motivated by the work done by Erhan [23], which demonstrated – with extensive experiments – that unsupervised pre-training provides solid marginal distribution and captures more intricate dependencies between parameters. In other words, unsupervised, pre-training functions similar to a pre-conditioner, providing a more suitable runway for future supervised training.

However, the challenge has been that, even though deep learning techniques extract high-level features from the data, these features may not always be relevant to discriminating the specific problem at hand; for example, detecting the quality of a product or a fault in a piece of equipment. To address this, several researchers have tried to inject relevant output variables during pre-training to guide the training towards learning features that are more discriminating of the classes and outputs of interest.

There have been several studies that have used artificial neural networks for feature extraction and representation learning. Below, we discuss the related studies that have addressed previous industrial process monitoring that employed unsupervised pre-training approaches, or semi-supervised or supervised techniques.

An automated key feature learning method was presented by Zhang [128] for nonlinear process monitoring, which applies the k-nearest neighbor rule to find the nearest samples in the training set. The autoencoder is utilized to convert the input space into a feature and residual space. Control limits are derived by kernel density estimation, so pertinent fault detection alarms can be triggered when the appropriate thresholds are crossed.

Han et al extracted more discriminative features by proposing an autoencoder-inspired unsupervised feature selection method that can jointly learn to reconstruct its inputs and the importance weights of each feature by simultaneously minimizing reconstruction error and group sparsity regularization. [37]

Sagheer et al leveraged the unsupervised stacked autoencoder to replace the random weight initialization strategy adopted in deep LSTM recurrent networks [92]. Their proposed LSTM-SAE used unsupervised, pre-trained LSTMs in an autoencoder fashion. Their results show that an unsupervised, AE-based, pre-training approach improves the performance of deep LSTM using time series data and leads to faster convergence than other models.

Autoencoders are also implemented by Yao [115] for feature extraction of process variables. Their solution implements a semi-supervised extreme learning machine based on manifold regularization in the last hidden layer of the deep autoencoder to obtain better prediction of process variables. In addition to exploiting a large number of unlabeled process data in the learning process, these extreme learning machines have the advantage of being computationally more efficient because they do not use resource-intensive back propagation.

However, as applied research for process monitoring in the industrial space continues, it is becoming clear that while unsupervised pre-training techniques do a good job of extracting high-level features from the data, these features do not always represent the specific metrics of interest such as quality of a product or fault in an equipment unit, etc. This failure inhibits their widespread implementation in the actual process control. To address this gap of making representations learn statistical structures of process interest, there have been multiple efforts at making representation more relevant to outputs of interest by injecting relevancy into the pre-training process.

Yuan et al introduced variable wise feature selection, in which process variables are weighted by their correlations with the output variable so as to extract high-level output-related features [118]. This technique uses the Pearson correlation, which is a linear correlation measure, and the features it extracts may not fully represent the non-linear relationships. Yuan et al also proposed a high-level feature learning by introducing an output-relevant loss function into the pre-training process so that the network can simultaneously learn to reconstruct input features along with the quality variable [119]. They applied this to obtain higher performance in regression tasks. Information considered to be irrelevant was eliminated by a relevant representation learning method [114] calculating the mutual information in each layer between the representation and output layer to reveal high-order relationships. Their results show that irrelevant representations are eliminated during the training of a stacked autoencoder's subsequent layer.

In 2020, Sun et al exploited the idea of gated units [102] to modulate the flow of information in a stacked autoencoder where each hidden layer‘s contribution is considered to predict the target variable. They hypothesize that there are important representations at even shallow layers and therefore, each must be able to contribute to the final prediction. They also use an ensemble strategy to make more effective use of unlabeled samples.

Luo et al also attempted to make the latent structure more relevant to the outcome of interest by adding a distance penalty into the loss function [69], which enabled the autoencoder to extract more suitable representations by increasing the labeled data's inter-class separability. This model provided a mutual, information-based mathematical interpretation to support their proposed technique's effectiveness.

There have also been studies leveraging convolutions and sequence modeling with variants of CNN together with LSTMs. Huang et al explore multichannel CNNs to extract the correlations of different process variables regardless of their topological distance [42]. Yuan et al integrate the CNN and LSTM architectures to predict particulate matter concentration [120]. Their proposed architecture includes a CNN to extract features in both spatial and temporal domains. IT then uses LSTM to analyze the extracted features and perform forecasting.

As can be seen from these studies, autoencoder-based feature extraction and relevant feature learning using a combination deep learning techniques – such as stacked autoencoders, semi-supervised autoencoders, CNN, and LSTMs – have been widely studied with varying levels of success in industrial data.

4.1.3 Methodology

Autoencoder

An autoencoder is a neural network that attempts to learn the underlying latent variables from high-dimensional input. Raw input data x is encoded into a low dimensional latent feature space h with an encoder function. The decoder function then reconstruct the input as $\hat{x}$ using the latent variables. The autoencoder is trained by minimizing the distance between the original input x and its reconstructed counterpart $\hat{x}$. For example, suppose $x \in R^{d_x}$ where d_x is the dimension of the input data and $h \in R^{d_h}$ where d_h is the dimension of the encoded $\hat{x}$. The encoder, decoder, and the loss functions then can be expressed as follows:

$$h = f_{encoder}(W_{encoder} \cdot x + b_{encoder}) \tag{4.1}$$

$$\hat{x} = f_{decoder}(W_{decoder} \cdot h + b_{decoder}) \tag{4.2}$$

$$J(\theta_{AE}) = \frac{1}{N} \sum_{i}^{N} |\hat{x}_i - x_i|^2 \tag{4.3}$$

where h is the encoded feature and $f_{encoder}$ and $f_{decoder}$ are non-linear activation functions that are used in the encoding and decoding process. N is the total number of training samples. Autoencoders can be seen as a special case of feed-forward networks and may be trained with optimization techniques such as gradient descent, following gradients derived by the back-propagation algorithm. Therefore, the parameters $\theta_{AE} = \{W, b\}$ in Equation 4.3 can be updated using gradient descent as follows:

$$\theta* = argmin_{\theta} J(\theta_{AE}) \tag{4.4}$$

Autoencoders are ideal for data which are characterized by high volume, co-linearity and not-fully-understood non-linearity. By using a non-linear activation function such a sigmoid or a rectified linear unit (ReLU), the autoencoders can not only reduce the dimension space, but they can also learn a more powerful generalization compared to principle component analysis.

However, a shallow network may not be able to fully represent the complex statistical structures of the input data. A common strategy is to greedily pre-train the deep autoencoders by training a stack of shallow autoencoders, thereby allowing the deep network to learn more abstract features hierarchically.

The self-supervised representational capabilities of the autoencoders can be further exploited to make them more relevant towards specific tasks such as predicting a failure of specific processes. This has been done by Yuan et al. [119] with improved experimental results. They introduced a quality-based autoencoder (QAE) which attempts to reconstruct both the input data x and the quality data y simultaneously, which are denoted as $\hat{x}$ and $\hat{y}$. They added a prediction error for the target variable in the loss function of the traditional autoencoder in Equation 4.3. As shown in Equation 4.5, the loss function then jointly minimizes the reconstruction loss and the prediction error between the true and the predicted label.

$$J(\theta_{QAE}) = \frac{1}{N}\sum_{i}^{N}(|\hat{x}_i - x_i|^2 + |\hat{y}_i - y_i|^2) \tag{4.5}$$

The main intuition here is that a combined loss function that minimizes the gap between true and predicted labels when added to the autoencoder reconstruction loss function, can guide the autoencoders towards reconstructing more specialized or relevant feature space.

This idea of guiding the unsupervised learning with a target variable can then be further developed to multi-outcome tasks. Consider a learning task that maps the relationship between the sensors of one process machine in an industrial plant and the various outcomes as measured by multiple measurement machines down the line.

Instead of modeling each combination of process stage and measurement outcome step separately, we can model that entire process, with all its measurement outcomes, to jointly learn the outcome-relevant representation of each tool and the outcomes of a specific family of measurement tools. At each process of stage completion, a classifier then predicts class labels – not just for the next step – but all the subsequent steps at which the process would be measured, thereby enabling continuous monitoring and proactive control of downstream processes.

Variance-weighted Multi-headed Quality-driven Autoencoder

While existing autoencoder-based models have shown good performance on public datasets, their work does not consider the imbalance in the class labels and may be prohibitive in environments where there are a large amount of process and output quality variables. To address these practical limitations, we propose a variance-weighted multi-headed quality-driven autoencoder (VWMHQAE) classification architecture which learns and predicts multiple quality-relevant features simultaneously. This is a combined method that predicts all measurement steps within the same model, and is thus much more efficient and scalable than individual methods.

We simplified the soft sensing problem into a classification problem because our main objective was to predict whether a wafer passes measurements. This is commonly considered to be more easily interpreted when compared to a numerical output. At a mathematical level, a classification model is no different from regression except that the loss function is cross-entropy instead of mean-square-error.

In a Seagate Technology manufacturing line at any given processing step, there are a number of sensor values being captured which can be used to monitor and analyze multiple quality attributes throughout that process. Typically, these quality measurements are performed across multiple steps. To mimic this process, we designed a model with multiple heads, with each head denoting a measurement step. At each measurement step, specific

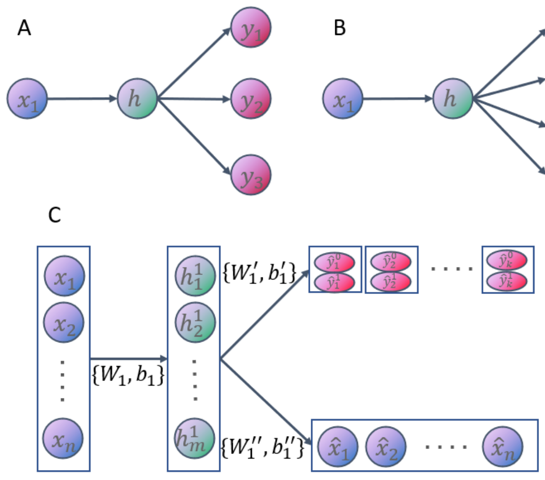


Figure 4.1: A) Illustration of the multi-headed model. B) Learning the reconstruction and target variable simultaneously. C) The multi-headed autoencoder model

properties of the wafer are measured in order to determine whether the properties pass the specification, which corresponds to a binary classification problem. As shown in Figure 4.1, several binary classification units are put together, to form a combined model to predict all the measurement steps simultaneously.

To calculate the loss functions for all the binary classifications together, we added a weight to each class for each head to handle the imbalance from the source data. The loss function for the entire classification problem is as Equation 4.6.

$$J_y(W,b) = \frac{1}{N}\sum_{j}^{N_h}\sum_{i}^{n_j}[-y_{ij}^0 log(\hat{y}_{ij}^0) * w_j^0 - y_{ij}^1 log(\hat{y}_{ij}^1) * w_j^1] \tag{4.6}$$

N is the number of data points, N_h is the number of heads, n_j is the number of data points for the j^{th} head, W, b are the model weights and bias, and w_j^0, w_j^1 are the class weights for negative and positive cases for the j^{th} head. The class weights are calculated as Equation 4.7, with t indicating the negative and positive labels.

$$w_i^t = \frac{N}{2N_h * n_j^t} \tag{4.7}$$

Note that the J_y in Equation 4.6 is not yet the final loss function for our model. To learn a representation of the input data and the features relating to the target variables at the same time, we used an autoencoder-based architecture. This required the objective function to be modified to include loss functions for both input reconstruction and target prediction. In our case, the reconstruction loss is an MSE loss for the input variables J_x, and the target prediction loss is J_y as Equation 4.6. To make it more intuitive, Figure 4.1B illustrates the workflow for the two loss functions, in which the target prediction loss is shown as multiple heads. Figure 4.1C shows the details on the multi-headed binary classification model.

However, to combine J_x and J_y, it is not acceptable to simply add them up, because they have different scales. Kendall et al. [55] proposed a method to minimize multi-task loss functions simultaneously. The authors showed that maximizing a multi-task likelihood is equivalent to minimizing a combination of individual task losses with variance-dependent weights, with an assumption of homoscedastic uncertainty for each task. They achieved better performance with the multi-task weightings than individual tasks trained separately. Applying their method to our model, we have the loss function as Equation 4.8, with σ_1 and σ_2 standing for trainable variance parameters for J_x and J_y:

$$J = \frac{1}{2\sigma_1^2} J_x + \frac{1}{\sigma_2^2} J_y + log\sigma_1 + log\sigma_2 \tag{4.8}$$

The Stacked Model

We used manufacturing data from Seagate Technology, specifically sensor data captured during the wafer processing steps within one family of deposition tools. This data has hundreds of dimensions. To render the model powerful enough to deal with the complicated datasets, we developed a stacked version of the variance-weighted multi-headed autoencoder-

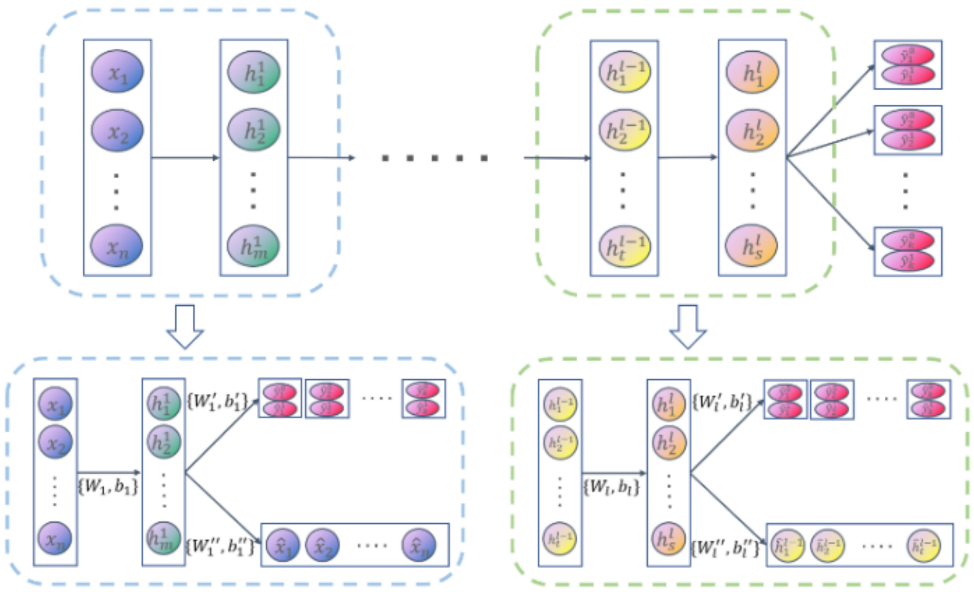


Figure 4.2: The stacked VWMHQAE classifier

based classifier, as shown in Figure 4.2.

As we discussed above, the autoencoder part of this model is encoding the input data as a lower-dimension representation and reconstructing both the input layer and the target variables. Here, the target variables include multiple heads, and each head is a binary classification. The loss function for this model contains two parts: one is the autoencoder loss function, and the other is the multi-binary-classification loss function, as in Equation 4.8.

The stacked model also added a multi-headed classifier to the original autoencoder, enabling it to learn a representation that captures those features most relevant to characterizing and predicting a series of key wafer quality-control measurements. Given that the heads' dimensions are much smaller than the features, this model adds negligible complexity to the autoencoder, and thus trains just as fast as traditional autoencoders. On the other hand, the calculation is accelerated by the multiple heads' vectorization.

During each stage of the wafer manufacturing process, the equipment's sensors acquire data points, each of which may contain multiple, relevant quality properties, which can

Table 4.1: VWMHQAE pseudo code

```
Training process
> Decide the structure of the model with number of layers $n$ and each layer $[l_1 - l_n]$, and input data $X_0$, output data $y$
> Loop: for $k$ in 1 to $n$:
    – Train a layer of the encoder-decoder with input $X_{k-1}$ and output $(X_{k-1}, y)$, and get the hidden layer $X_k$ with dimension $l_k$
¿ Based on the final layer of encoder-decoder, extract the hidden layer $X_n$, and train a logistic regression classifier with output $y$, get prediction $\hat{y}$
> Evaluate the results by comparing $y$ and the predicted $\hat{y}$:
For each output $y_i$ in $y$:
    – Count the correctly predicted positive/negative cases and calculate the recall and accuracy
```

then be measured by multiple types of measurement equipment. Our model addresses that reality by taking a series of these single-labeled data points to learn the behavior of multiple target variables. Our proposed model can be seen as jointly learning and solving two tasks: reconstructing the input data, and; predicting each head's target class.

Model Implementation

Given the high-dimensional input, we used a stacked encoder-decoder structure to reduce the number of dimensions gradually. While the input data dimension was 632, and the output (target) dimension was 16, we chose four layers of the stacked encoder-decoder structure, with the layers as [400, 200, 100, 50], so that in each iteration, the dimension was effectively reduced and little information was lost.

To further illustrate the effectiveness of the multi-heading mechanism, Figure 4.3 shows the comparison between the multi-headed model and the single-headed model, instead taking the heads as input categorical variables. In the original data, the feature indicating which measurement belong to what wafer was a column that had no difference with any other categorical variables. As far as we know, researchers have traditionally been taking these kind of variables as categorical variables in the input. However, we found that the measurement

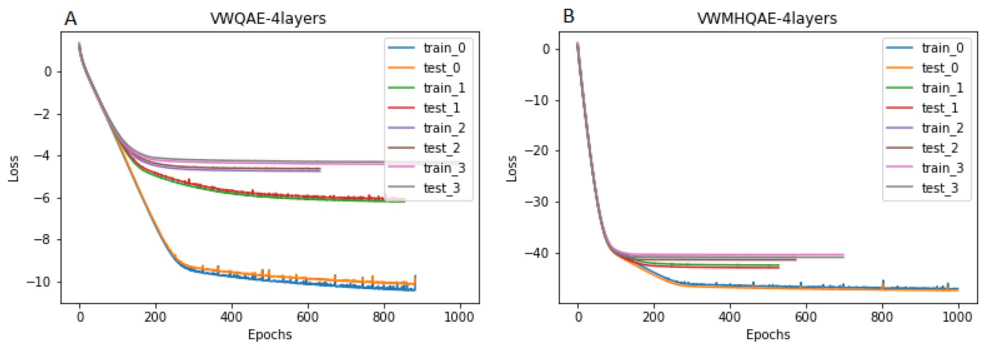


Figure 4.3: Comparison for the multi-headed model with a model taking the heads as a categorical feature. A) Variance-weighted quality-driven autoencoder model, treating measurement feature as an input categorical variable, B) Variance-weighted multi-headed quality-driven autoencoder model, treating measurement features as output heads.

feature can not only be a categorical variable in the input data, but can also be an output head – given that there are not too many distinct values (eight in this dataset). As shown in Figure 4.3, it turns out that taking the measurement variable as output heads not only improved the interpretability, but also helped the model converges much faster and with less fluctuations in the loss function as in Equation 4.12. Note that the loss values went to negative due to the formula's variance-weighting, and further note that the absolute values in Figure 4.3 may not linearly reflect the accuracy of the model. However, it's shown that the multi-headed mechanism accelerated the convergence and led to a lower loss function value.

In our experiments, we used a four-layer stacked model to learn on a dataset with 632 dimensions which has about 0.2 million data points. We used an Adam optimizer with the learning rate as 1e-3, the batch size as 512, and an early-stopping with a minimum delta as 1e-5. There were eight different measurement steps, leading to 16 heads in the model. The dimensions of the four hidden layers were [400, 200, 100, 50]. When we compared this with a regular autoencoder model with the same layers, our VWMHQAE model had only 1.7 percent more parameters. The complexity of our model is almost the same as a stacked autoencoder. With a training data of 0.16 million and 632 dimensions, the training for a four-layer VWMHQAE model took about 30 minutes on a NVIDIA Tesla V100 SXM2 GPU.

4.1.4 Data Description

The main purpose of this work was to find a generalized and computationally feasible soft sensing technique that could be applied to a diverse family of equipment. The model also needed to perform well in class imbalance and be able to simultaneously learn to predict the output of multiple measurement steps that are critical to ensure the quality of a particular point.

In complex manufacturing settings, effective process requires that we have measurements in place for our processes that can be monitored and controlled automatically. While these measurements are critical to assess processes quality, they also add non-trivial cycle time into the manufacturing process. We needed to maintain an optimal trade-off to ensure the right balance of quality and time-to-market. A soft sensing methodology that maps the abundant process variables to quality metrics, with a high degree of recall could potentially be exploited to reduce the time-to-ship by enabling such processes to be skipped based on the process variable readings.

For the purpose of this work, we used one year's worth of process variables (sensor) data, with the corresponding measurement (quality) variables. The sampling rates of the sensors were mostly per second. We collected the measurement data as a wafer was being physically measured, a process which could range from minutes to hours. The data is published at `https://github.com/Seagate/softsensing_data`.

The process tool generating the dataset is a vacuum tool that deposited thin film. Processes on this tool included both etching and deposition steps. Materials were sputtered from the target deposits on the substrate to create a film.

The tool family we considered in this study comprises 14 pieces of physical deposition equipment. Each piece of equipment has 130 sensors installed, which are capturing data at a frequency of about every second.

The critical parameters we measured for this family of tools were magnetics, thickness, composition, resistivity, density, and roughness. These parameters were critical to the wafers' quality and were measured by a family of high precision measurement equipment that came with high capital costs. A wafer can go through one or more measurement tools, depending on the criticality and complexity of the required measurement process. The measurement variable sample size is much lower than the process variables. A measurement taken by any physical measurement tool comprises a step in the process. Therefore, in this paper, we regarded each measurement step as an individual task, referred to as Y_1-Y_8. The measurements examine different wafer quality parameters, including thickness, magnetics, resistivity, composition, etc. The number of data samples varied from 2,000 to 25,000, and the imbalance ratios ranged from two to 225.

As to be expected from most real-world data, the sensor data we used here was highly imbalanced. In this study, we considered our data imbalance issue from two perspectives. First, there was the class imbalance, which results from the ratio of positive versus negative classes. Second, we referred to the difference in scale or unit between the mean squared and the cross-entropy losses used in the combined loss function as imbalance in the task. The class imbalance biased the learning towards the majority class due to its increased prior probability. The task imbalance, or the difference in the combined loss function units used in the proposed model's two tasks, could result in imbalanced weights for the different losses, which biased the learning towards the higher weighted loss and also possibly causes the lower weighted loss to suffer from vanishing gradients.

In order to determine the most reasonable fit for the proposed soft sensing architecture, we experimented with various techniques, and the results are discussed in the section below.

4.1.5 Results and Discussion

Despite recent advances and successes of deep learning, there has not been enough empirical work in problems where high class imbalance exists [49]. Methods for handling class imbalance in machine learning fall into three categories: data-level techniques, algorithm-level methods, and hybrid approaches [58].

Data level techniques address the issue of class imbalance through data sampling methods. Algorithm-level techniques are usually incorporated in the cost function, so the learner attempts to mitigate the bias towards the majority class during learning. Hybrid approaches combine the two.

Methods for handling the imbalance in the units or the weights of different losses in a combined loss function involve using a naïve weighted sum of losses [22]. Kendal et al use a Bayesian approach to learn multiple objectives in a loss function [55].

We tested both the data-level and algorithm-level techniques to address the class imbalance. First, we used the synthetic minority over-sampling technique (SMOTE), as it is a widely accepted benchmark for learning from imbalanced data [27]. The SMOTE method utilizes the oversampling approach to rebalance the original training set. Instead of applying a simple replication of the minority class instances, it introduces synthetic examples which are created by interpolation between several minority class instances within a defined neighborhood.

For soft sensor modeling, however, algorithm-level approaches were shown to be more effective. We adjusted the loss function to take weights into consideration in order to reduce bias towards the negative class, as shown in Equation 4.6. The introduction of the weights in the loss function allowed the minority samples to contribute more to the loss. Successful use of class weight has been demonstrated by Wang et al [110] and Lin et al [108].

For evaluation and comparison purposes, we considered true positive rate(recall) to be the most useful metric because, in this case, the cost of the false negative was much higher than the cost of false positive.

We tested the proposed architecture using several data imbalance techniques to determine what best suited the data. We tested both SMOTE and Weighted Class techniques on models which included logistic regression (LR), neural network (NN) with three fully connected layers, VWMHQAE+LR/NN (LR/NN using quality-based latent variables), and Stacked VWMHQAE+LR/NN. In this case, the results showed that in the algorithm-based approach, weighted class outperformed SMOTE, in most cases. Therefore, in our subsequent experiments, we used class weights to handle class imbalance.

Additionally, we used two different approaches in order to handle the different scales in the combined model loss. In the first approach, we used a naïve weighted sum of losses to combine the reconstruction loss and the binary cross entropy loss. The second approach saw the two loss functions as a multi-task learning problem and used a variance-based weighting mechanism, as proposed by Rijsbergen et al [105].

As Figure 4.4 shows, autoencoder-based models consistently performed better than models without pre-trained autoencoders. The highest-performing model across all outputs was a weighted class and variance-weighted, stacked, quality-driven autoencoder with a logistic regression classifier (recall=0.75), followed very closely by a naively weighted, quality-driven, autoencoder model (recall=0.74). *F_beta_imb* was the *F_beta* score with β equals to the imbalance ratio. *F_beta_imb* showed the same conclusion as recall.

4.1.6 Conclusion and Future work

In this study, we aimed to create an efficient, multi-headed autoencoder-based architecture that learns to simultaneously classify multiple outcomes by exploiting the feature representation guided by quality relevant features. We consider this to be an important contribution to the implementation of soft sensor modeling in manufacturing environments that involve complex, non-linear interactions among thousands of complex operations. The capability of

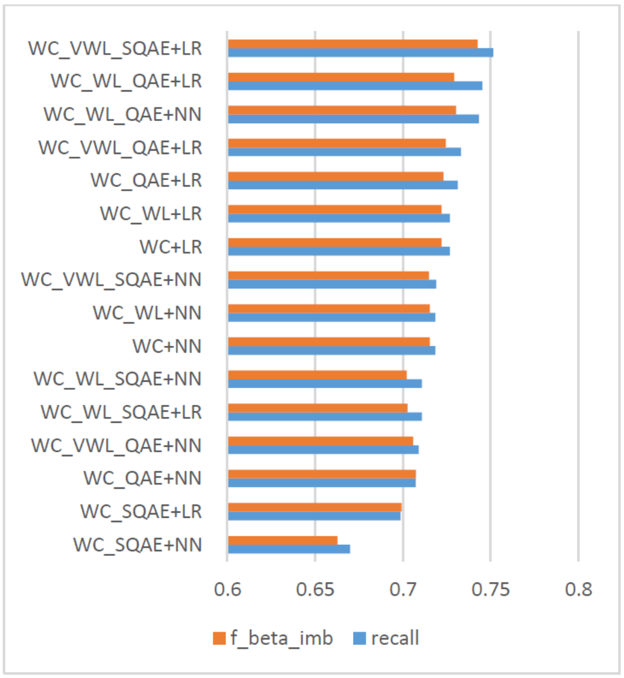


Figure 4.4: Average Output Performance for the VWMHQAE model's variations. The abbreviations are: AE: autoencoder; Q: short for 'VWMHQ'; S: stacked (with hidden layers [400,200,100,50]); WC: weighted class; WL: weighted loss; VWL: variance-weighted loss; LR: linear regression classifier; NN: fully-connected neural net classifier (with hidden layers [100,50]).

using sensors to characterize the impact of multiple steps' outcome in a feed-forward manner allows manufacturers to be more proactive about potential issues that may arise, and at the same time, enables them to eliminate certain measurement steps, thereby saving tool capacity and improving their time-to-ship window.

We used several techniques to address the challenge of class imbalance and the imbalance in the scale of the different loss function and concluded with an optimal model suited for this dataset. This architecture can also be trivially generalized to regression models or multi-class classification models.

Future work can extend the architecture to consider the stacked autoencoder's contribution at each layer for an ensemble model. Convolutional neural networks can also be applied on raw time-series sensor data to exploit the data's locality and stationarity features. We also believe that this architecture's generality can further be improved by exercising these soft sensing models on a diverse set of equipment with their own function-specific sensors. In the meantime, we've made Seagate Technology's manufacturing and sensing datasets available, with the hope that it will enable the research community to develop more scalable and applicable AI techniques.

4.2 Soft Sensing Transformer: Hundreds of Sensors Are Worth a Single Word

4.2.1 Introduction

(This work has been published on IEEE International Conference on Big Data [126] ©20XX IEEE.)

In the last decades, smart sensor development has attracted a lot of attention from government, academia, and industry. In 2013, Germany proposed the concept of Industry 4.0, the main aim of which is to develop smart factories for producing smart products. In

September 2020, the US government announced that they will be providing more than $1 billion towards establishing research and hubs for Industry 4.0 technologies. Singapore's current five-year SU$13.8 billion R&D is injecting more funds into expanding fields such as advanced manufacturing. China's "China Manufacturing 2025" goal is also to make the manufacturing process more intelligent. These initiatives require that we possess better sensing technologies to understand and drive our processes. Sensors have the potential to capture information about process variables which can be exploited by data-driven techniques for smarter manufacturing process monitoring and control.

As the industrial process has become more complicated, and the size of available data has increased dramatically, the body of research into deep learning methods with applications in the soft sensing field has grown in parallel. A soft sensor is a software application or even an algorithm that functions like a sensor to detect a product's properties. For example, in our experiments with the Seagate data, we used soft sensors to predict wafer quality, a particularly difficult and expensive metric to measure. Many other factories use soft sensors to detect various properties for their products, such as chemicals or control systems. A recent survey on deep learning methods for soft sensors [103] has illustrated the significance of deep learning applications and reviewed the most recent studies in this field. The deep learning models are mostly based on autoencoders [65], the restricted Boltzmann machine [98], convolutional neural networks [60], and recurrent neural networks [91]. The applications for these models vary from traditional factories [119] to wearable IoT devices [85, 86].

There have been a variety of novel deep learning models, such as variational autoencoder models, which attempts to enhance the representation ability or to augment the data [43, 44], a semi-supervised ensemble learning model that quantifies the contribution of different hidden layers in a stacked autoencoder [102], and a gated convolutional transformer neural network that combines several state-of-art algorithms together to deal with a time series dataset [31]. As the deep learning models become more and more complex, their capabilities to handle complex processes and large datasets also increase. However, in these studies, researchers are still using very small datasets such as data from a wastewater treatment plant or a Debutanizer column [99, 28], both of which contain low-dimensional data with only hundreds to thousands of data samples. These types of small datasets are not sufficient to illustrate the effectiveness of these advanced, deep learning models, which contain millions of parameters. As a solution to this issue, we collected gigabytes of numerical sensor data from Seagate's wafer manufacturing factories in the US and Ireland. These datasets contain high-dimensional time series sensor data that is collected directly from the Seagate wafer factories, with only the necessary anonymization, and these data are big, complex, noisy, and impossible to interpret in their raw form by humans. In this article, we evaluate a soft sensing transformer model against the most commonly-used methods applied to soft sensing problems – including models based on autoencoder and LSTM [39]. We maintained the key components of the original transformer model and modified other parts of the architecture to fit into our datasets and tasks.

The transformer model, since proposed in 2017 [107], together with its derivatives such as BERT[18], have been the most active research topic in the natural language processing (NLP) field, as well as being the top performer in most NLP tasks[64]. Due to its extraordinary representative capability, the transformer model has also shown equally high performance in the computer vision area [38]. First proposed in 2020, vision transformer [21] and its variants have achieved the state-of-art performances on many computer vision benchmarks such as image classification, semantic segmentation and object detection [66, 122].

From texts in NLP, which can be regarded as categorical data, to images (two- or three-dimensional integer values) in computer vision, a natural further extension would be sensor data, which is time series data with continuous floating numbers. While the Bayes error rate [30] in both NLP and computer vision tasks are usually defined as human-level performance, our soft sensing task is impossible for a human to classify based on the hundreds of sensor values. In this section, we show that Transformer architecture not only works great for natural language and images, but also for numerical data, and it is able to represent the data that is not interpretable by humans. As an outgrowth of this research, we have also explored this soft sensor problem from other aspects such as spatial correlation using convolution or graph neural networks [45, 116] and feature importance [87].

The rest of this section is organized as the following: We discuss the soft sensing transformer model in subsection 4.2.2, describe several industrial soft sensing datasets in subsection 4.2.3, show the results of the soft sensing transformer on these datasets in subsection 4.2.4, and make discussions and conclusions in subsection 4.2.4.

4.2.2 Methodology

While implementing the soft sensing model, we followed the original transformer architecture as closely as possible. We modified the module's input module to fit the time series sensor data and modified the output module for multi-task classification problems. This is the first study to use these deep learning methods to find benchmark results on these large-scale sensor datasets, as well as the first study to apply the transformer model on large-scale numerical sensor data.

Soft Sensing Transformer (SST)

The architecture of the soft sensing transformer model is illustrated in Figure 4.5. Given that the format of time series sensor data is different from texts, we used a dense layer for the embedding at the starting point, which reduced the dimension of the high-dimensional sensor data. After this layer, the data format is the same as embedded sentences so that it can be fed into the transformer encoder without any modifications. We added a positional encoding using sine and cosine functions of different frequencies to cover the information of relative positions of different time steps, as shown in Equation. 4.9 and Equation 4.10.

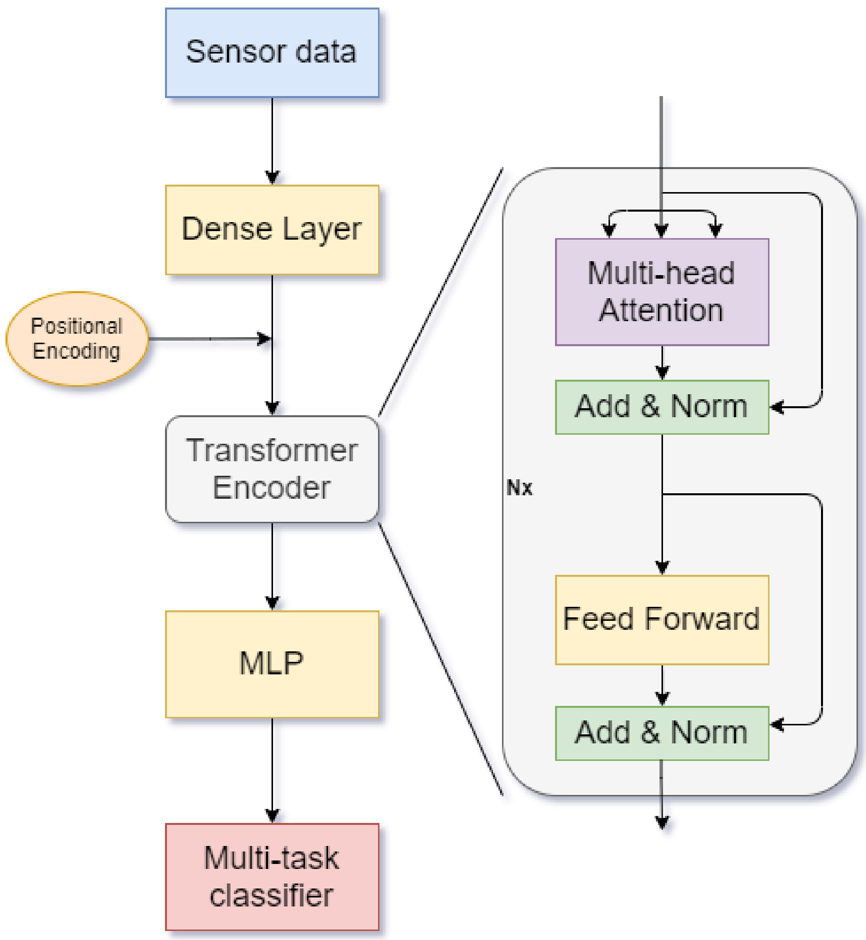


Figure 4.5: Architecture of the soft sensing transformer model

PE stands for positional encoding, pos is the position of a time step, and $dmodel$ is the dimension of embedded vectors.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/dmodel}) \tag{4.9}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/dmodel}) \tag{4.10}$$

Because the SST model requires a fixed size of input data, we added padding to those samples with too few time steps, so that each sample has the same time length. In the raw data, we chose the time length as 99 percentiles of the sequence lengths to cover most of the data and exclude outliers. We also applied the padding masks accordingly. In the encoder, multi-head scaled dot product attention, we designed feed forward and residual connections in the same way as in the original transformer paper [107]. The multi-head attention is described as in Equation 4.11. The query, key and value are projected to h heads with the weight matrices W_i^q, W_i^k, W_i^v. Each head has a dimension of $dmodel/h$, and we calculated a scaled dot-product attention for each head. We then concatenated the heads and projected them back to the original shape.

$$MHA(Q,K,V) = Concat(softmax(\frac{QW_i^q(KW_i^k)^T}{\sqrt{d_k}})VW_i^v)W^o \tag{4.11}$$

The Seagate datasets contain measurement pass/fail information, so we built the SST model as a classification model. After the encoder blocks, we attached a multi-layer perceptron (MLP) classifier on top, after a global average pooling. Because of the data's intrinsic complexity, the classifier comprises a few individual binary classifiers. These binary classifiers partly share the input data and may be correlated with each other, resulting in an inter-correlated, multi-task problem (further discussed in Subsection 4.2.3). In order to achieve the best performance in the multi-task learning, we applied a weighting method based on uncertainty [55], and defined the combined loss function as Equation 4.12:

$$J = \sum_i (\frac{1}{\sigma_i^2} J_i + log\sigma_i) \tag{4.12}$$

where J is the total loss, J_i is the loss of the i_{th} classification task, and σ_i is the uncertainty of the i_{th} classification loss, which is trainable during the model fitting.

Optimization

In industrial settings, the data are highly imbalanced. As a classification model, we found only one to two percent of the data samples as positive. To deal with this imbalance, we experimented with both weighting methods and data sampling algorithms like SMOTE

[13]. In our experiments, we found that class weighting provided the best efficiency and performance. We calculated the weight of the j_{th} task, label t (0 or 1) based on the number of samples:

$$w_j^t = \frac{N}{2m * n_j^t} \tag{4.13}$$

in which N is the total number of samples, m is the number of tasks, and n_j^t is the number of samples for label t in the j_{th} task.

Combined with the uncertainty-based multi-task learning in Equation 4.12, we defined the final loss function of SST model as weighted cross entropy:

$$J = \sum_{j}^{m} \sum_{t=0}^{1} [\frac{1}{\sigma_{jt}^2} \sum_{i}^{n_j^t} (y_{ijt} * log\hat{y}_{ijt}) + log(\sigma_{jt})] \tag{4.14}$$

where y_{ijt} and $\hat{y}_{ijt}$ are the true labels and predicted probabilities for the i_{th} sample in task j for label t. Note that the cross-entropy loss is calculated in a multi-label classification manner and the loss for positive and negative cases were computed separately. We take $y_{ij1} = 1$ for positive samples, and $y_{ij0} = 1$ for negative samples. The weights for the positive and negative cases in a single binary classification task were also further tuned by σ_{jt}, which is the uncertainty or variance of the loss for label t in task j. In this multi-task learning setting, we have $2m$ 'tasks' for the m binary classifications.

Activation Functions For the transformer encoder part, we applied a ReLU activation function [80] in the feed forward layer, which consisted of two dense layers that project the $dmodel$ dimensional vector to dff dimension and project back to $dmodel$ dimension, respectively. The ReLu activation function was set for the first dense layer in the feed forward layers. As for the MLP classifier, we applied sigmoid activation functions for all three layers because we found that it produced more stable results than ReLu in this case.

Regularization We applied L2 regularizers to all the dense layers in SST model, with a regularization factor of 10^{-4}. We also applied Dropout [100] to residual layers and embedding layers. We further applied dropout to each layer in the MLP block except for the final prediction layer. All dropout ratios were kept the same and we performed a grid search in $[0.1, 0.3, 0.5]$ to find the best dropout ratio.

Optimizers We experimented with two kinds of optimizers: a default adam optimizer [56] with a fixed learning rate, and a scheduled adam optimizer as used in the original transformer paper [107]. The learning rate scheduled optimizer has shown a more stable result, so we've continued to use it in further experiments.

For the scheduled adam optimizer, the parameters were set as $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. The learning rate was varied during the training process based on Equation 4.15. d is *dmodel* in the SST model, *step* is the training step, and *warmup* is set as 4000. We added an extra factor to tune the overall learning rate. We performed a grid search for the factor in $[0.1, 0.3, 0.5]$ to find the optimal factor.

$$lr = factor * d^{-0.5} * min(step^{-0.5}, \frac{step}{warmup^{1.5}}) \tag{4.15}$$

Hyper-parameter Tuning As shown in Table 4.2, we tuned seven hyper-parameters using a grid search. These hyper-parameters included the number of the encoder block (*n_layer*), the size of embedding layer (*dmodel*), the size of feed forward layers (*dff*), the dropout ratio, the learning rate factor as in Equation 4.15, the batch size, the number of heads for the multi-head attention layer (*n_heads*), and whether or not to use the uncertainty based weighting as in Equation 4.12. For the grid search process, we randomly sampled a smaller data size from the datasets, which contained 5000 samples for training and 3000 for validation. We selected the best model based on the validation results, evaluated by the area under a Receiver Operating Characteristic Curve (ROC-AUC) [25].

Table 4.2: Hyper-parameter search space

Hyper-parameter	Values
n_layers	2, 3, 4
dff	32, 128, 512
dmodel	64, 128, 256
dropout_ratio	0.1, 0.3, 0.5
learning rate factor	0.1, 0.3, 0.5
batch size	512, 1024, 2048
n_heads	1, 2, 4
uncertainty based weighting	on, off

Early-stopping Instead of setting an epoch number, we used an Keras early-stopping callback method in the training processing, with a patience of 100 epochs, and $restore\_best\_weights = True$. In this way, we obtained an optimized epoch number for each experiment without manually tuning.

Hardware All the models were trained on an AWS instance with an NVIDIA Tesla V100 SXM2 GPU. It took around 20ms for each step, and about 30 minutes for the entire training. The grid search for hyper-parameters took about 36 hours. All the models were written with TensorFlow [1] version 2.2 with Keras[15].

4.2.3 Data

To fill the gap of publicly available large-scale soft sensing datasets, we queried and processed several gigabytes of datasets from Seagate manufacturing factories in both the US and Ireland. These datasets contain high-dimensional time series sensor data coming from different manufacturing machines.

As shown in Figure 4.6, to fabricate a slider used for hard drives, an AlTiC wafer goes through multiple processing stages including deposition, coating, lithography, etching, and polishing. Different products have different manufacturing lines; Figure 4.6 shows a simplified manufacturing process. After each processing stage, the wafer is sent to metrology tools for

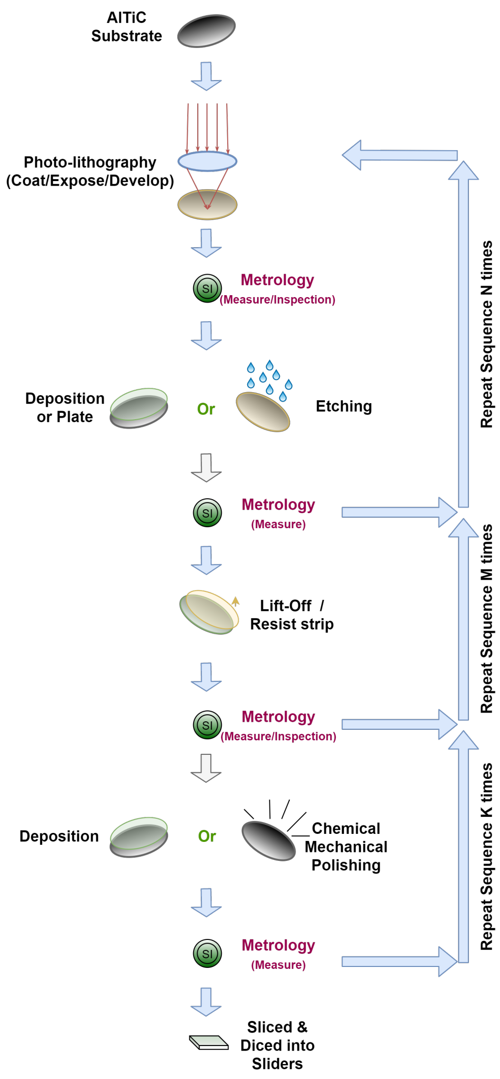


Figure 4.6: High-level workflow of wafer manufacturing. Each wafer goes through multiple processing stages, and each stage posesses corresponding metrology, in which a few quality control measurements are performed. The measurement results are used to decide whether the wafer is in acceptable shape to proceed to the next stage. Figure from `https://github.com/Seagate/softsensing_data`.

quality control measurements. A metrology step may generate a single or multiple different measurements, each of which could have varying degrees of importance.

These processes are highly complex and are sensitive to both incoming as well as point-of-process effects, and they employ a significant amount of engineering and systems resources to monitor and control the variability intrinsic to those factors known to affect a process.

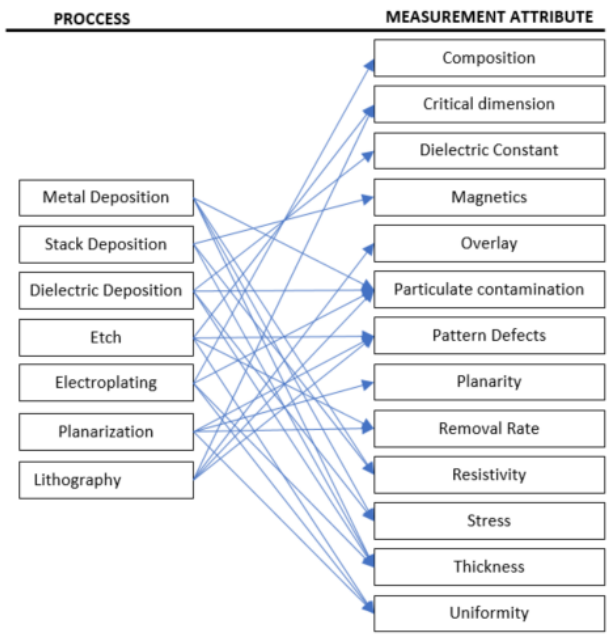


Figure 4.7: Overview of the main process categories of processes and their corresponding critical measurement variables per each category. Figure from `https://github.com/Seagate/softsensing_data`.

Metrology serves a critical function in managing these complexities for early learning cycles and quality control. However, this comes at high capital costs, increased cycle time, and there is considerable overhead involved in setting up the correct recipes for measurements, appropriate process control, and workflow mechanisms. In each processing tool, there are anywhere from dozens to hundreds of on-board sensors in the processing machines that monitor the state of the tool. These sensors collect information every few seconds, and all these sensing values are collected and stored along with the measurement results.

As shown in Figure 4.7, one time series of sensor data are mapped to several measurements, and the same measurement can be applied to multiple processing sensor data points. Each measurement contains a few numerical values to indicate a wafer's condition, and the pass or fail decision is made based on these numbers. For simplicity's sake, we only covered the pass/fail information for each measurement, so that each time series data sample was mapped to several binary classification labels, resulting in a multi-task classification problem. On the other hand, some of measurements are linked to multiple processing stages, so that the SST model can learn the representations from one stage and be applied to another stage when it's trained on data covering all the stages. Given this inter-correlation, training such a multi-task learning SST model leads to better performance when compared with the method of individually training the measurement tasks. From an industrial application perspective, the former method also allows for simpler maintenance and scalability, in that in involves a single model instead of multiple models.

The datasets in this paper cover 92 weeks of data. We took the first 70 weeks as training data, the following 14 weeks as validation data, and the last eight weeks as testing data. We prepared the datasets by querying from raw data and performing some necessary pre-processing steps. While the sensors were collecting data every few seconds, there were a lot of redundancies, so we aggregated the data into short sequences. In each processing stage, a wafer goes through a few modules, and we aggregate the data by the module and get short time sequences. Other pre-processing steps on the data included a min-max scaling, imputation with neighbors, one-hot encoding for categorical variables, and necessary anonymization. The min-max scaler was fit only on training data and applied on the entire datasets. Imputation was done by filling the missing values first by its neighbors (a forward filling followed by a backward filling) if non-missing values existed in the same processing stage, otherwise filling by the mode of all the data. Categorical variables such as the processing stage information, the type of the wafer, the manufacturing machine in function, were one-hot encoded and concatenated to the sensor data as model input. As for the anonymization, only confidential information like the data headers was removed.

Using data in different time frames for training and testing reflects the SST model's application prospect, because it is in this way that the model can be directly deployed into factories once it performs well enough in testing data. However, this setting also makes it harder for the model to achieve a high performance because, in "factory reality," there are

Table 4.3: Summary for the datasets: The number of positive and negative samples for each task

	P1		P2		P3	
Task	pos	neg	pos	neg	pos	neg
1	295	8328	256	6433	109	2496
2	40	12747	773	26811	335	12857
3	291	56198	2069	78844	46	1026
4	188	14697	582	27809	15	4180
5	568	40644	247	9652	300	22254
6	863	84963	884	27337	166	40811
7	2501	153970	2108	53921	875	75706
8	490	2919	2016	77473	1097	18890
9	104	29551	644	23305	537	4247
10	57	10813	270	25651	1547	129914
11	306	47219	3792	354328	-	-

too many uncontrollable factors and the data distribution between the training data and the testing data may differ from each other.

After the preprocessing, these datasets were turned into Numpy format, which only include numerical values without any headers. Input files are rank 3 tensors with dimension (n_sample, time_step, features), and outputs are rank 2 tensors with dimension (n_sample, 2*n_tasks). Each binary classification task has two columns in the output file: the first column contains negative cases and the second contains positive cases.

Three datasets are covered in this paper. They are from slightly different manufacturing tool families, and each has different processing stages and corresponding measurements. The number of samples for each measurement is summarized in Table 4.3. More detailed information for each tool family is described below, and all the data are available at `https://github.com/Seagate/softsensing_data`.

Dataset P1 The sensor data were generated by a deposition tool that include both deposition and etching steps. There are 90 sensors installed in the tool and they capture data at a frequency of about every second. The critical parameters measured in this family of tools are magnetics, thickness, composition, resistivity, density, and roughness.

After the pre-processing mentioned above, there were $194k$ data samples in training, $34k$ samples in validation, and $27k$ samples in testing data. Each sample had two time steps, with 817 features. Some of the second time steps were missing and replaced with zero padding, and the 817 features came from 90 sensors, one-hot encoded categorical variables. The features included each wafer type, the processing stages, specific manufacturing tools, etc., and a padding indicator as the last feature.

For the labels, there are eleven individual measurement tasks, each is a binary classification. We set the model output dimension as 22 to have separate predictions for negative and positive probabilities. As shown in Table 4.3, the datasets are highly imbalanced – about 1.2 percent of the samples have positive labels.

Dataset P2 This second dataset contained data generated by a family of ion milling (dry etch) equipment, which utilizes ions in plasma to remove material from the wafers' surfaces. There were 57 sensors included in this dataset, and the critical parameters measured for this family of tools were similar to those measured by the tools used in Dataset P1 tools, but the process used slightly different machines.

There are $457k$ training samples, $80k$ validation samples, and $66k$ testing samples in the dataset. For this dataset, there was no time series information, but we treated it as a single time step to fit into the same SST model. This dataset was more complex in terms of categorical variables, resulting in 1484 features in total.

The number of measurement tasks was eleven, with an output dimension of 22, and about 1.9 percent of the samples were positive, as shown in Table 4.3. Note that these eleven tasks are not the same as those in P1.

Dataset P3 The last dataset was generated by sputter deposition equipment, which contains multiple de-position chambers, each with unique targets. The number of sensors was 43, and the critical parameters measured were the same but, again, calculated using different machines.

There were $205k$ training data samples, $35k$ for validation, and $20k$ for testing. The maximum time series length was two, with outliers filtered out and short series padded. The number of features was 498, the least among these three datasets.

The number of measurement tasks was ten, and the output dimension was 20. These tasks were not the same as those in shown in the P1 and P2 datasets. The percentage of positive cases was about 1.6 percent.

4.2.4 Results

We ran the SST models on the three datasets described in the last section. We tuned the hyperparameters within the range shown in Table 4.2, and we present the best combinations of each dataset below.

To validate the SST's effectiveness, we compared the results with two baseline models. The first baseline model was the variance-weighted multi-headed quality driven autoencoder (VWMHQAE) [123], which our team developed in 2020, as described in Section 4.1. It has been proven to work well with non-time series data in our previous experiments with similar sensor data, and therefore serves as a good baseline model for SST. Because VWMHQAE does not have an architecture to cover the time dimension, we flattened the data before feeding it into the model. We also trained a second baseline model: a bidirectional LSTM model (Bi-LSTM), which is one of the golden standard models for time series data, to obtain a comprehensive benchmark on the performance of SST.

Due to the highly imbalanced nature of the datasets, accuracy would not make much sense as a metric by which to evaluate the models. At Seagate, the most important metrics

Table 4.4: Result comparison with the baseline models for the P1 dataset

Task	SST	VWMHQAE	Bi-LSTM
1	0.70 ± 0.12	0.63 ± 0.12	$\mathbf{0.73} \pm 0.13$
2	$\mathbf{0.60} \pm 0.18$	0.54 ± 0.03	0.56 ± 0.01
3	$\mathbf{0.86} \pm 0.01$	0.77 ± 0.05	$\mathbf{0.86} \pm 0.03$
4	$\mathbf{0.91} \pm 0.01$	0.89 ± 0.01	0.88 ± 0.01
5	$\mathbf{0.55} \pm 0.03$	$\mathbf{0.55} \pm 0.02$	0.51 ± 0.05
6	0.53 ± 0.05	$\mathbf{0.57} \pm 0.03$	0.50 ± 0.03
7	0.64 ± 0.02	$\mathbf{0.65} \pm 0.01$	0.64 ± 0.01
8	$\mathbf{0.82} \pm 0.03$	0.78 ± 0.04	0.78 ± 0.11
9	0.71 ± 0.09	$\mathbf{0.77} \pm 0.01$	$\mathbf{0.77} \pm 0.06$
10	$\mathbf{0.92} \pm 0.03$	0.82 ± 0.03	0.67 ± 0.23
11	$\mathbf{0.89} \pm 0.01$	0.77 ± 0.03	0.88 ± 0.01

are the True Positive Rate (TPR, also called recall or sensitivity) and the False Positive Rate (FPR, also called fall-out or false alarm ratio), because TPR implies the yield by which FPR indicates the efficiency of the manufacturing factory. However, comparing two metrics simultaneously is not intuitive, so we chose to use the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve as the main metric in this paper.

P1 Results For the P1 dataset, SST model was set as three layers, both the *dmodel* and the dff were 128, the dropout rate was 0.5, the batch size was 2048, *n_heads* was 1, the learning rate factor was 0.5, and the uncertainty-based weighting was off. The VWMHQAE model was set as three layers with hidden layer dimensions [512, 256, 128], and the Bi-LSTM model with dimension equal to dff. All models ended with a three-layer MLP classifier with all hidden dimensions as dff.

The results for the eleven tasks are summarized in Table 4.4. In seven of the tasks, SST was the best performer, especially for the high performing tasks where AUC larger than 0.8.

From the results we can also see that some of the tasks returned poor results for all three

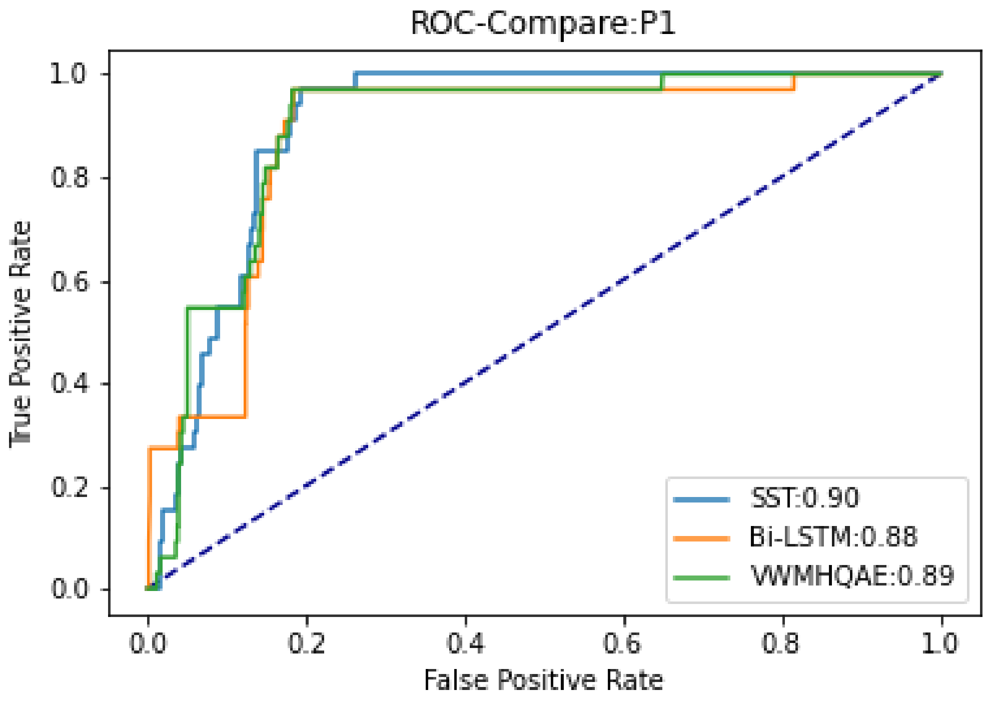


Figure 4.8: The ROC curve for SST and the baseline models on the P1 dataset, for the 4th task

models. For these tasks, it is difficult to get a high AUC value with any model due to the intrinsic complex and noisy nature. Only those measurement tasks with decent results can lead to realistic value in industry applications. This is one of the primary motivations behind our decision to open-access these datasets: Researchers all around the world are welcome to use and explore this data. This will not only help us to gain more understanding about the datasets, but also enrich the research field.

To further illustrate the results, we plotted the ROC curve for the task with highest AUC value, as shown in Figure 4.8. SST has a higher score than the two baseline models, and the curve is smoother, indicating a more even distribution of the prediction probabilities and a finer grid in the prediction space. The source code can be found at `https://github.com/Seagate/SoftSensingTransformer`.

Table 4.5: Result comparison with the baseline models for the P2 dataset

Task	SST	VWMHQAE	Bi-LSTM
1	**0.89** $\pm$ 0.01	0.87 $\pm$ 0.02	0.88 $\pm$ 0.04
2	0.64 $\pm$ 0.01	**0.66** $\pm$ 0.02	0.59 $\pm$ 0.07
3	0.60 $\pm$ 0.06	**0.62** $\pm$ 0.01	0.55 $\pm$ 0.01
4	**0.85** $\pm$ 0.01	0.82 $\pm$ 0.01	0.84 $\pm$ 0.01
5	0.45 $\pm$ 0.07	**0.53** $\pm$ 0.10	0.52 $\pm$ 0.10
6	0.64 $\pm$ 0.02	0.71 $\pm$ 0.04	**0.72** $\pm$ 0.02
7	0.78 $\pm$ 0.01	**0.81** $\pm$ 0.02	0.79 $\pm$ 0.01
8	**0.72** $\pm$ 0.03	0.70 $\pm$ 0.09	0.69 $\pm$ 0.03
9	0.71 $\pm$ 0.12	**0.77** $\pm$ 0.08	**0.48** $\pm$ 0.14
10	**0.76** $\pm$ 0.02	0.75 $\pm$ 0.01	**0.76** $\pm$ 0.04
11	0.79 $\pm$ 0.01	0.80 $\pm$ 0.01	**0.82** $\pm$ 0.01

P2 Results The SST model was set as three layers, both *dmodel* and *dff* were 128, the dropout rate was 0.3, the batch size was 2048, *n_heads* was 1, the learning rate factor was 0.5, and the uncertainty-based weighting was on. The baseline models were the same as those used for the P1 dataset.

The results for the eleven tasks are summarized in Table 4.5. SST is the best performer in four of the tasks, including the two tasks with the best prediction performance. Same as in P1 dataset, some of the tasks have poor results for all three models due to the intrinsic complexity and noise in the dataset, but we care about the tasks with best results. In this dataset, there is only one time step, and – as expected – the VWMHQAE model, which is not designed for time series data, has shown better results when compared to the P1 data, and it performed the best in five out of the eleven tasks.

The ROC curve for the task with the highest AUC value, as shown in Figure 4.9, is very similar to Figure 4.8, showing the SST model's consistent performance of SST model over different datasets. SST is slightly smoother than the baseline models, with a higher AUC.

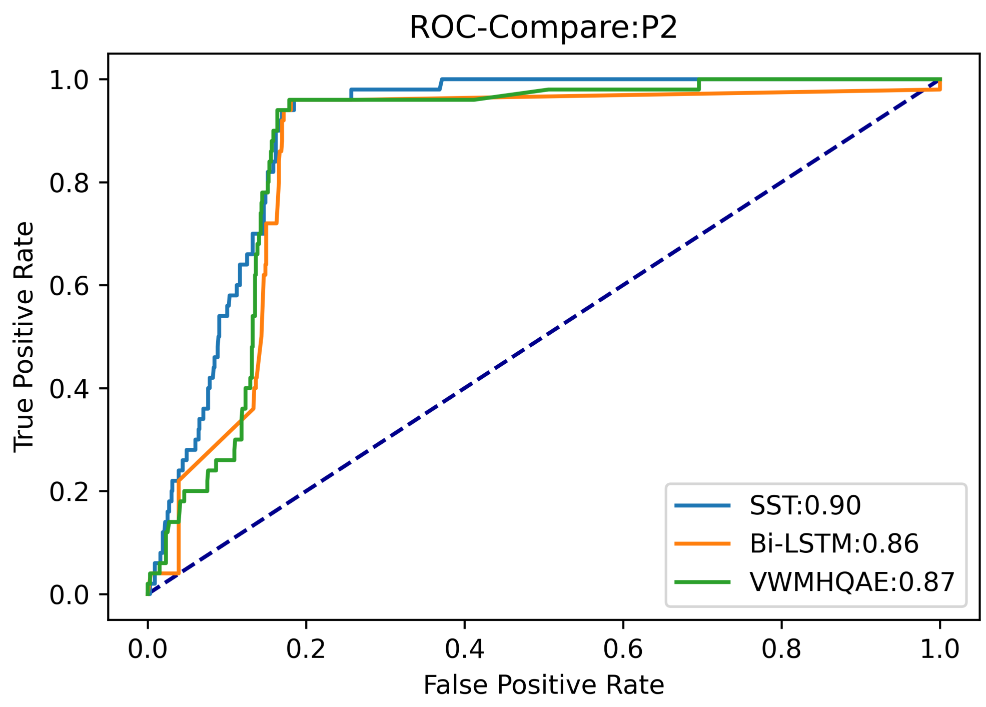


Figure 4.9: The ROC curve for the SST and baseline models on the P2 dataset, for the 1st task

P3 Results The SST model was set as three layers, both *dmodel* and *dff* were 128, the dropout rate was 0.3, the batch size was 2048, *n_heads* was 1, the learning rate factor was 0.3, and the uncertainty-based weighting was on. The baseline models were the same as those used for the P1 dataset.

The results for seven out of ten tasks are summarized in Table 4.6, because the other tasks had too few testing data samples. SST was the best performer in four out of the seven tasks, including the task with the best prediction performance. Some of the tasks have poor results for all three models and even with an AUC lower than 0.5, meaning the result was worse than a random guess. The main cause for this is the distribution shift, and we will carry out further experiments when we accumulate more data from the Seagate factories. The ROC curve for the task with highest AUC value is shown in Figure 4.10, showing the SST model's consistent performance of SST model over different datasets.

Table 4.6: Result comparison with the baseline models for the P3 dataset

Task	SST	VWMHQAE	Bi-LSTM
1	**0.42** $\pm$ 0.08	0.23 $\pm$ 0.12	0.32 $\pm$ 0.08
2	**0.94** $\pm$ 0.02	**0.94** $\pm$ 0.02	0.93 $\pm$ 0.01
3	**0.68** $\pm$ 0.08	0.53 $\pm$ 0.10	0.56 $\pm$ 0.05
4	–	–	–
5	0.53 $\pm$ 0.08	**0.65** $\pm$ 0.05	0.58 $\pm$ 0.05
6	–	–	–
7	0.60 $\pm$ 0.05	0.56 $\pm$ 0.02	**0.61** $\pm$ 0.07
8	**0.72** $\pm$ 0.05	0.62 $\pm$ 0.08	0.65 $\pm$ 0.08
9	–	–	–
10	0.81 $\pm$ 0.01	0.81 $\pm$ 0.01	**0.82** $\pm$ 0.01

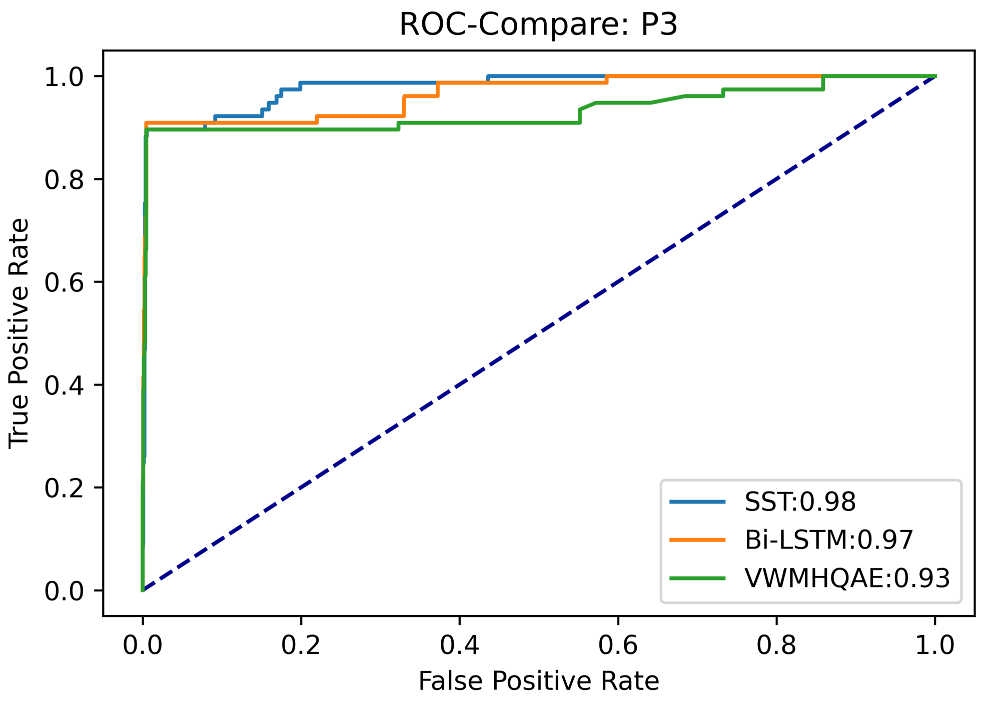


Figure 4.10: ROC curve for the SST and baseline models on the P3 dataset, 2nd task

Discussion and Conclusion

We have explored the direct application of Transformers to soft sensing. To our knowledge, we are the first to provide large-scale soft sensing datasets, and the first to benchmark the results with the transformer model in the soft sensing field. Also, this is the first time that transformer model has gone beyond human capability in the sense that the input data is not human-interpretable. We analogized the time series data as a sequence of sensor values, taking each time step as a word, and processed the sentence-like data using a standard transformer encoder exactly as in NLP tasks. This direct and intuitive strategy has shown an exciting result for our datasets that outperformed both our previous model and Bi-LSTM model.

We share these datasets with the excitement of advancing interest and work in soft sensing research and applications. We invite future work into the exploration of improving SST performance on those tasks that have proven particularly challenging in our experiments. Another future direction might involve the examination of appropriate time sequences for these datasets, and explorations of better ways to address missing data. We are working on acquiring more data with longer sequences, in order to better understand the impact of time series length in prediction quality. In the meantime, we have provided three datasets to cover a variety of sensors, and to examine the generalizability of deep learning models, and we believe these datasets can enrich the soft sensing research field, as well as serve as one of the standard tools to by which future researchers can evaluate the effectiveness of their own research.

4.3 Learning from Failures in Large-scale Soft Sensing

(This work has been published on IEEE International Conference on Big Data [124] ©20XX IEEE.)

Scientific publications typically favor studies that show successful results. Experiments that did not work either are relegated to footnotes, or worse, never get mentioned. This practice is short-sighted; the publication of failures can potentially contract research learning curves because failures are discoveries in their own right. In this section, we'll share those strategies that did not work in our soft sensing deep learning experiments, in the hope that future researchers and application engineers can find lessons and guidance from our failures. We've compiled this article from our experiments using multiple deep learning algorithms, (including autoencoder, CNN, GNN, and transformer), to predict the values of quality control variables in the wafer manufacturing process at Seagate Technology, based on the sensor data collected during the deposition and etching processes. We experimented on various pre-processing and modeling methods, and what follows is a list of some of the methods that did not return satisfactory performances: using focal loss or hinge loss to replace cross entropy; using a standard or robust scaler to replace a min-max scaler; scaling the data by groups; using oversampling methods such as random sampling or SMOTE, or; selecting features based on domain knowledge. We hope that these experiments will inspire intuition in future researchers and help save time in their experiments and research. We acknowledge that these conclusions are drawn only from our experiments, and therefore accept the possibility that the same experiments may work in different contexts. We welcome all kinds of discussion or collaboration.

4.3.1 Introduction

Based on the Seagate wafer manufacturing sensor data, we have been developing models for Seagate manufacturing lines [123]. While we found many methods – such as regularization, multi-task learning and class weighting – contributing positively to the model performance; we also discovered certain methods that didn't work in our experiments. These unsuccessful methods were just as valuable as the successful methods because both of them have guided our subsequent experiments.

During our study, we had been focusing on a classification problem based on the Seagate factory wafer sensor data. The details of the problem and datasets are described at `https://github.com/Seagate/softsensing_data`. The input data are high dimensional time-series data containing about 100 sensors and categorical variables with hundreds of values, and the output data are the classification results for multiple measurement tasks.

While the successful methods are described and implemented in references [123, 125, 45, 116, 87], here we present those experiments that did not work, but which still provided learning to further our aim to advance soft sensing in industrial applications.

4.3.2 Learning from Failures

In this section we list some methods that did not work well in our experiments. We tested the following methods and compared the model performance with metrics including recall and AUC-ROC. The following methods either decreased the performance or increased model complexity without any performance improvement.

Hinge Loss

For a classification problem, the two most commonly-used algorithms are logistic regression and support vector machines [17], with the key difference between them being the loss functions: hinge loss vs. cross entropy.

Hinge loss maximizes the margin of classification boundary and ignores the correct predictions, while cross entropy penalizes heavily on the wrong predictions and also counts for the correct predictions. Hinge loss has shown both good performance on high-dimensional data and a reduced risk of over-fitting.

However, in our experiments using these two loss functions in our deep learning classification model, hinge loss function showed unstable results and tended to be under-fitted. More specifically, with hinge loss, the model often predicted all the results as negative.

There are two possible reasons for this result. The first is that hinge loss has zero loss for some of the data, so it cannot handle imbalance very well, even when class weighting is applied. The second reason is that the hinge loss function does not have a continuous

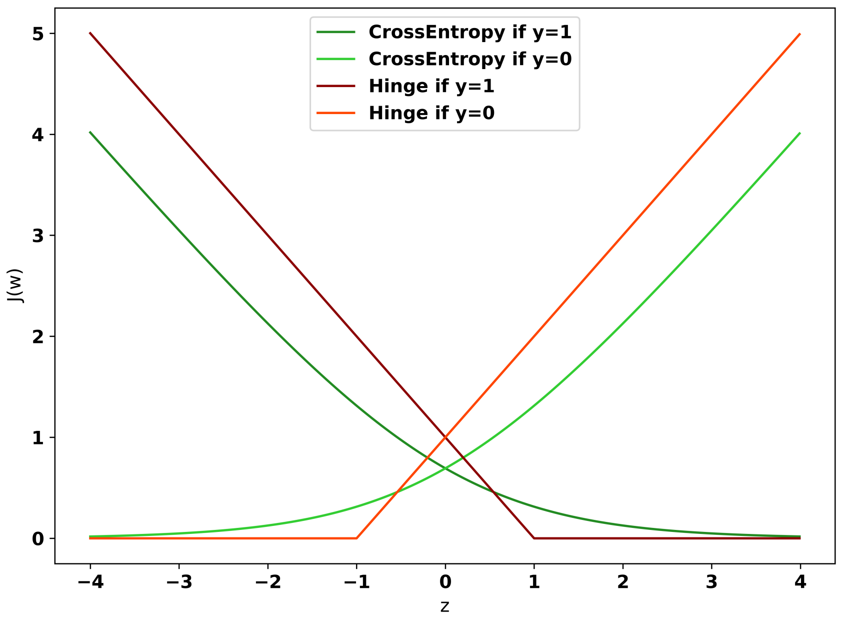


Figure 4.11: Comparison between cross entropy and hinge loss

gradient, so it is easy to get stuck at local minimum, where the model goes to the state where it predicts everything as negative, and stops updating the parameters.

We also experimented with focal loss function, and it showed similar results as hinge loss.

Standard or Robust Scaler

The input sensor data is composed of hundreds of different sensors. These sensors have different numerical value scales, so we needed to normalize them to the same range for faster training and better convergence.

There are a few commonly-used scalers: min-max, standard, and robust scalers. We used the Scikit-Learn package for all of these. The min-max scaler constrains all the data to the range of [0,1], the standard scaler normalizes data based on both the average and standard deviations, and the robust scaler normalizes data based on their median and percentiles.

Robust scaler is called 'robust' because, in it, medians are not affected by outliers. While standard and robust scalers consume more computing resources, they did not show any

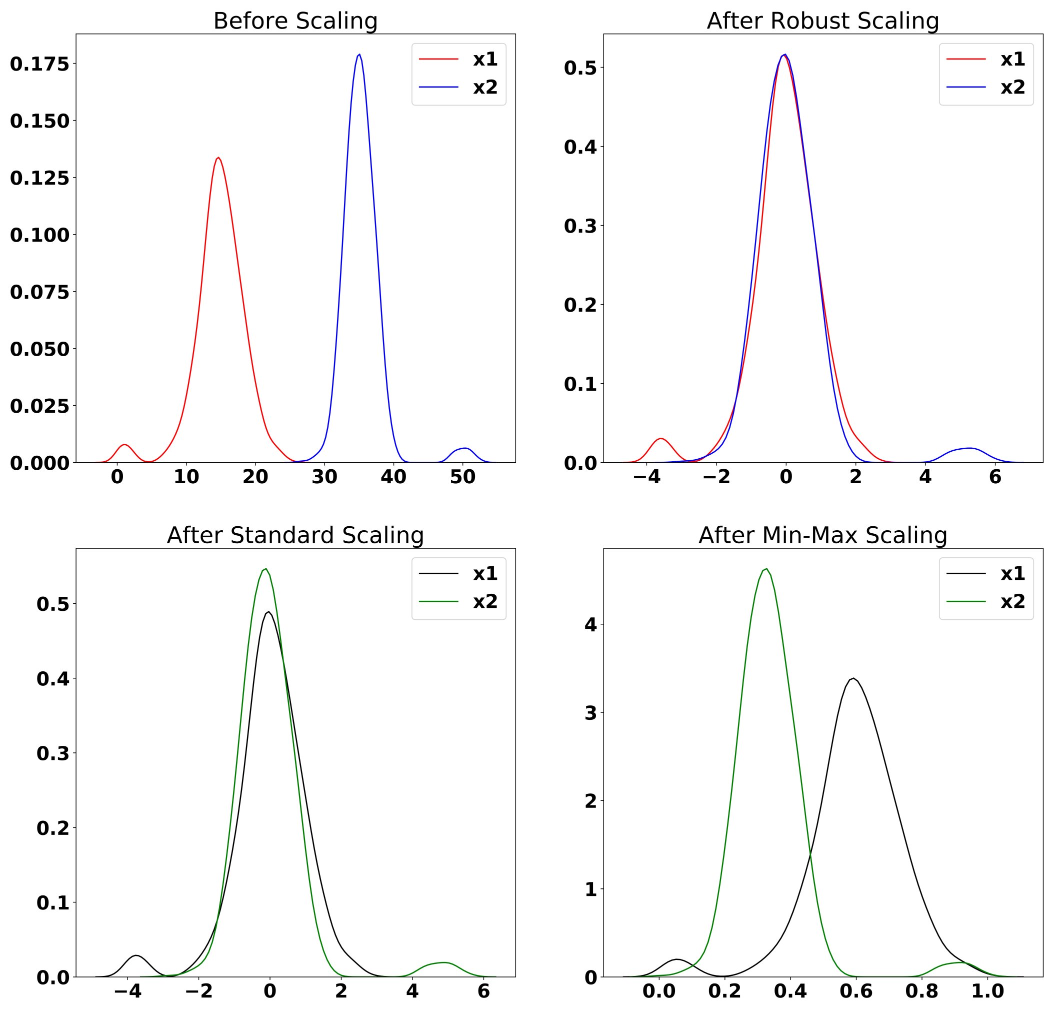


Figure 4.12: Comparison between robust, standard and min-max scalers

improvement in our experiments, and the results for all three scalers are identical. These results are not as expected because the real-world sensor data had untypical distributions. It is possible that the outliers do not contribute overmuch to the results, or the deep learning models are able to capture the information even when the normalization has not been performed perfectly.

Grouped Scaler

Other than the sensor data, the datasets also contained a few categorical features. Based on these categorical features, the data can be separated into groups, each group with different properties. Scaling the data by groups is intuitively more reasonable than scaling all the data simultaneously.

We experimented grouping the data by several different categorical variables and performed normalization within each group, respectively. However, the model's performance became worse when compared to normalizing the entire dataset. On the other hand, the grouped scaler method also made the deployment a bit more complex because we then needed to scale the new input data based on its categorical features before feeding it into the deep learning model. Nevertheless, it is still counter-intuitive that the grouped scaler resulted in worse model performance. We hypothesize that it this due to the imputation, in which the data were mixed among different groups, making it difficult for the grouped scaler to effectively scale each group. We are exploring imputation methods and will apply grouped scaler again to further examine these results.

Oversampling

One of the biggest challenges in this classification problem is the data imbalance. Only one to two percent of the labels are positive while the rest are negative. To obtain a reasonable prediction on both positive and negative labels, we experimented with the most common algorithmic and data manipulation methods: class weighting and oversampling, respectively. Class weighting is the method in which an extra weight is applied to the loss function based on the number of positive/negative labels, in order to achieve a balance. Oversampling methods generate new data points in order to obtain a comparable number of positive/negative labels, thus resolving the imbalance issue at the data level. We tested using two oversampling methods: random oversampling, which randomly duplicates the positive data, and SMOTE

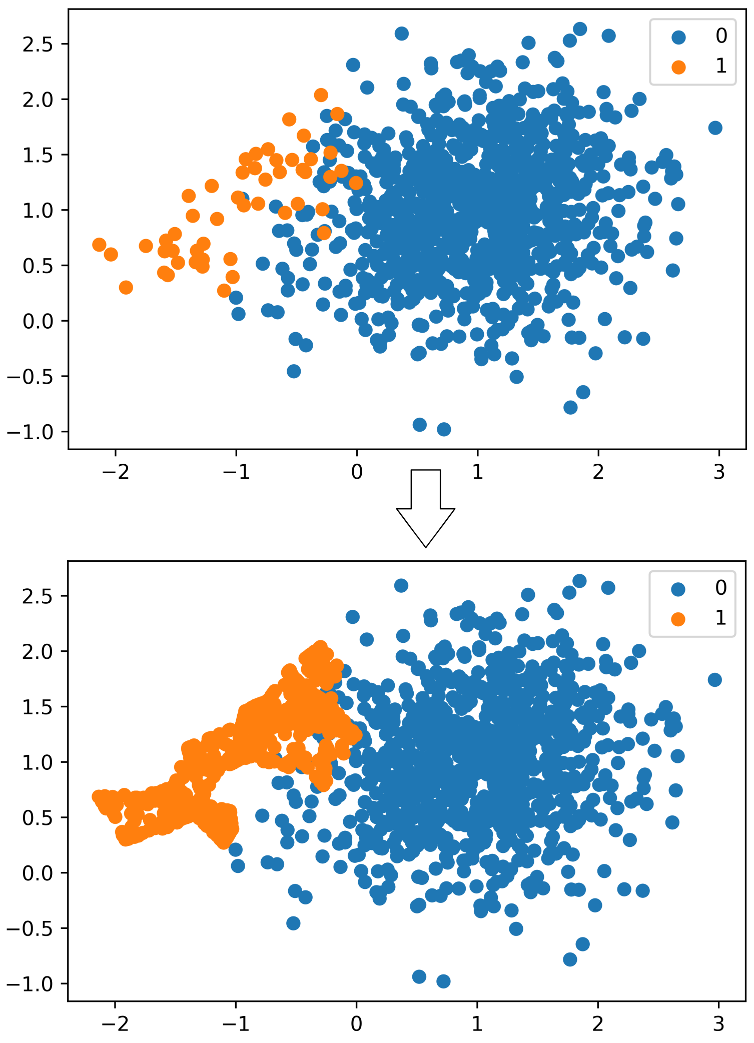


Figure 4.13: Illustration of SMOTE

[13] which generates new data points based on a linear combination of original data. While the oversampling methods made the dataset about twice as large and slowed the training process, they still performed worse than class weighting.

In theory, random sampling is equivalent to class weighting because both methods count the training loss multiple times, either by duplicating the loss itself or the data points. SMOTE generates new data and brings a smoothing effect to the model training, but it does not recognize the high-dimensional data's non-linearity, the result of which is that the newly generated positive data are mixed with the original negative data. Given this result, we concluded that class weighting was the optimal method to handle the data imbalance in such classification problems. Our reasoning was that because simple oversampling methods are not able to capture the data's non-linearity, and complex oversampling methods are already classifiers if they capture all the non-linear relation.

Feature Selection

The datasets contain hundreds of sensors, not all of which are informative for the classification tasks, and some of this sensor data may be too noisy. In traditional machine learning modeling, feature engineering would be a key step for these kinds of datasets. However, for deep learning methods, the burden of information representation is shifted to the model itself. Thus, feature selection was not a critical factor in our case.

To verify feature engineering's critical importance, we experimented on a set of sensors that were selected by manufacturing engineers, which they categorized as the key sensors based on their domain knowledge. Our experiments with this data yielded results confirming the hypothesis that feature selection is not helpful for deep learning models. The knowledge-based feature selection did reduce the size of the data and accelerated the training process, but came with a sacrifice in performance.

4.3.3 Discussion

In this section, we summarized several methods that did not work in our soft sensing deep learning experiments. While these methods failed to improve our models, the experience we gained was even more valuable for us because it gave us directions regarding what to do next and may possibly save a lot of time during future experiments. We hope that sharing this kind of experience with researchers working in similar areas and inviting them to share their learning from failed soft sensing experiments, will hasten discoveries in the field.

CHAPTER 5
SUMMARY AND FUTURE DIRECTIONS

The 21st century has witnessed a prosperity of text mining techniques and applications. The success of text mining would not have been possible without the proliferation of large-scale datasets. This paper has described various datasets from different fields, and made use of them by applying state-of-the-art text mining techniques to further develop models and applications, which were already shown to be efficient and effective. These models made it possible to automate and scale up the deployment of AI technology in the medical clinical field in lower-resource areas, in chemistry naming systems, and in traditional manufacturing.

Chapter 2 presents the results of the diagnosis assistant system which we developed based on Word2Vec and decision tree models. The datasets used here include the medical claims from millions of patients, gigabytes of text corpus from medical literature, and a knowledge database from medical professionals. We also showcased the model's application in the format of an Android app.

Chapter 3shows the results of chemical substructures' fuzzy frequency in terabytes of Elsevier chemical literature and PubChem datasets which covers the names and structures of almost all the existing chemical compounds in different formats. For this, we used the BLAST method, which was originally used for comparing biological sequences; here, we used it to compare sequences of words.

Chapter 4 displays the results of applying text mining techniques on semiconductor manufacturing lines. We used the autoencoder and transformer techniques, and we altered both model architectures to better fit the datasets. The data contained gigabytes of numerical time sequence sensor data, which we gathered from the factories.

Some of the aforementioned projects are still on-going; my collaborators and I will continue to work on them. In the meantime, there is much more to explore in these fields. In clinics, the applications we developed are in their early stages, and there are chances to scale them forward with new technologies and new datasets in the near future. In chemistry, we have only covered organic compounds; in the next step, we can expand these methods to inorganic compounds. In manufacturing, the future work could focus on more efficient models that train and run faster to meet changing industry criteria.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[3] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. *An efficient k-means clustering algorithm*. Syracuse University, 1997.

[4] Sofia J Athenikos and Hyoil Han. Biomedical question answering: A survey. *Computer methods and programs in biomedicine*, 99(1):1–24, 2010.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] G Octo Barnett, Kathleen T Famiglietti, Richard J Kim, Edward P Hoffer, and Mitchell J Feldman. Dxplain on the internet. In *Proceedings of the AMIA Symposium*, page 607. American Medical Informatics Association, 1998.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

[8] Stephen A Berger. Gideon: a computer program for diagnosis, simulation, and informatics in the fields of geographic medicine and emerging diseases. *Emerging infectious diseases*, 7(3 Suppl):550, 2001.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.

[11] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[12] Amos Cahan and James J Cimino. A learning health care system using computer-aided diagnosis. *Journal of medical Internet research*, 19(3):e54, 2017.

[13] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[14] Applied Chemistry. *Nomenclature of Organic Chemistry: Definitive Rules for Section C. Characteristic Groups Containing Carbon, Hydrogen, Oxygen, Nitrogen, Halogen, Sulfur, Selenium, And/or Tellurium*, volume 2. Butterworths, 1965.

[15] François Chollet et al. Keras. `https://keras.io`, 2015.

[16] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[23] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[24] Meherwar Fatima and Maruf Pasha. Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, 9(01):1, 2017.

[25] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[26] Ronen Feldman and Ido Dagan. Knowledge discovery in textual databases (kdt). In *KDD*, volume 95, pages 112–117, 1995.

[27] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.

[28] Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo, Maria G Xibilia, et al. *Soft sensors for monitoring and control of industrial processes*, volume 22. Springer, 2007.

[29] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[30] K Fukunada. Introduction to statistical pattern recognition. *Academic Press Inc., San Diego, CA, USA*, 1990.

[31] Zhiqiang Geng, Zhiwei Chen, Qingchao Meng, and Yongming Han. Novel transformer based on gated convolutional neural network for dynamic soft sensor modeling of industrial processes. *IEEE Transactions on Industrial Informatics*, 2021.

[32] Robert D Gibbons, Giles Hooker, Matthew D Finkelman, David J Weiss, Paul A Pilkonis, Ellen Frank, Tara Moore, and David J Kupfer. The computerized adaptive diagnostic test for major depressive disorder (cad-mdd): a screening tool for depression. *The Journal of clinical psychiatry*, 74(7):0–0, 2013.

[33] Robert D Gibbons, David J Weiss, Ellen Frank, and David Kupfer. Computerized adaptive diagnosis and testing of mental health disorders. *Annual review of clinical psychology*, 12:83–104, 2016.

[34] Siddharth Gopal and Yiming Yang. Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322, 2010.

[35] Tracy D Gunter and Nicolas P Terry. The emergence of national electronic health record architectures in the united states and australia: models, costs, and questions. *Journal of medical Internet research*, 7(1):e3, 2005.

[36] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*, pages 53–65. Springer, 2001.

[37] Kai Han, Yunhe Wang, Chao Zhang, Chao Li, and Chao Xu. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE, 2018.

[38] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[40] Thomas Hoffman. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval, 1999*, pages 50–57, 1999.

[41] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62. Citeseer, 2005.

[42] Chiou-Jye Huang and Ping-Huan Kuo. A deep cnn-lstm model for particulate matter (pm2. 5) forecasting in smart cities. *Sensors*, 18(7):2220, 2018.

[43] Yu Huang, Yufei Tang, James VanZwieten, and Jianxun Liu. Reliable machine prognostic health management in the presence of missing data. *Concurrency and Computation: Practice and Experience*, page e5762, 2020.

[44] Yu Huang, Yufei Tang, and James Vanzwieten. Prognostics with variational autoencoder by generative adversarial learning. *IEEE Transactions on Industrial Electronics*, 2021.

[45] Yu Huang, Chao Zhang, Jaswanth Yella, Xiaoye Qian, Sergei Petrov, Yufei Tang, Xingquan Zhu, and Sthitie Bom. Grassnet: Graph soft sensing neural networks. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[46] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[47] Elima Jedy-Agba, Maria Paula Curado, Olufemi Ogunbiyi, Emmanuel Oga, Toyin Fabowale, Festus Igbinoba, Gloria Osubor, Theresa Otu, Henry Kumai, Alice Koechlin, et al. Cancer incidence in nigeria: a report from population-based cancer registries. *Cancer epidemiology*, 36(5):e271–e278, 2012.

[48] Yuchen Jiang, Shen Yin, Jingwei Dong, and Okyay Kaynak. A review on soft sensors for monitoring, control and optimization of industrial processes. *IEEE Sensors Journal*, 2020.

[49] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. j. big data 6 (1), 1–54 (2019).

[50] RJ Johnson. A comprehensive review of an electronic health record system soon to assume market ascendancy: Epic. *J Healthc Commun*, 1(4):36, 2016.

[51] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[52] Daniel Kahneman and Gary Klein. Conditions for intuitive expertise: a failure to disagree. *American psychologist*, 64(6):515, 2009.

[53] Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181, 2004.

[54] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[55] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[56] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[57] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[58] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[59] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[61] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[62] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[63] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[64] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[65] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.

[66] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[67] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Transl. Comput. Linguistics*, 11(1-2):22–31, 1968.

[68] Hans Peter Luhn. A business intelligence system. *IBM Journal of research and development*, 2(4):314–319, 1958.

[69] Xiaoyi Luo, Xianmin Li, Ziyang Wang, and Jun Liang. Discriminant autoencoder for feature extraction in fault diagnosis. *Chemometrics and Intelligent Laboratory Systems*, 192:103814, 2019.

[70] Melvin Earl Maron and John Larry Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960.

[71] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[72] Andrew McCallum, Ronald Rosenfeld, Tom M Mitchell, and Andrew Y Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, volume 98, pages 359–367. Citeseer, 1998.

[73] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[74] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[75] Randolph A Miller. Medical diagnostic decision support systems—past, present, and future: a threaded bibliography and brief commentary. *Journal of the American Medical Informatics Association*, 1(1):8–27, 1994.

[76] Randolph A Miller. Computer-assisted diagnostic decision support: history, challenges, and possible paths forward. *Advances in health sciences education*, 14(1):89–106, 2009.

[77] Tom Mitchell. *Machine learning*. McGraw hill Burr Ridge, 1997.

[78] Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, 2015.

[79] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4):354–359, 1983.

[80] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[81] Martine Nurek, Olga Kostopoulou, Brendan C Delaney, and Aneez Esmail. Reducing diagnostic errors in primary care. a systematic meta-review of computerized diagnostic decision support systems by the linneaus collaboration on patient safety in primary care. *European Journal of General Practice*, 21(sup1):8–13, 2015.

[82] Witold Pedrycz and Shyi-Ming Chen. *Development and Analysis of Deep Learning Architectures*. Springer, 2020.

[83] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[84] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[85] Xiaoye Qian, Haoyou Cheng, Diliang Chen, Quan Liu, Huan Chen, Haotian Jiang, and Ming-Chun Huang. The smart insole: A pilot study of fall detection. In *EAI International Conference on Body Area Networks*, pages 37–49. Springer, 2019.

[86] Xiaoye Qian, Huan Chen, Haotian Jiang, Justin Green, Haoyou Cheng, and Ming-Chun Huang. Wearable computing with distributed deep learning hierarchy: a study of fall detection. *IEEE Sensors Journal*, 20(16):9408–9416, 2020.

[87] Xiaoye Qian, Chao Zhang, Jaswanth Yella, Yu Huang, Sergei Petrov, Ming-Chun Huang, and Sthitie Bom. Soft sensing model visualization: Fine-tuning neural network from what model learned. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[88] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[89] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[90] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[91] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[92] Alaa Sagheer and Mostafa Kotb. Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems. *Scientific reports*, 9(1):1–16, 2019.

[93] Paul Sajda. Machine learning for detection and diagnosis of disease. *Annu. Rev. Biomed. Eng.*, 8:537–565, 2006.

[94] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[95] Shivani Sharma and Saurabh Kr. Review on Text Mining Algorithms. *International Journal of Computer Applications*, 134(8):39–43, 2016. doi: 10.5120/ijca2016907972.

[96] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666. IEEE, 2003.

[97] Hardeep Singh, Ashley ND Meyer, and Eric J Thomas. The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving us adult populations. *BMJ quality & safety*, 23(9):727–731, 2014.

[98] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[99] Francisco A.A. Souza, Rui Araújo, Tiago Matias, and Jérôme Mendes. A multilayer-perceptron based method for variable selection in soft sensor design. *Journal of Process Control*, 23(10):1371 – 1378, 2013. ISSN 0959-1524. doi: http://dx.doi.org/10.1016/j.jprocont.2013.09.014. URL `http://www.sciencedirect.com/science/article/pii/S0959152413001832`.

[100] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[101] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473, 2019.

[102] Qingqiang Sun and Zhiqiang Ge. Deep learning for industrial kpi prediction: When ensemble learning meets semi-supervised data. *IEEE Transactions on Industrial Informatics*, 17(1):260–269, 2020.

[103] Qingqiang Sun and Zhiqiang Ge. A survey on deep learning for data-driven soft sensors. *IEEE Transactions on Industrial Informatics*, 2021.

[104] Ah-Hwee Tan et al. Text mining: The state of the art and the challenges. In *Proceedings of the pakdd 1999 workshop on knowledge disocovery from advanced databases*, volume 8, pages 65–70. Citeseer, 1999.

[105] CJ van Rijsbergen. Information retrieval, 2nd edbutterworths, 1979.

[106] Emily Vardell and Mary Moore. Isabel, a clinical decision support system. *Medical reference services quarterly*, 30(2):158–166, 2011.

[107] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[108] Haishuai Wang, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallah, and Alexander Kronzer. Predicting hospital readmission via cost-sensitive deep learning. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(6): 1968–1978, 2018.

[109] Shengfeng Wang, Temidayo Ogundiran, Adeyinka Ademola, Oluwasola A Olayiwola, Adewunmi Adeoye, Adenike Sofoluwe, Imran Morhason-Bello, Stella Odedina, Imaria Agwai, Clement Adebamowo, et al. Development of a breast cancer risk prediction model for women in nigeria. *Cancer Epidemiology and Prevention Biomarkers*, 27(6): 636–643, 2018.

[110] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *2016 international joint conference on neural networks (IJCNN)*, pages 4368–4374. IEEE, 2016.

[111] Xiaoyan Wang, Amy Chused, Noémie Elhadad, Carol Friedman, and Marianthi Markatou. Automated knowledge acquisition from clinical narrative reports. In *AMIA Annual Symposium Proceedings*, volume 2008, page 783. American Medical Informatics Association, 2008.

[112] Homer R Warner, Alan F Toronto, L George Veasey, and Robert Stephenson. A mathematical approach to medical diagnosis: application to congenital heart disease. *Jama*, 177(3):177–183, 1961.

[113] Jonathan J Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*, 1992.

[114] Xuefeng Yan, Jie Wang, and Qingchao Jiang. Deep relevant representation learning for soft sensing. *Information Sciences*, 514:263–274, 2020.

[115] Le Yao and Zhiqiang Ge. Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application. *IEEE Transactions on Industrial Electronics*, 65(2):1490–1498, 2017.

[116] Jaswanth Yella, Chao Zhang, Yu Huang, Xiaoye Qian, Sergei Petrov, Ali Minai, and Sthitie Bom. Soft-sensing conformer: A curriculum learning-based convolutional transformer. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[117] Liyang Yu. *A developer's guide to the semantic Web*. Springer Science & Business Media, 2011.

[118] Xiaofeng Yuan, Biao Huang, Yalin Wang, Chunhua Yang, and Weihua Gui. Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted sae. *IEEE Transactions on Industrial Informatics*, 14(7):3235–3243, 2018.

[119] Xiaofeng Yuan, Jiao Zhou, Biao Huang, Yalin Wang, Chunhua Yang, and Weihua Gui. Hierarchical quality-relevant feature representation for soft sensor modeling: a novel deep learning strategy. *IEEE transactions on industrial informatics*, 16(6):3721–3730, 2019.

[120] Xiaofeng Yuan, Shuaibin Qi, Yuri AW Shardt, Yalin Wang, Chunhua Yang, and Weihua Gui. Soft sensor model for dynamic processes based on multichannel convolutional neural network. *Chemometrics and Intelligent Laboratory Systems*, 203:104050, 2020.

[121] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[122] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

[123] Chao Zhang and Sthitie Bom. Auto-encoder based model for high-dimensional imbalanced industrial data. In *International Conference on Neural Information Processing*, pages 265–273. Springer, 2021.

[124] Chao Zhang, Jaswanth Yella, Yu Huang, and Sthitie Bom. Learning from failures in large scale soft sensing. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 6067–6070. IEEE, 2021.

[125] Chao Zhang, Jaswanth Yella, Yu Huang, Xiaoye Qian, Sergei Petrov, Andrey Rzhetsky, and Sthitie Bom. Soft sensing transformer:hundreds of sensors are worth a single word. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[126] Chao Zhang, Jaswanth Yella, Yu Huang, Xiaoye Qian, Sergei Petrov, Andrey Rzhetsky, and Sthitie Bom. Soft sensing transformer:hundreds of sensors are worth a single word. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[127] Yu Zhang, Mengdong Chen, and Lianzhong Liu. A review on text mining. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2015-Novem:681–685, 2015. ISSN 23270594. doi: 10.1109/ICSESS.2015.7339149.

[128] Zehan Zhang, Teng Jiang, Shuanghong Li, and Yupu Yang. Automated feature learning for nonlinear process monitoring–an approach using stacked denoising autoencoder and k-nearest neighbor rule. *Journal of Process Control*, 64:49–61, 2018.