PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهـــوريـــــة الجزائـريـــــة الديـمقـراطيـــــة الشعبيـــــة
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليـم العالـــي والبحـــث العلمـي
HIGHER SCHOOL OF COMPUTER SCIENCE AND DIGITAL TECHNOLOGIES-BEJAIA
المـــدرسة العـــليا في علـــوم وتكنـولوجيـات الاعـــلام الالـــي بجـــاية

Dissertation Submitted to the Department Of Computer Science in Partial Fulfillment of the Requirements for Engineer's Degree in Computer Science

Specialty: Artificial Intelligence and Data Sciences

Submitted By:

Asma BOUROUBA

Narima ALKAMA

Text Mining for Predictive Maintenance

Case Study: Cevital agro-industry

Supervised by:

Dr. Meroua DAOUDI, ESTIN
Mr. Anis IDIR, Cevital

Members of jury:

▪ **Pr.** Hamid KHERBACHI	President	ESTIN
▪ **Dr.** Youssef ELMIR	Examiner	ESTIN
▪ **Mr.** Amine MAMMASSE	Examiner	ESTIN
▪ **Mr.** Amine BECHAR	Examiner	ESTIN

Academic year: 2023/2024

Abstract

Achieving operational excellence is vital for maintaining competitiveness in today's industrial landscape. Traditional maintenance strategies, both reactive and preventive, often fail to fully leverage the extensive data available. The advent of Industry 4.0, with its focus on data acquisition and analytics, has led to the development of predictive maintenance, enabling real-time insights into equipment health and proactive interventions. This thesis introduces an innovative approach to predictive maintenance by applying NLP techniques to textual maintenance logs. Our method utilizes BERTopic for embedding, K-means clustering for grouping intervention descriptions, and autoencoders for feature extraction to identify equipment degradation states. These states are then used to develop a Bayesian network-based predictive model. Applied to Cevital, an Algerian agri-food conglomerate using Coswin 8i, our approach aims to enhance maintenance planning and operational efficiency. The clustering model achieved a high silhouette score of 97%, and by integrating it with the Bayesian model, we attained an accuracy of 83%. This study highlights the potential of integrating advanced NLP and predictive analytics into maintenance management.

This work underscores the transformative potential of integrating cutting-edge NLP techniques and predictive analytics into traditional maintenance management practices. By harnessing textual data effectively, organizations can transition from reactive to proactive maintenance strategies, thereby minimizing downtime, reducing operational costs, and ultimately enhancing overall productivity and competitiveness in the industry.

keywords : Predictive Maintenance, Maintenance Logs, Artificial Intelligence (AI), Data Analytics, Natural Language Processing (NLP), Text Mining, Bayesian Network, Equipment Degradation.

Résumé

Atteindre l'excellence opérationnelle est crucial pour maintenir la compétitivité dans le paysage industriel actuel. Les stratégies de maintenance traditionnelles, qu'elles soient réactives ou préventives, ne parviennent souvent pas à exploiter pleinement les vastes données disponibles. L'avènement de l'industrie 4.0, avec son accent sur l'acquisition et l'analyse des données, a conduit au développement de la maintenance prédictive, permettant des informations en temps réel sur l'état des équipements et des interventions proactives. Cette thèse introduit une approche innovante de la maintenance prédictive en appliquant des techniques de NLP (traitement du langage naturel) aux journaux de maintenance textuels. Notre méthode utilise BERTopic pour l'embedding, le clustering K-means pour regrouper les descriptions d'intervention, et des autoencodeurs pour l'extraction de caractéristiques afin d'identifier les états de dégradation des équipements. Ces états sont ensuite utilisés pour développer un modèle prédictif basé sur un réseau bayésien. Appliquée à Cevital, un conglomérat agroalimentaire algérien utilisant Coswin 8i, notre approche vise à améliorer la planification de la maintenance et l'efficacité opérationnelle. Le modèle de clustering a atteint un score de silhouette élevé de 97 %, et en l'intégrant au modèle bayésien, nous avons atteint une précision de 83 %. Cette étude met en lumière le potentiel de l'intégration avancée du NLP et de l'analytique prédictive dans la gestion de la maintenance.

Ce travail souligne le potentiel transformateur de l'intégration des techniques avancées de NLP et de l'analytique prédictive dans les pratiques traditionnelles de gestion de la maintenance. En exploitant efficacement les données textuelles, les organisations peuvent passer de stratégies de maintenance réactives à proactives, minimisant ainsi les temps d'arrêt, réduisant les coûts opérationnels et améliorant finalement la productivité globale et la compétitivité dans l'industrie.

Mots clés : Maintenance Prédictive, Fichiers de Maintenance, Intelligence Artificielle (IA), Analyse de Données, Traitement du Langage Naturel (NLP), Exploration de Texte, Réseau Bayésien, Dégradation des Équipements.

Acknowledgements

First and foremost, we express our deepest gratitude to Allah for granting us the strength and perseverance throughout this endeavor.

We would like to extend our heartfelt appreciation to our dedicated supervisor, Mrs DAOUDI Maroua, for her invaluable guidance, encouragement, and unwavering support throughout the course of this project.

Special thanks are also due to Mr IDIR Anis from Cevital, for providing us with the opportunity to gain practical experience and for his mentorship during our time there.

We are grateful to Mr KHERBACHI Hamid and Mrs KHERBACHI Sonia for their generous assistance, insightful guidance, and invaluable feedback that greatly enhanced the quality of our work.

Lastly, we would like to acknowledge the judges Pr KHERBACHI Hamid, Dr ELMIR Youssef, Mr MAMMASSE Amine and Mr BECHAR Amine who generously accepted to be part of the committee and dedicated their time and expertise to evaluate our project.

Dedications

In the name of God, the most gracious, the most merciful

I dedicate this work,

To my dear parents

for their unwavering support, presence, and abundant love. Their sacrifices and encouragement have been the foundation of my journey and i pray Allah to reward them in this world and the last,

To my lovely siblings

Soumaya and Oussama for bringing humor and lightening my days,

To my best friends Meriem and Rezkia

that i thank god for blessing me with thier presence in my life,

To my friend and project partner Narima

for her support, dedication that have been invaluable throughout our journey. I am so lucky to have a best friend and a partner like her, funny, hardworking, patient, and always bringing positivity to everything we do together.

From the heart, your support and collaboration have been invaluable and deeply appreciated.

Asma

Dedications

I dedicate this modest work,

To my mother,

Your unwavering love, patience, and sacrifice have shaped me into who I am today. This thesis is a testament to your endless love and dedication. Thank you for being the heart and soul of my journey.

To my father,

You have been my pillar of strength, teaching me the value of hard work, discipline, and perseverance. This thesis is a reflection of your faith in me and your endless support. Thank you for being my rock and guiding light.

To my siblings,

The branches of my life. You are the shining examples I strive to emulate in my life. Your unwavering support and belief in me have been a constant source of motivation.

To my best friend Meriem,

With whom I have shared a friendship spanning over a decade. Your presence has been an unspoken source of emotional support.

To my partner and friend Asma,

Whose collaboration has been instrumental in our academic journey, your dedication, humor, understanding, expertise, and positivity throughout this work, as well as during our years of study together, have made this endeavor both productive and meaningful.

To all my family, my friends and all those who have helped me and contributed to near or far to carry out this work.

Narima

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area under the Curve
BERT	Bidirectional Encoder Representations from Transformers
BI	Business Intelligence
BIC	Bayesian Information Criterion
BIS	Business intelligence systems
BoW	Bag of Words
BN	Bayesian Network
BPTT	Backpropagation Through Time
CBM	Condition-based maintenance
CMMS	Computerized Maintenance Management Systems
CNN	Convolutional Neural Networks
CPD	Conditional Probability Distributions
CTMC	Continuous-Time Markov Chain
DAG	Directed Acyclic Graph
EDA	Exploratory Data Analysis
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMM	Hidden Markov Model
HMM-AO	Hidden Markov Model with Auto-Correlated Observations
IDF	Inverse Document Frequency

IoT	Internet of Things
IoU	Intersection over Union
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
LSTM-Bayes	Long Short-Term Memory Bayesian
LSTM-RNN	Long Short-Term Memory Recurrent Neural Network
MAE	Mean Absolute Error
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MN	Markov Network
MSE	Mean Squared Error
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PCA	Principal Component Analysis
PGM	Probabilistic Graphical Model
PGMPY	Probabilistic Graphical Models in Python
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
SBERT	Sentence-BERT
SQL	Structured Query Language
SSMS	SQL Server Management Studio
T-SNE	t-distributed Stochastic Neighbor Embedding
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TLP	Technical Language Processing
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
UMAP	Uniform Manifold Approximation and Projection

General Introduction

In an ever-evolving industrial environment, effective maintenance management is of paramount importance. In this context, the optimal use of maintenance data represents a major challenge for companies wishing to ensure the reliability and availability of their equipment while controlling the associated costs. Traditional approaches to maintenance planning often rely on reactive or preventive strategies, which, while effective to a certain extent, may fall short in fully leveraging the potential of available data to predict and preempt equipment failures. As industries strive to optimize their operations and meet increasingly demanding market requirements, there is a growing recognition of the need to adopt innovative strategies to enhance maintenance practices.

Cevital is an Algerian conglomerate in the agri-food industry that aims to implement AI and utilize its data in order to extract actionable insights to help stakeholders better plan interventions. Using Coswin 8i, the computer-assisted maintenance management software for planning interventions proactively which is an expensive and less efficient method for maintenance management; The question is, how to use the data we have to improve intervention planning and higher the data maturity of the enterprise.

Leveraging maintenance data extracted from Coswin 8i, stakeholders at Cevital seek to elevate the maturity of their decision-making processes. By integrating data and AI, predictive maintenance emerges as the preferred strategy for optimizing intervention planning. This approach has demonstrably minimized machine downtime, leading to increased machine availability, maximized lifespan, and reduced labor costs.

Following a comprehensive exploration of the provided data, we identified a lack of features suitable for conventional machine learning models that rely on real-time sensor data (e.g., temperature, vibration, rotation) or historical machine data. Consequently, we pursued alternative predictive strategies applicable to the available data format. This led us to explore

the utilization of maintenance logs files, which are textual data, for machine failure prediction.

This work explores the realm of predictive maintenance, specifically focusing on the use of text mining and NLP techniques on textual data coming from maintenance log files. The methodology employed in this study involves the identification of equipment degradation states through the clustering of maintenance records. To achieve this, we utilize the BERTopic algorithm for embedding, followed by K-means clustering for grouping similar interventions descriptions. Additionally, we employ an autoencoder for feature extraction to enhance the clustering process. The resulting clusters represent different degradation states of the equipment. Subsequently, we develop a stochastic multi-state degradation model based on these clusters. This methodology is applied to a database comprising maintenance records, labor costs, and dates extracted from the CMMS known as Coswin 8i. The research endeavors to bridge the gap between theoretical knowledge and practical application of predictive maintenance in an industrial setting. The ultimate goal of applying the adapted approach is to optimize the overall efficiency of maintenance operations and enhance the intervention planning process.

The remainder of this dissertation is structured as follows:

The first chapter, explores core concepts and techniques for predictive maintenance, including some Business intelligence concepts, the evolution of industrial maintenance, various strategies, key machine learning concepts, and NLP techniques for maintenance data analysis.

The second chapter, presents a novel approach to analyzing textual data using BERTopic for topic modeling and autoencoders for feature extraction, integrating identified equipment states into a Bayesian network for prediction and decision-making.

The third chapter, details the methodologies tested for predictive maintenance, covering experimental approaches, feature extraction and topic modeling techniques, graphical models, and results.

In Conclusion, the dissertation concludes with a summary of the research, highlighting the impact of predictive maintenance, discussing limitations, and providing future research recommendations.

Chapter1

General Concepts and Related Works

Contents

In the fast-paced industrial landscape of today, maintaining operational efficiency and minimizing downtime are essential for staying competitive. Predictive maintenance has become a crucial strategy to prevent equipment failures and optimize maintenance activities. This chapter delves into the core concepts and techniques necessary for implementing predictive maintenance systems, with a focus on their application in contemporary industries.

We start by exploring the evolution of industrial maintenance and its importance in business intelligence. Various types of industrial maintenance strategies are discussed to provide a well-rounded understanding of their functions and advantages. Following this, we introduce key machine learning concepts such as K-means clustering, Bayesian networks, and Markov models, which underpin predictive maintenance algorithms. Additionally, we cover evaluation metrics that are vital for measuring the performance and reliability of these models.

The chapter also examines Natural Language Processing (NLP) techniques, including text preprocessing, feature extraction, and topic modeling, emphasizing their role in extracting valuable insights from maintenance logs and related textual data. We conclude with a review of the latest methods and technologies used in predictive maintenance across different industries.

1.1 Business Intelligence

Business Intelligence (BI) involves a set of concepts, methods, and processes aimed at enhancing business decisions by using information from multiple sources to develop a precise understanding of business dynamics [1]. According to reference , BI can be defined as a system comprising both technical and organizational elements that provide users with historical information and analysis, enabling effective decision-making and management support with the overall goal of improving organizational performance.

1.1.1 Business Intelligence Systems

Business intelligence systems (BIS) have emerged as technological solutions that offer data integration and key performance indicators to provide stakeholders at various organizational levels with valuable information for their decision-making [1]. The main tasks of a BI system include intelligent exploration, integration, aggregation, and multidimensional analysis of data gathered from various sources, thereby transforming data from quantity to quality [2].

1.1.2 Main Objective of Business Intelligence in Companies

The primary objective of BI is to help business users convert business-related data into actionable knowledge. Traditionally, BI focused on generating reports, dashboards, and answering predefined questions [3]. However, modern BI now incorporates deep, exploratory, and interactive data studies through Business Analytics, which includes data mining, predictive analytics, statistical analysis, and natural language processing tools. These components are crucial for providing comprehensive insights and supporting strategic decision-making .

1.1.3 Evolution of industries

The evolution of industries can be traced through four distinct phases. **Industry 1.0** marks the advent of industrialization, characterized by the introduction of steam power and machines, which began replacing manual labor, leading to significant advancements in textiles, iron, coal production, and transportation networks. **Industry 2.0** represents the second phase, where the rise of electricity and further mechanization spurred substantial growth in the automobile and chemical industries, and introduced transformative technologies like the telephone and radio. **Industry 3.0** highlights the third phase, defined by the emergence of

computers and the automation of knowledge-based tasks, with major advancements in information technology, the expansion of the service industry, and innovations such as the internet and artificial intelligence. Finally, **Industry 4.0** symbolizes the current phase, characterized by the convergence of physical, digital, and biological spheres, marked by the proliferation of the Internet of Things (IoT), growth in the biotechnology sector, and the development of advanced technologies like quantum computing and nanotechnology.

1.2 Industrial Maintenance

Industrial maintenance is vital in manufacturing, significantly reducing machine downtime and costs, particularly in the context of Industry 4.0 [4]. The integration of advanced technologies such as IoT sensors, predictive analytics, and automation has made maintenance processes more proactive and efficient. Predictive maintenance enables early detection of potential equipment failures, allowing for timely interventions to avoid costly breakdowns and unplanned downtime. This proactive approach enhances overall equipment effectiveness and ensures smoother production, aligning with Industry 4.0's goals of increased efficiency and competitiveness. Key aspects include preventive measures, predictive techniques, corrective actions, routine tasks, emergency repairs, asset management, and safety measures.

1.2.1 Types of Industrial Maintenance

1.2.1.1 Preventive Maintenance

Preventive maintenance involves scheduling regular maintenance to prevent equipment deterioration and extend its lifespan, ensuring continuous production [5]. The challenge lies in determining the optimal timing, as premature maintenance can waste resources, while delayed maintenance can compromise safety and performance [6].

1.2.1.2 Predictive Maintenance

Predictive maintenance uses data analysis to forecast potential machine breakdowns, addressing issues before significant wear occurs. Unlike preventive maintenance, it relies on the current condition of the machine to detect maintenance needs [7]. This approach optimizes maintenance schedules, reduces unnecessary downtime, and improves resource allocation, enhancing operational efficiency and cost-effectiveness.

1.2.1.3 Corrective Maintenance

Corrective maintenance identifies and corrects the causes of system failures, focusing on resolving issues that cause breakdowns [8]. While crucial for addressing unforeseen problems promptly to minimize production interruptions, relying solely on corrective maintenance can lead to higher costs due to unscheduled downtime, increased safety risks, and potential equipment damage. Effective corrective maintenance includes root cause analysis to prevent recurrent issues.

1.2.1.4 Condition-Based Maintenance

Condition-based maintenance (CBM) [8] is a proactive form of preventive maintenance based on the actual condition of equipment. It involves monitoring equipment to decide when maintenance is necessary, using data on the equipment's condition, such as age, usage, or performance. CBM aims to perform maintenance before failures occur, optimizing maintenance efforts and preventing unexpected breakdowns by conducting maintenance based on real-time equipment conditions.

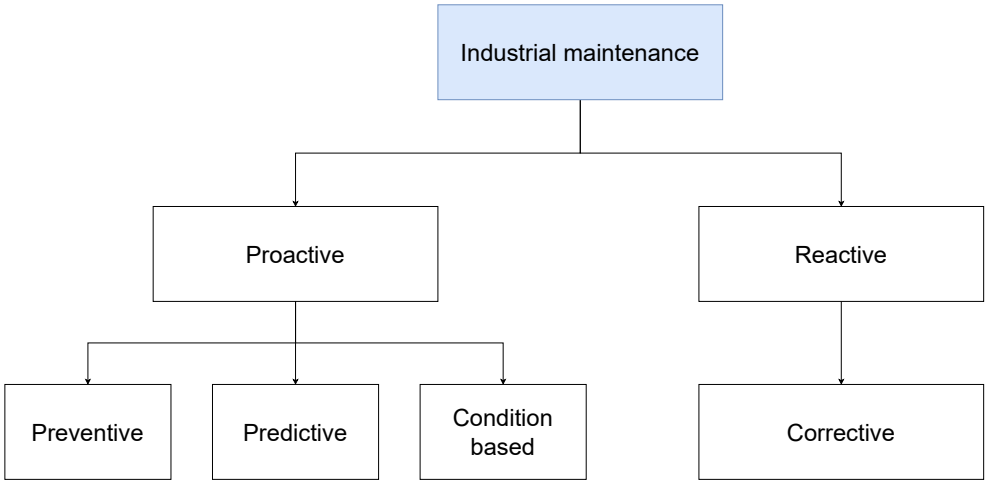


Figure 1.1: Industrial Maintenance's types

1.3 Machine Learning concepts

This section introduces essential machine learning concepts integral to the proposed approach. We begin with K-means Clustering, an algorithm for partitioning datasets into clusters based on feature similarity. Next, we discuss Bayesian Networks, detailing their structure, inference techniques, and the processes for both parameter and structure learning. We then

cover Markov Models, focusing on Markov Chains and Hidden Markov Models, which are pivotal for sequential data modeling. Finally, we explore various Evaluation Metrics, including accuracy, precision, recall, F1 score, and silhouette score, which are crucial for assessing model performance. Each concept is thoroughly examined to provide a foundational understanding necessary for the application in our approach.

1.3.1 K-means Clustering

The k-means clustering algorithm is a widely utilized method in unsupervised learning, particularly for partitioning data into clusters of similar variance. Its primary objective is to minimize inertia, also referred to as within-cluster sum-of-squares, by assigning data points to centroids iteratively. A drawback of this method, common to centroid-based clustering, is the necessity to predefine the number of clusters (k). Despite this limitation, its popularity stems from its simplicity and scalability with large datasets [9].

Formally, the algorithm 1 [10] segregates a dataset with X data points into K disjoint clusters C_k, each described by cluster centroids μ_k, not restricted to being actual data points. The objective is to minimize inertia, represented mathematically as:

$$\min_{\mathbf{C},\mu} \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

Algorithm 1 K-means Algorithm

Require: **K** (number of clusters), X (data points), max_iterations
1: Initialize **K** cluster centroids randomly
2: iterations $\leftarrow 0$
3: **while** iterations $<$ max_iterations **and** changes in centroids **do**
4: Assign each data point to the nearest centroid
5: **for** $k = 1$ **to K do**
6: Calculate the mean of data points assigned to cluster k
7: Update centroid k to the calculated mean
8: **end for**
9: iterations $\leftarrow$ iterations $+ 1$
10: **end while**
11: **return** Final cluster assignments, cluster centroids

1.3.2 Dimensionality reduction techniques

1.3.2.1 Autoencoder

Autoencoders are artificial neural networks designed to learn dense, efficient representations of input data, known as latent representations or codings, without requiring labeled training data. These codings typically have a significantly lower dimensionality than the input data, making autoencoders particularly valuable for dimensionality reduction and visualization purposes. In the context of text data, autoencoders can extract meaningful features by compressing the information into a lower-dimensional space and then reconstructing the original text data from this compact representation.

This process not only aids in reducing data complexity but also helps in identifying underlying patterns and structures within the text. Additionally, autoencoders serve as powerful feature detectors, facilitating unsupervised pre-training of deep neural networks, which can enhance the performance of subsequent supervised learning tasks by providing a rich set of learned features.

An autoencoder consists of two primary components: the encoder, which translates the input into a compressed code, and the decoder, which reconstructs the original input from this code. The goal of an effective autoencoder is to achieve high-quality reconstruction, meaning the output should be as similar as possible to the input, as measured by a predefined reconstruction quality function [11].

- **Encoder :** The encoder part of the autoencoder compresses the input data into a lower-dimensional representation. For text, this involves converting the input text into dense vectors and then transforming these vectors into a compressed form.

  - **Embedding Layer :** Converts words into dense vectors:

    $$Embedding(word) \rightarrow DenseVector$$

  - **Dense Layer :** Apply a series of transformations to compress the data:

    $$h = \sigma(Wx + b)$$

    Where h is the hidden (encoded) represenation, W is the weight matrix, x is the input, b is the bias, and σ is the activation function.

- **Bottleneck :** The bottleneck layer is the smallest layer in the network, representing the compressed version of the input. This latent space captures the most critical features of the text data.

- **Decoder :** The decoder reconstructs the original input from the compressed representation. It mirrors the encoder's structure but in reverse, expanding the compressed data back to its original dimensions.

  - **Dense Layer :** Transform the encoded data back to the original dimensions:

$$x' = \sigma(W'h + b')$$

    Where x' is the reconstructed output, W' is the weight matrix, h is the hidden representation, b is the bias, and σ is the activation function.

- **Reconstruction Loss :** The autoencoder is trained to minimize the difference between the input and the reconstructed output. This difference is quantified using a reconstruction loss function, such as MSE :

$$Loss = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_i')^2$$

  Once trained, the encoder part of the autoencoder is used to extract features from new text data. These features are the compressed representations learned during training, which capture the most significant aspects of the input data.

1.3.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that transforms datasets into a lower-dimensional space while retaining significant information. It identifies directions of maximum variance, known as principal components, and projects the data onto them. This reduces dimensional complexity and preserves essential information. PCA involves standardizing the data, computing the covariance matrix, finding and sorting eigenvectors and eigenvalues, selecting the top k eigenvectors, and projecting the data onto the new coordinate system. This results in a simplified dataset that enhances machine learning model performance and data interpretation [12, 13].

1.3.2.3 T-distributed Stochastic Neighbor Embedding

T-distributed Stochastic Neighbor Embedding (T-SNE) is widely used for dimensionality reduction. This method maps high-dimensional data into a lower-dimensional space, facilitating visualization by preserving the local structure of the data. It achieves this by minimizing the divergence between two probability distributions: one that measures pairwise similarities in the original high-dimensional space and another in the lower-dimensional embedding. This process helps to maintain the meaningful patterns and clusters within the data while reducing its complexity, thus making it more interpretable [14].

1.3.2.4 Uniform Manifold Approximation and Projection

UMAP is a dimensionality reduction technique that preserves meaningful relationships between data points. It is commonly used, like t-SNE, to visualize complex datasets in lower dimensions while maintaining the inherent structure of the data. Unlike linear methods such as PCA, UMAP assumes data exists on a manifold and seeks to find a compact representation that captures both local and global relationships. This approach makes UMAP versatile for tasks in machine learning, bioinformatics, and natural language processing, facilitating tasks from data exploration to feature extraction and visualization [15].

1.3.3 Bayesian network

Bayesian Networks (BNs), also known as Belief Networks or Bayes Nets, are probabilistic graphical models designed to represent knowledge in uncertain domains. They effectively model conditional dependencies among a set of random variables through a structured framework [9, 16, 17, 18].

1.3.3.1 Structure of a Bayesian Network

A Bayesian Network consists of nodes and directed arcs. Each node represents a random variable X, associated with a probability $P(X)$. A directed arc from node X_i to node X_j indicates that X_i directly influences X_j, quantified by the conditional probability $P(X_j|X_i)$. This structure forms a Directed Acyclic Graph (DAG), ensuring no cycles exist within the network. The joint probability distribution over all variables $X_1, X_2, \ldots, X_n$ is expressed using the chain rule of probability, considering conditional dependencies:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{Pa}(X_i))$$

where $\text{Pa}(X_i)$ denotes the set of parent nodes of X_i.

1.3.3.2 Inference in Bayesian Networks

Inference in BNs involves calculating the probability of certain variables given known values of others. For a set of variables X with evidence E, the posterior probability is:

$$P(X|E) = \frac{P(X, E)}{P(E)}$$

Common techniques for inference include:

- **Variable Elimination**: This method sums out irrelevant variables from the joint distribution to compute the probability of the query variables given the evidence.
- **Belief Propagation**: Used for exact inference in tree-structured networks, this technique updates beliefs about variable distributions by passing messages between nodes.

1.3.3.3 Parameter Learning in Bayesian Networks

Parameter learning in Bayesian Networks involves estimating the conditional probability distributions (CPDs) from data. This can be achieved through methods like Maximum Likelihood Estimation (MLE) and Bayesian estimation.

Maximum Likelihood Estimation : This method aims to find parameters that maximize the likelihood of the observed data given the model. For a dataset D with N instances, the MLE for parameters of a CPD $P(X_i|\text{Pa}(X_i))$ is:

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} P(x_i|\text{Pa}(x_i); \theta)$$

1.3.3.4 Structure Learning in Bayesian Networks

Structure learning involves discovering the optimal graph structure from data, which can be approached through:

- **Constraint-Based Methods**: These methods infer dependency conditions from the data and construct the network accordingly. A major limitation is the potential for failure if dependence tests are inaccurate.

- **Score-Based Methods**: These treat the Bayesian Network as a statistical model, evaluating different structures using a scoring function (e.g., Bayesian Information Criterion, BIC):

$$\text{BIC}(G|D) = \log P(D|G) - \frac{\log N}{2} \cdot |G|$$

  Heuristic search techniques are often employed due to the large hypothesis space.

- **Bayesian Model Averaging**: This approach involves learning multiple structures and using an ensemble of these structures. Approximate methods may be necessary due to the vast number of potential network structures.

In real-world applications, Bayesian Networks can manage numerous random variables by leveraging their conditional independencies, reducing the computational burden and memory requirements for representing joint probability distributions. Despite their complexity, BNs provide a powerful tool for modeling uncertain domains and making informed predictions based on observed data.

To construct a BN from a given set of random variables, we follow a structured algorithm as given in algorithm 2.

Algorithm 2 Build Bayesian Network

Input: Set of random variables $V = \{X_1, X_2, \ldots, X_n\}$
Output: Bayesian Network BN
procedure BUILDBN(V)
    Initialize BN with nodes for each variable in V
    for each pair of variables (X_i, X_j) **do**
        Determine if there is a direct dependency between X_i and X_j
        if there is a dependency **then**
            Create a directed arc from X_i to X_j
        end if
    end for
    while BN has cycles **do**
        Remove or adjust arcs to eliminate cycles
    end while
    for each node X_i in BN **do**
        Assign the Conditional Probability Distribution (CPD) $P(X_i|\text{Pa}(X_i))$
    end for
    return BN
end procedure

1.3.4 Markov Models

Markov Models are stochastic models that describe sequences of possible events, where the probability of each event depends solely on the state achieved in the preceding event. These models are particularly valuable for representing time-dependent phenomena.

In graphical models, Markov chains are represented as graphs where the nodes denote states of variables X, and edges represent the transition probabilities between states [16].

1.3.4.1 Markov Chains

A Markov chain is a sequence of random variables $X_1, X_2, \ldots$ that possess the Markov property, indicating that the next state depends only on the current state and the transition probability, independent of prior states:

$$P(X_{k+1} = s \mid X_k = s_k, X_{k-1} = s_{k-1}, \ldots, X_1 = s_1) = P(X_{k+1} = s \mid X_k = s_k)$$

As k approaches infinity, such chains can achieve stable probability distributions that become independent of their initial states. Markov chains are critical for estimating the states of random parameters, relying on current values and transition probabilities [16, 18].

The algorithm 3 describes the process of simulating a Markov chain, which is a sequence of states generated based on given initial conditions and transition probabilities. The algorithm takes as input an initial state, a transition probability matrix, and the number of steps to simulate. The output is a sequence of states representing the Markov chain's evolution.

Algorithm 3 Simulate Markov Chain

Input: Initial state X_0, transition probability matrix P, number of steps N
Output: Sequence of states $X_0, X_1, \ldots, X_N$
procedure SIMULATEMARKOVCHAIN(X_0, P, N)
    Initialize $X \leftarrow [X_0]$
    for $k = 0$ to $N - 1$ **do**
        Current state $s \leftarrow X[k]$
        Sample next state s' from the distribution $P(s, \cdot)$
        Append s' to X
    end for
    return X
end procedure

1.3.4.2 Parameter Estimation in Markov Models

Learning parameters and structures in Markov networks is a complex task due to the partition function in the probability distribution. This function's dependence on all factors prevents the decomposition of optimization functions into separate terms, as seen in Bayesian networks. Consequently, iterative methods must be employed to optimize the parameter space.

Similar to Bayesian networks, parameters in Markov networks can be estimated using maximum likelihood estimation. This process involves iteratively optimizing the function to identify the optimal parameters, accommodating the intricacies introduced by the partition function [18].

1.3.4.3 Hidden Markov models

Hidden Markov models (HMMs) extend Markov Chains by incorporating hidden states that generate observable events, making them powerful tools for modeling sequences where the underlying system is not directly observable. Given a sequence of observations, the goal is to infer the underlying dynamical system that produced the sequence, resulting in a model of the process [13].

1.3.5 Evaluation metrics

Machine learning hinges on quantitative evaluation metrics to assess model performance and compare their effectiveness across diverse tasks. Classification, regression, image segmentation, and object detection are just a few examples. These metrics empower researchers and practitioners to gain valuable insights into model strengths and weaknesses. A diverse set of metrics, including accuracy, precision, recall, F1 score, AUC, MSE, MAE, and IoU, provide quantitative assessments of model performance. This wealth of information ultimately fuels informed decision-making and facilitates targeted improvements within machine learning projects [19, 20].

1.3.5.1 Accuracy

Accuracy is one of the most widely used measures of classifier performance. It is defined as the overall proportion of correct predictions made by the model. The formula for computing

accuracy from is:

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \in [0, 1]$$

1.3.5.2 Precision

Precision, also known as positive predictive value, measures the proportion of positive predictions that are actually correct. It is defined as the number of true positive predictions divided by the total number of positive predictions. The formula for precision is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \in [0, 1]$$

1.3.5.3 Recall

Recall, also known as sensitivity, measures the proportion of actual positive cases that were correctly identified by the model. It is defined as the number of true positive predictions divided by the total number of actual positive cases. The formula for recall is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \in [0, 1]$$

1.3.5.4 F1 Score

The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both precision and recall. This is particularly useful in cases where we want to optimize both metrics simultaneously. The formula for the F1 score is:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \in [0, 1]$$

To understand these metrics, it is important to understand the four main terminologies : TP, TN, FP and FN. Here is the precise definition of each of these terms :

- **TP (True Positive)**: Cases where the prediction is positive and the actual value is positive.
- **TN (True Negative)**: Cases where the prediction is negative and the actual value is negative.
- **FP (False Positive)**: Cases where the prediction is positive, but the actual value is negative.
- **FN (False Negative)**: Cases where the prediction is negative, but the actual value is positive.

1.3.5.5 Silhouette score

The Silhouette Index [21] is a measure used to evaluate the quality of clusters in a dataset. It assesses how well an object is clustered by comparing its similarity to its own cluster (cohesion) with its similarity to the nearest neighboring cluster (separation).The Silhouette index presents a significant advantage for evaluating the performance of clustering algorithms compared to many other methods. Unlike metrics commonly employed in classification tasks, the Silhouette index does not require a separate training set to assess the quality of the clustering solution. This characteristic makes it particularly well-suited for evaluating clustering results, where a designated training set might not be readily available or applicable. The silhouette score $s(i)$ for an individual data point i calculated as follows :

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

where :

- $a(i)$ is the average distance between i and all other points in the same cluster.
- $b(i)$ is the minimum average distance between i and all points in any other cluster, of which i is not a member.

The silhouette score ranges from -1 to 1 :

- A score close to 1 indicates that the object is well clusteres.
- A score close to 0 suggests that the object is on or very close to the decision boundary between two neighboring clusters.
- A negative score indicates that the object might have been assigned to the wrong cluster.

For an entire clustering solution, the average silhouette score over all data points is computed to provide an overall assessment of the clustering quality. A higher average silhouette score suggests a better-defined clustering structure.

1.4 Natural Language Processing techniques

Natural Language Processing (NLP) has seen significant advancements, enabling machines to understand and generate human language. These techniques are fundamental in numerous applications, from sentiment analysis and translation to more complex tasks like text clustering and summarization.

This section outlines the NLP techniques employed in our research. We start with the crucial preprocessing steps necessary to clean and prepare textual data for analysis. This includes tokenization, stemming, lemmatization, and the removal of noise, such as stopwords and punctuation.

Next, we cover advanced methods for converting text into numerical representations that can be processed by machine learning algorithms. We focus on embedding models, particularly SBERT, BERTopic, TF-IDF, and CNN, which effectively capture term frequency or semantic meaning within text data.

We then discuss dimensionality reduction techniques, such as PCA and autoencoders, used to simplify high-dimensional embedding spaces. This step is essential for improving the efficiency and performance of subsequent clustering algorithms.

1.4.1 Text preprocessing

Text mining heavily relies on preprocessing techniques to enhance the quality of data and facilitate subsequent analysis. The following preprocessing techniques are commonly employed [22, 23].

1.4.1.1 Tokenization

In computer processing, bodies of text are treated as single string objects, regardless of punctuation. To enable the computer to analyze each word individually, we need to segment this unified text into distinct tokens, representing words or characters for further evaluation. This process, known as word tokenization, is fundamental in NLP and serves as a type of document segmentation [24].

As the initial step in an NLP pipeline, tokenization significantly influences subsequent processing stages. A tokenizer divides unstructured natural language text into discrete elements, facilitating counting and analysis of token occurrences within a document. These token counts can be directly utilized as numerical representations of the text, suitable for machine learning applications. They serve as features in machine learning pipelines, triggering various actions, responses, or complex decision-making processes [25].

1.4.1.2 Stop word removal

Stop words are frequently occurring words in a language (e.g., "the," "a," "and") that

often carry little semantic meaning. Removing stop words helps reduce noise in the data and improves the efficiency of subsequent analysis [25].

1.4.1.3 Text normalization

Text normalization aims to standardize the text data to reduce the number of distinct tokens by combining similar-meaning tokens into a single, normalized form, which can improve the performance of NLP models. Key techniques for text normalization include [25]:

- **Case Normalization :** Converting all characters in the text to a uniform case, typically lowercase, to ensure that words are treated equally regardless of their original casing.
- **Stemming :** involves reducing words to their base or root form by removing suffixes. This technique groups words with similar meanings under a common stem, although the resulting stems may not always be valid words.
- **Lemmatization :** reduces words to their base or dictionary form, known as a lemma, considering the word's meaning. Unlike stemming, lemmatization uses a knowledge base to ensure that only words with similar meanings are consolidated into a single token. This process is more accurate than stemming and is preferable for most applications.

1.4.1.4 Preprocessing considerations for text clustering

The effectiveness of various text clustering techniques hinges on the selection and implementation of appropriate preprocessing methods. Here, we explore the rationale behind these preprocessing choices for two distinct clustering approaches:

- **Word Frequency-based preprocessing :** Preprocessing here typically involves removing stop words, numbers, and punctuation. These elements contribute minimal semantic content and inflate the sparsity of the resulting document-term matrix, hindering clustering performance. Additionally, correcting misspelled words and applying techniques like lemmatization or stemming can further reduce the vocabulary size employed in the sparse matrix, mitigating sparsity issues.

  This preprocessing method is found to be more suitable for word frequency-based clustering techniques because it reduces the corpus and removes useless words for more efficiency and helps avoid the sparsity problem.

- **Semantic-Based preprocessing :** This approach emphasizes capturing the underlying semantic meaning of words within the text data. Unlike methods based on word frequency, this approach may preserve stop words and certain levels of morphological variation (such as stemming or lemmatization) during preprocessing. The removal of stop words or alteration of word forms can potentially modify the semantic meaning of sentences. For instance, the sentence "non démarrage du convoyeur" loses crucial information if it is transformed into "démarrage convoyeur" through the removal of stop words.

  Using this preprocessing method, the semantics of the descriptions are preserved, allowing semantic-based clustering models to extract meaningful insights from them.

Therefore, the selection of preprocessing techniques for text clustering should be guided by the specific clustering approach employed. Tailoring preprocessing steps to the chosen method optimizes the performance of the clustering task.

1.4.2 Feature extraction techniques

Effective text representation is critical in clustering tasks and feature extraction, as it transforms raw textual data into meaningful numerical vectors that capture the semantic essence of the text. Feature extraction techniques play a vital role in this process by converting text into a structured format suitable for analysis. These representations are essential for accurately grouping similar texts together, enabling more precise and insightful clustering outcomes. By leveraging advanced feature extraction methods, we can enhance the quality of text clustering, leading to better interpretation and utilization of textual data.

1.4.2.1 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) builds on the BoW model by providing a more nuanced measure of a word's importance in a document relative to its prevalence across all documents. The TF-IDF value helps to highlight words that are significant in specific documents while downweighting common words that are less informative. It comprises two main elements [12, 24, 25, 26]:

- **Term Frequency (TF) :** TF measures how often a term or word appears in a given document. To normalize these counts, especially since different documents in the corpus

may vary in length, the number of occurrences is divided by the total number of terms in the document. The term frequency TF(t,d) of a term t in a document d is defined as :

$$TF(t,d) = \frac{\text{Number of occurances of term t in document d}}{\text{Total number of terms in document d}}$$

- **Inverse Document Frequency (IDF) :** IDF measures the importance of a term across the entire corpus. While TF assigns equal weight to all terms, common terms (e.g., "is," "are," "am") are not as informative. IDF addresses this by weighing down frequently occurring terms and emphasizing rare terms. The inverse document frequency IDF(t) of a term t is calculated as :

$$IDF(t) = \log \frac{\text{Number of documents in the corpus}}{\text{Number of documents containing term t}}$$

The TF-IDF score for a term in a document is computed by multiplying its TF by its IDF, offering a nuanced understanding of its significance.

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

TF-IDF is instrumental in feature extraction, converting unprocessed text data into numerical feature vectors. These vectors are pivotal for computational analysis and the comparison of documents based on their content. Furthermore, in text clustering, TF-IDF contributes to feature extraction from documents, supporting clustering algorithms in grouping similar documents together. This greatly expedites tasks such as document organization and topic identification. The versatility of TF-IDF spans across various text analysis tasks, facilitating the seamless transformation of raw textual data into structured numerical representations, all while mitigating concerns of originality.

1.4.2.2 Convolutional Neural Network

In the realm of NLP, Convolutional Neural Networks (CNNs) emerge as formidable allies, specifically tailored for feature extraction tasks. Here's an overview of how CNNs Components and how they operate in this context [27]:

- **Convolutional Layers :** These layers are pivotal in extracting hierarchical features from textual data. By applying convolution operations using learnable filters or kernels, CNNs adeptly capture patterns and features within the text, discerning linguistic nuances at various scales.

- **Convolution Operation :** The convolution operation in CNNs is defined as:

$$(f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a)$$

where f represents the input text and g denotes the learnable filter or kernel.

- **Activation Function :** To introduce non-linearity and enhance the model's ability to learn intricate linguistic patterns, CNNs typically employ activation functions like ReLU (Rectified Linear Unit).

  - **ReLU Activation :** The Rectified Linear Unit (ReLU) activation function is defined as:

$$ReLU = max(0, x)$$

- **Pooling Layers :** Pooling layers play a crucial role in downsampling the feature maps obtained from the convolutional layers. By reducing the spatial dimensions while retaining essential textual features, pooling layers facilitate the extraction of salient linguistic information.

  - **Max Pooling :** Max pooling extracts the maximum value from each region of the feature map, downsampling the spatial dimensions. It is defined as:

$$MaxPooling(x) = max(x)$$

- **Fully Connected Layers :** These layers establish connections between neurons across different layers of the network, enabling the CNN to discern complex linguistic patterns and relationships. Through fully connected layers, the network gains the ability to extract and comprehend intricate linguistic structures embedded within the text.

  - **Fully Connected Layer Operation :** The operation in a fully connected layer can be represented as:

$$y = f(Wx + b)$$

where c represents the input, W denotes the weight matrix, b is the bias, and f is the activation function.

- **Feature Extraction :** As the input text traverses through the convolutional and pooling layers, the CNN meticulously extracts hierarchical linguistic features. Lower layers

specialize in capturing basic linguistic elements such as word sequences and syntactic structures, while deeper layers ascend to discerning more abstract and contextually rich linguistic features relevant to the NLP task at hand.

By iteratively applying convolution, activation, pooling, and fully connected layers as given in algorithm 4, CNNs excel in extracting hierarchical representations of textual data. This makes them invaluable tools for feature extraction tasks in NLP, empowering applications such as sentiment analysis, text classification, and named entity recognition with rich and informative linguistic features.

Algorithm 4 CNN for NLP Feature Extraction

Input: Text data X
Output: Hierarchical linguistic features
for each convolutional layer **do**
    Apply convolution: $(f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a)$
end for
Apply ReLU activation: $\text{ReLU}(x) = \max(0, x)$
for each pooling layer **do**
    Apply max pooling: $\text{MaxPooling}(x) = \max(x)$
end for
for each fully connected layer **do**
    Apply: $y = f(Wx + b)$
end for
Extract hierarchical linguistic features at different layers.

1.4.2.3 Sentence BERT

Sentence BERT (SBERT) represents a significant advancement in sentence embedding creation. It offers concise representations that capture semantic subtleties. This innovation is driven by pre-trained transformer models like BERT , which excel at generating embeddings rich in semantic information [28].

- **Pre-trained Transformer :** SBERT capitalizes on pre-trained transformer models such as BERT to encode input sentences into dense vector representations. These models undergo extensive training on vast text corpora using self-supervised learning, enabling them to adeptly capture intricate semantic nuances.

- **Sentence Embeddings :** The pre-trained transformer model processes input sentences, crafting high-dimensional embeddings that encapsulate their semantic meanings with finesse, ensuring a nuanced representation of the text.

- **Pooling Strategies :** SBERT employs diverse pooling strategies to consolidate token-level embeddings into singular sentence-level embeddings. These strategies encompass mean pooling, max pooling, and pooling informed by attention weights.

  - **Mean Pooling :** Computes the average of token embeddings.

  - **Max Pooling :** Extracts the maximum value from each dimension of the token embeddings.

  - **Pooling based on attention weights :**Computes a weighted sum of token embeddings based on attention weights.

- **Fine-Tunning :** SBERT offers the flexibility of fine-tuning the pre-trained transformer model to suit specific downstream tasks. This fine-tuning process enables the model to refine its embeddings to cater to the domain or task at hand, thereby enhancing its efficacy and relevance.

Upon training or fine-tuning, SBERT adeptly extracts sentence embeddings from novel text data. These embeddings encapsulate the semantic essence of sentences, serving as invaluable assets for tasks such as semantic search, text similarity evaluation, and text classification.

1.4.3 Topic modeling techniques

1.4.3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that the documents are represented as random mixtures over latent topics, where a topic is characterized by a distribution over words [29]. The underlying assumption is that the prominence of specific topics within a document is indicative of its relevance to a particular query. This approach is highly beneficial in numerous applications. For example, topic models can be utilized to generate pertinent keywords, thereby improving information retrieval processes and enabling more intuitive search experiences. Furthermore, these models can be used to produce concise thematic summaries of textual data [30].

1.4.3.2 BERTopic

In topic modeling, BERT can be used to create sophisticated models like BERTopic, which combines contextual word embeddings with clustering algorithms such as UMAP and HDBSAN. BERTopic offers a novel approach to topic modeling by leveraging the power of BERT embeddings. These embeddings capture rich semantic information about the text. BERTopic then utilizes a class-based TF-IDF weighting scheme to further refine the representation of each document. To facilitate efficient clustering, the high-dimensional BERT embeddings are projected into a lower-dimensional space using the UMAP technique. This dimensionality reduction allows for efficient clustering of documents based on their thematic similarity [25]. BERTopic generates topic representations through three main steps [31]:

1. **Document Embedding :** Each document is transformed into an embedding representation using the SBERT framework, which converts sentences and paragraphs into dense vector representations with a pretrained language model.

2. **Dimensionality Reduction :** The dimensionality of these embeddings is reduced using techniques like UMAP to improve the clustering process. The embeddings are primarily used to cluster similar documents but not directly for generating topics.

3. **Topic Extraction :** Using a custom-based variation of TF-IDF, topic representations are extracted from the document clusters. This assumes that documents containing the same topic are semantically similar.

This multi-step process allows BERTopic to effectively identify and represent topics in a corpus, making it a powerful tool for text analysis. Below is the algorithm 5 for implementing BERTopic.

1.4.3.3 Sentence BERT

SBERT adapts the pre-trained BERT model by employing siamese and triplet network structures. This modification generates semantically rich sentence embeddings, which can be efficiently compared using cosine similarity. Consequently, SBERT produces fixed-size vectors for input sentences, facilitating the comparison of semantically similar sentences [32].

Algorithm 5 BERTopic for Topic Modeling

Input: Corpus D
Output: Topic representations
for each document $d \in D$ **do**
    Convert document to embedding using Sentence-BERT (SBERT):

$$\text{embedding}(d) = \text{SBERT}(d)$$

end for
Apply UMAP to reduce embedding dimensions:

$$\text{reduced\_embeddings} = \text{UMAP}(\{\text{embedding}(d) \mid d \in D\})$$

Cluster the reduced embeddings using HDBSCAN:

$$\text{clusters} = \text{HDBSCAN}(\text{reduced\_embeddings})$$

for each cluster $c \in \text{clusters}$ **do**
    Extract topic representations using class-based TF-IDF:

$$\text{topic}(c) = \text{TF-IDF}(\{d \mid d \in c\})$$

end for

1.5 State of the art

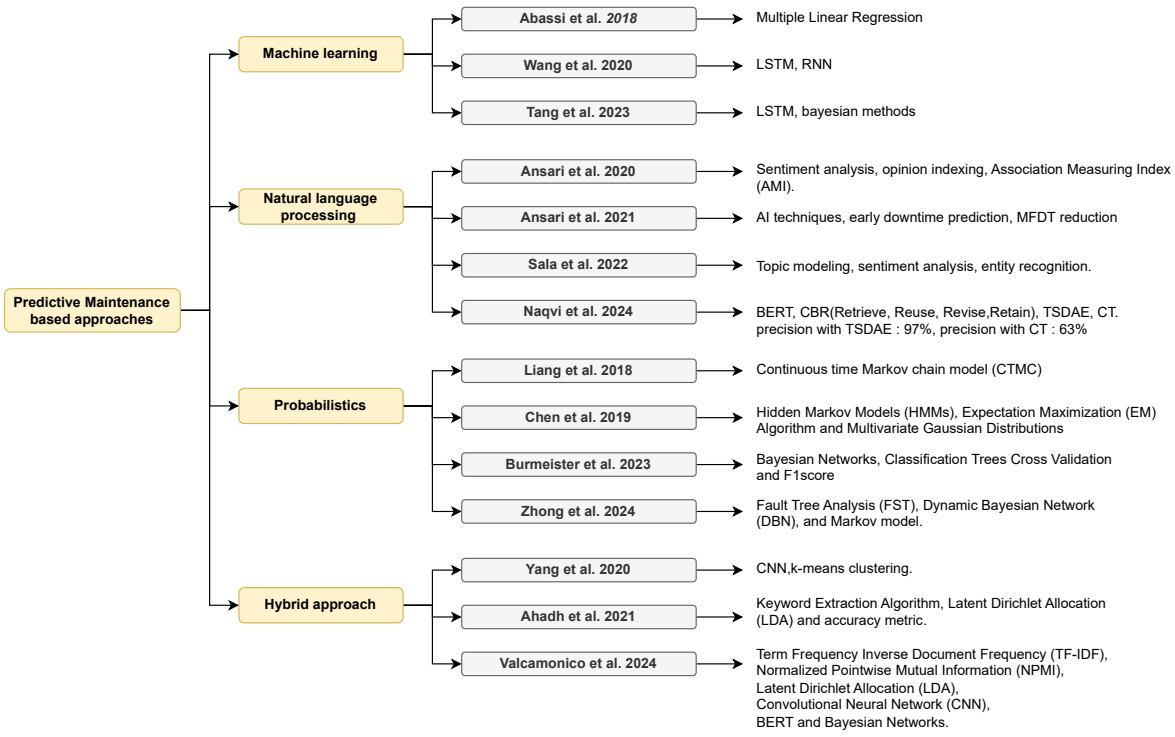


Figure 1.2: Classification of Modern Techniques in Predictive Maintenance

To explore predictive maintenance, various techniques have been implemented, including machine learning, NLP, and statistical methods. Authors have conducted numerous studies to assess the effectiveness of these approaches in predicting and preventing equipment failures. The graph above 1.2 explains different techniques used by different researchers to achieve this purpose.

By categorizing works into machine learning and natural language processing approaches, researchers aim to better understand the strengths and limitations of different predictive maintenance techniques.

1.6 Analysis and discussion

The state of the art in predictive maintenance includes different advanced methodologies, each with distinct advantages and limitations. The following table 1.1 explains the advantage and limitation of each of the previous methods.

Table 1.1: Comparison of Different Methods

Method	**Advantage**	**Limitation**
ML [33, 34, 35]	- Significantly enhances the accuracy of predictive models.	- Highly dependent on the availability of high-quality historical data.
NLP [36, 37, 38, 39]	- Effective in extracting insights from textual maintenance records.	- Highly dependent on high-quality, consistent data; - Computationally demanding; - It struggles with technical language nuances.
Probabilistic [40, 41, 42, 43]	- Improve accuracy of RUL predictions.	- Require precise historical data; - Complex to develop; - Substantial computational overhead.
Hybrid modeling [44, 45, 46]	- Enhance robustness and versatility by combining various techniques	- Complex; - Resource-intensive; - Challenging to maintain and update

Given these considerations, our approach will integrate Probabilistic Graphical Models and Natural Language Processing techniques to address the limitations and leverage the strengths of these methodologies.

1.7 Conclusion

In conclusion, this chapter has provided a comprehensive overview of the concepts and techniques involved in implementing predictive maintenance systems. From the evolution of industrial maintenance to the application of machine learning and natural language processing techniques, we have explored the foundational elements essential for effective predictive maintenance strategies.

Looking ahead, the subsequent chapter will showcase our innovative contributions in implementing predictive maintenance within Cevital, a leading player in the agro-alimentary industry. By applying these advanced techniques, we aim to demonstrate tangible improvements in operational efficiency and continuous production, underscoring the transformative potential of predictive maintenance in modern industrial settings.

Chapter2

Proposed Approach

Contents

The vast amount of textual data available today presents both opportunities and challenges. Analyzing this data effectively requires innovative approaches to extract meaningful insights. This chapter details our contribution to this field, focusing on extracting valuable information from intervention descriptions to enhance maintenance planning. While previous research has explored machine learning with textual data using classification techniques, limitations arise when domain expertise and labeled data are lacking.

To address these limitations, we propose a novel approach incorporating BERTopic for topic modeling alongside autoencoders for robust feature extraction, aiding in the identification of different equipment states. These states will be integrated into a Bayesian network for prediction and decision-making. By leveraging these advanced techniques, we aim to improve the predictive maintenance process, providing more accurate and actionable insights to optimize operational efficiency and reduce downtime.

2.1 Methodology

The methodology employed in this study is designed to systematically develop and refine a predictive maintenance model for the CEVITAL company. The following sections provide a detailed description of each step involved, ensuring clarity and reproducibility of the process.

This approach architecture is sketched in Figure 2.1 and involves four main steps.

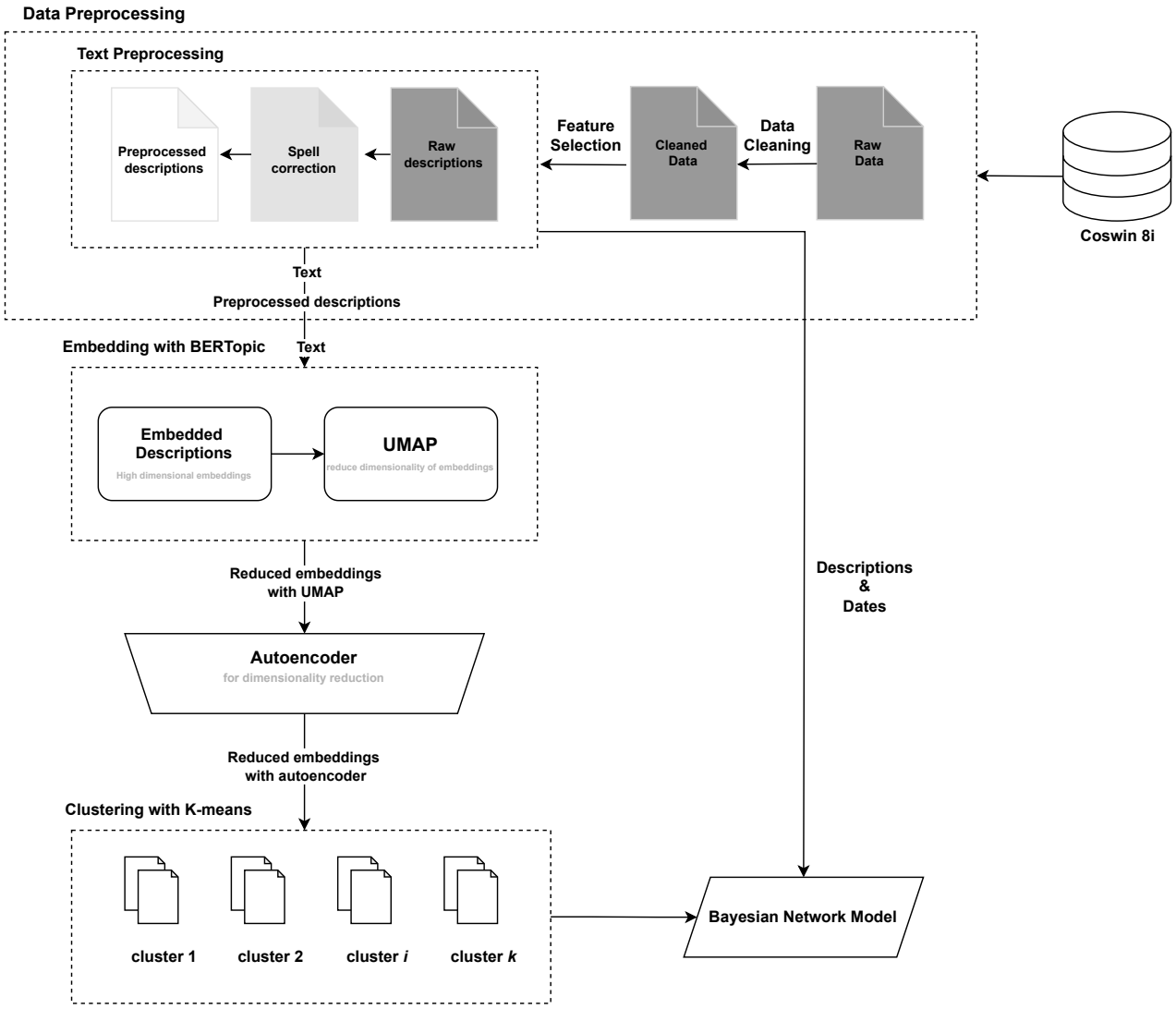


Figure 2.1: Architecture of the proposed approach

2.1.1 Data collection

The data collection phase is the foundational step in our predictive maintenance process. The data was sourced from the Computerized Maintenance Management System (CMMS) Coswin 8i, specifically from the CEVITAL company. To extract the necessary data, we utilized SQL Server Management Studio (SSMS), a robust database management tool, through which SQL queries were executed to gather data in various tabular formats. The data collected spans the entire year of 2023. As a result, we obtained five distinct tables.

The figure 2.2 illustrates the Entity-Relationship Diagram, which presents the tables and their relationships clearly.

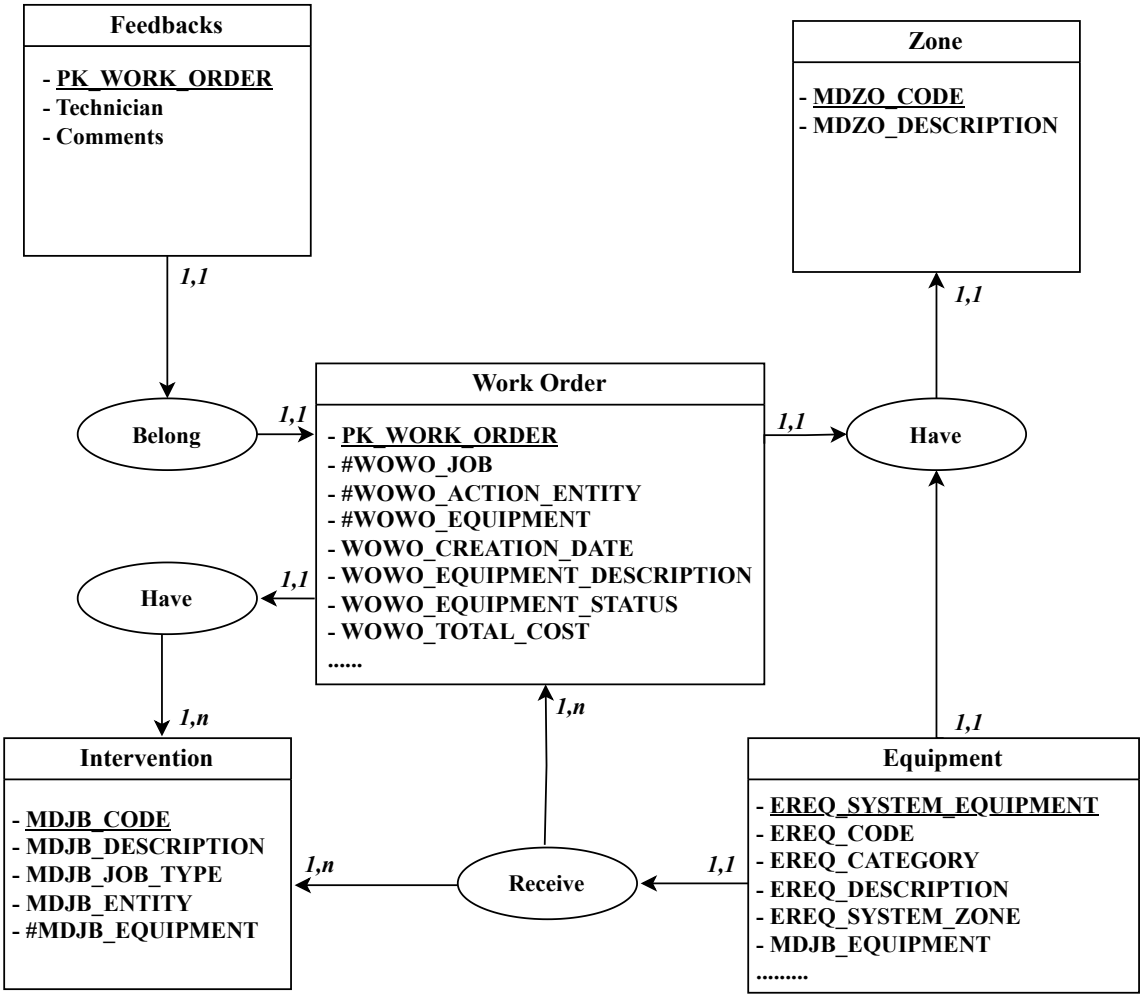


Figure 2.2: Entity-Relationship Diagram

2.1.1.1 Data description

To provide a comprehensive understanding of the collected data, it is essential to describe the structure and contents of each table in our dataset. The following table 2.1 summarizes the key attributes and dimensions of the data we gathered from the CMMS Coswin 8i system, covering various aspects of the maintenance activities at CEVITAL.

2.1.2 Exploratory Data Analysis

The EDA phase is a crucial step in our methodology, designed to uncover insights and underlying patterns within the collected data. This phase involves a series of analytical techniques aimed at summarizing the main characteristics of the data, often with visual methods. By thoroughly exploring the dataset, we can identify significant trends, relationships, and anomalies that inform the subsequent stages of our predictive maintenance model develop-

Table 2.1: Data description

Table	Size	Description
Intervention	(144851, 8)	This table records the maintenance interventions carried out, including details such as the intervention ID, description, type, and location.
Work Order	(11461, 89)	Captures work orders issued for maintenance tasks, including fields like work order ID, priority, status, the person in charge, and associated equipment and intervention.
Equipment	(35256, 60)	Lists all equipment managed within the CMMS, with attributes such as equipment ID, category, description, and location.
Zone	(890, 49)	Defines the zones or areas where equipment is located, containing zone ID, name, and description.
Feedback	(11461, 3)	Contains feedback related to the maintenance work performed, including work order ID, the technician or personnel responsible for carrying out the maintenance task, and comments.

ment.

2.1.2.1 Statistical Summaries

The first step in our EDA process involves generating descriptive statistics for the numerical columns in our dataset. Descriptive statistics provide a summary of the central tendencies, dispersion, and overall shape of the distribution of the data. Specifically, we calculated measures such as the mean, median, standard deviation, minimum, and maximum values for each numerical attribute. For instance, figure 2.3 presents a statistical summary of the *Work Order* table, offering insights into the distribution and variability of key maintenance metrics.

	PK_WORK_ORDER	TIMESTAMP	WOWO_ALLOCATED_HOURS	WOWO_CAN_CHANGE_EQUIPMENT	WOWO_CODE	WOWO_COMPLETION_RATE	WOWO_CON_AVAILABILITY	WOWO_CONFORMITY_STATUS	WOWO_DOWN_TIME
count	11461.000000	11461.000000	11461.000000	11461.000000	1.146100e+04	11461.000000	11461.0	11402.0	11461.000000
mean	792533.666259	17.784487	0.000873	0.774278	2.023071e+09	88.625925	4.0	0.0	174.506413
std	39023.737970	43.673703	0.029527	0.418075	3.904124e+04	94.467491	0.0	0.0	13209.451717
min	721732.000000	2.000000	0.000000	0.000000	2.023000e+09	0.000000	4.0	0.0	0.000000
25%	758930.000000	10.000000	0.000000	1.000000	2.023037e+09	100.000000	4.0	0.0	0.000000
50%	790939.000000	14.000000	0.000000	1.000000	2.023069e+09	100.000000	4.0	0.0	0.000000
75%	828686.000000	19.000000	0.000000	1.000000	2.023107e+09	100.000000	4.0	0.0	0.000000
max	856681.000000	4028.000000	1.000000	1.000000	2.023135e+09	5000.000000	4.0	0.0	1000000.000000

Figure 2.3: Statistical Summaries

2.1.2.2 Data Visualization

To complement the numerical summaries, we employed various data visualization techniques to illustrate the distributions, relationships, and trends within the data. Visualizations are powerful tools for interpreting complex datasets, making patterns more discernible and accessible.

Figure 2.4 presents a bar plot detailing the distribution of entities across CEVITAL's factories. The horizontal axis enumerates the different factories, while the vertical axis quantifies the number of entities within each facility. This visualization underscores the varying operational scales of CEVITAL's factories, offering insights into the organizational structure and resource allocation within the company.

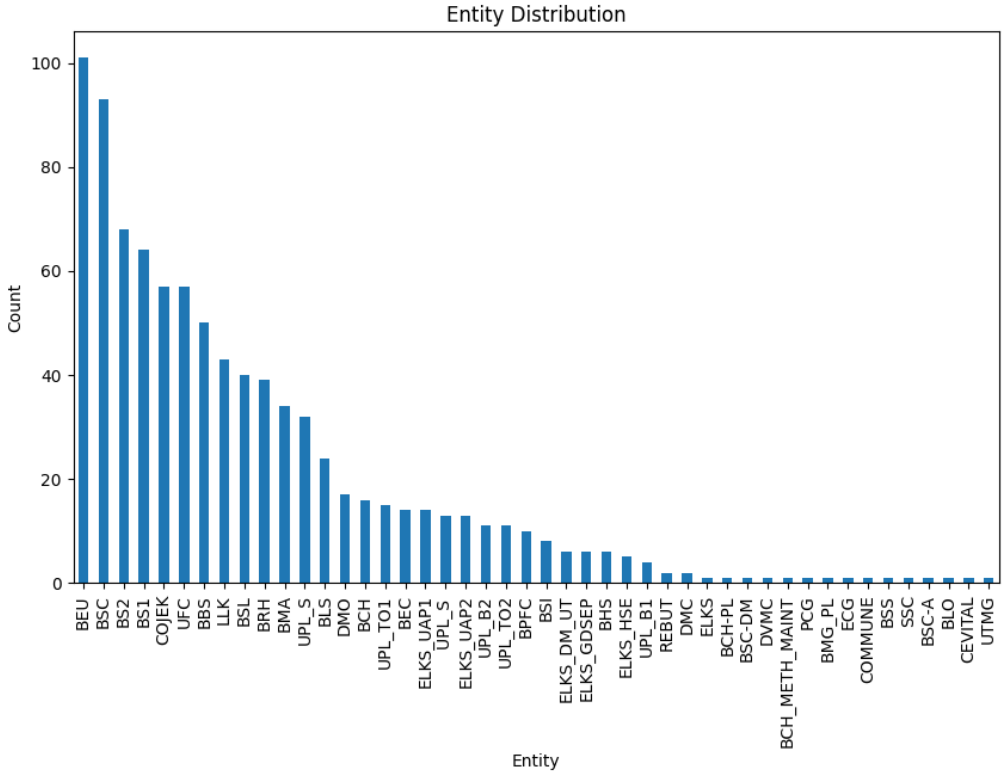


Figure 2.4: Entities distribution across CEVITAL's factories

Figure 2.5 illustrates a bar plot depicting the various types of interventions carried out. The horizontal axis categorizes the different types of interventions, while the vertical axis represents the frequency of each type. Each bar corresponds to a specific intervention type, providing a clear visual comparison of their prevalence.

To supplement the bar plot, Table 2.2 outlines the detailed descriptions of each intervention type, as identified by their respective codes.

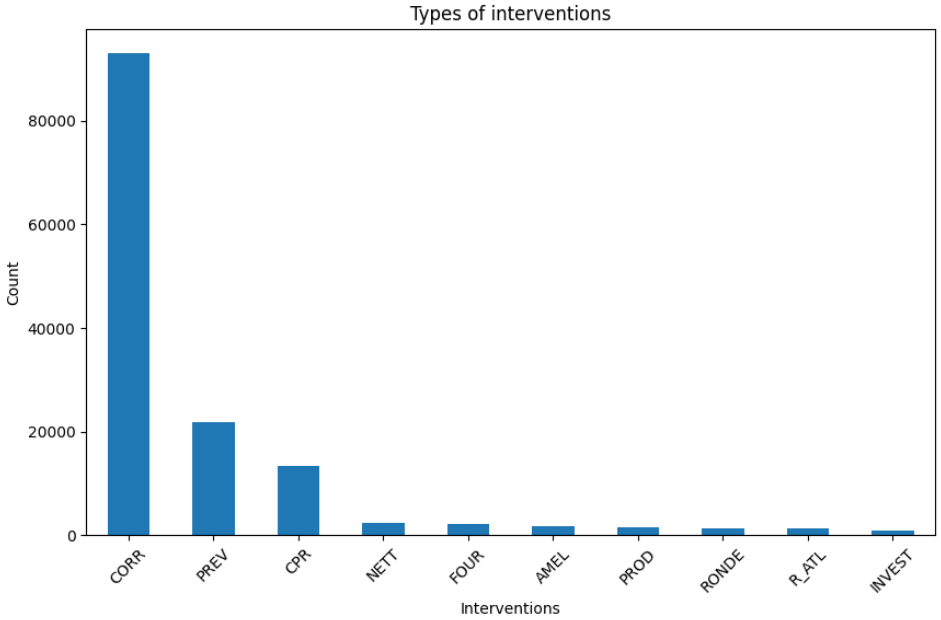


Figure 2.5: Types of interventions

Table 2.2: Intervention types descriptions

Code	**Description**
CORR	Correctif
PREV	Préventif
CPR	Correctif réalisé par la production
NETT	NETTOYAGE
FOUR	FOURNITURES DIVERSES
AMEL	Amélioratif
PROD	PRODUCTION
RONDE	RONDE
R_ATL	REPARATION ATELIER
INVEST	Investissement (Nouvelle Installation)

The bar plot in Figure 2.5 indicates that corrective interventions are the most common, reflecting the frequent need for corrective actions within the factories. This is followed by preventive interventions, highlighting a significant focus on preventative measures. In contrast, interventions such as R_ATL and INVEST are less frequent, indicating specific, less recurrent maintenance activities. This comprehensive visualization and accompanying table provide valuable insights into the maintenance strategies and operational priorities.

(a) Equipment level (b) Completion rate (c) Work order duration

Figure 2.6: Exploratory Data Analysis of the table Work order

Figure 2.6 provides an exploratory data analysis of the work order table through three distinct subfigures, each highlighting a specific aspect of the data.

Subfigure 2.6a illustrates the distribution of work orders across different equipment levels. The plot reveals the concentration of work orders in specific equipment levels, indicating the areas with higher maintenance needs and potential focus for operational efficiency improvements.

Subfigure 2.6b presents the work order completion rate. The x-axis denotes the percentage of work orders completed, while the y-axis indicates the count of work orders of each rate.

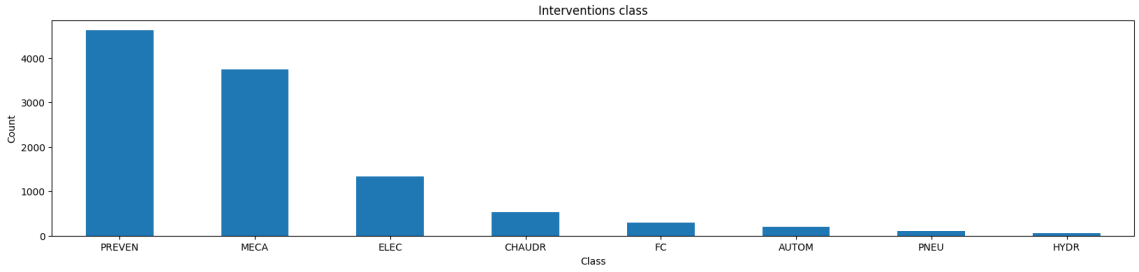
The final subfigure 2.6c depicts the duration of work orders in hours. The x-axis categorizes different durations, and the y-axis shows the frequency of work orders falling within each duration category. This visualization helps in identifying the average and outlier durations of work orders, providing insights into the efficiency of the maintenance operations and highlighting areas that may require process optimization.

Figure 2.7 presents an exploratory data analysis of the work order table through two insightful subfigures, each highlighting a different dimension of the data.

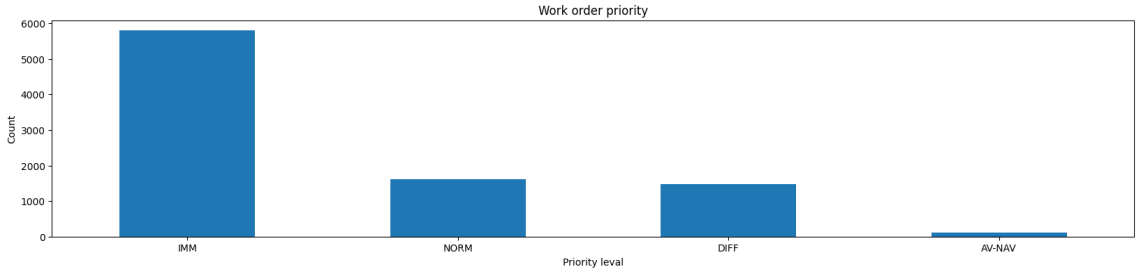
Subfigure 2.7a illustrates the distribution of various intervention classes within the work orders. The x-axis represents the different classes of interventions, while the y-axis indicates the frequency of each class. This visualization helps in identifying which types of interventions are most prevalent, providing a clear overview of the maintenance strategies and their emphasis on specific intervention categories such as preventive, corrective, or palliative measures.

Subfigure 2.7b depicts the distribution of work order priorities. The x-axis categorizes the different priority levels, while the y-axis shows the number of work orders associated with each

priority level. This plot is essential for understanding how maintenance tasks are prioritized within the organization, highlighting the allocation of resources and the urgency of different maintenance activities.



(a) Interventions class



(b) Work order priority

Figure 2.7: Exploratory Data Analysis of the table Work order

2.1.2.3 Correlation Analysis

In addition to the exploratory data analysis presented in Figures 2.6 and 2.7, a correlation analysis was performed on the work order table between different features to identify potential predictive relationships. This table was selected for its rich numerical data across numerous columns, making it a prime candidate for identifying potential predictive relationships. A heatmap was utilized to visually represent the correlation matrix, highlighting both positive and negative correlations among the variables as illustrated in the figure 2.8.

The analysis revealed that most correlations were either weak or negative. However, a few key positive correlations were identified:

- **WOWO_EQUIPMENT_LEVEL and WOWO_EQUIPMENT_TYPE**: This positive correlation suggests that certain equipment levels are consistently associated with specific types of equipment, indicating standardized equipment classifications.
- **WOWO_EQUIPMENT_JOB and WOWO_JOB_STATUS**: A positive corre-

lation here implies that certain jobs are closely linked with specific job statuses, potentially indicating standardized job processes.

- **WOWO_DURATION and WOWO_PLAN**: The positive correlation between work order duration and planning suggests that better-planned work orders tend to have longer durations, possibly due to more comprehensive maintenance activities.
- **WOWO_JOB_PLANNING_TYPE and WOWO_PLAN_PRIORITY**: This correlation indicates that different job planning types are associated with varying levels of plan priority, highlighting how planning strategy influences priority settings.

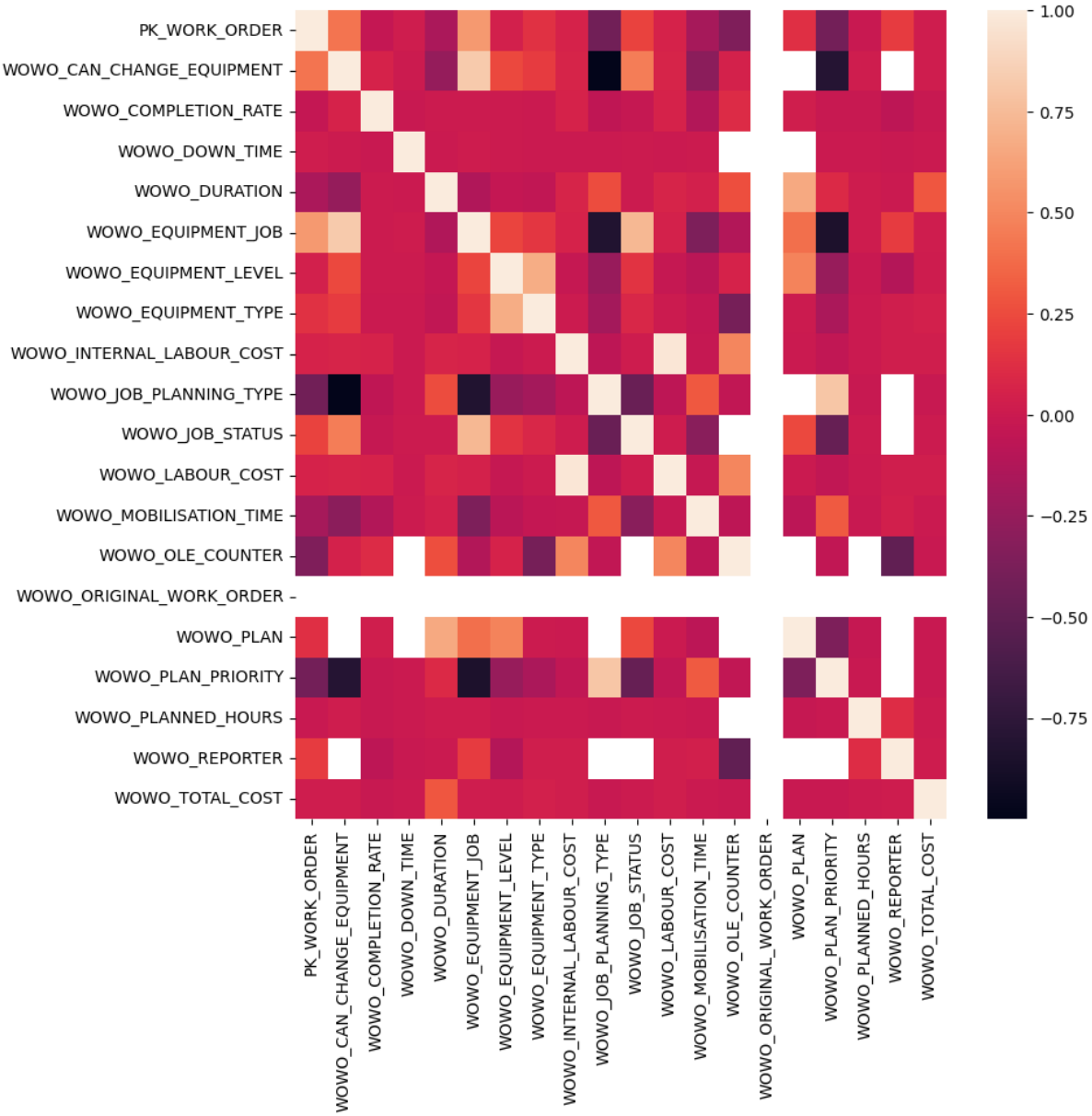


Figure 2.8: Correlation matrix

2.1.3 Data preprocessing

Data preprocessing is a critical step in machine learning tasks, involving the transformation of raw data into a format suitable for modeling. This process includes cleaning data by handling missing values, removing duplicates, and correcting errors. It also involves normalizing or standardizing data to ensure consistent scale, encoding categorical variables, and feature selection to narrow down the set of input variables to those that contribute most significantly to the predictive task, thus reducing dimensionality and computational complexity. Effective data preprocessing enhances the model's performance, reliability, and generalizability, ultimately leading to more accurate and robust predictions.

2.1.3.1 Data cleaning

In the data cleaning phase, conducted across all tables within our dataset, we systematically addressed various issues to ensure data integrity and reliability for subsequent analysis. This comprehensive approach involved handling missing values and removing redundant columns across all tables.

Handling Missing Values

Missing values were addressed across all tables in the dataset by adopting a tailored approach based on the extent and impact of missingness. For tables where missing values constituted a small percentage of the total records, rows containing missing values were selectively removed to preserve the integrity of the dataset. This targeted deletion of rows with missing values ensures that the majority of the dataset remains intact while mitigating the impact of missing values on subsequent analysis. In cases where missing values were more pervasive or concentrated within specific columns, a column-wise approach was employed. Columns with a significant proportion of missing values, deemed to have limited utility for analysis, were removed entirely from the dataset. This strategic removal of columns with high missingness reduces noise and simplifies subsequent analysis without compromising the overall integrity of the dataset.

Removing redundant columns

Another crucial aspect of data cleaning involves identifying and removing redundant columns that do not contribute meaningful information to the analysis. In our dataset, we

observed columns where all records held identical values, providing no variability or predictive power as illustrated in 2.9. Consequently, we made the decision to remove such columns from each table, streamlining the dataset and reducing unnecessary complexity. By eliminating redundant columns across all tables, we not only enhance computational efficiency but also improve the interpretability and effectiveness of subsequent modeling efforts.

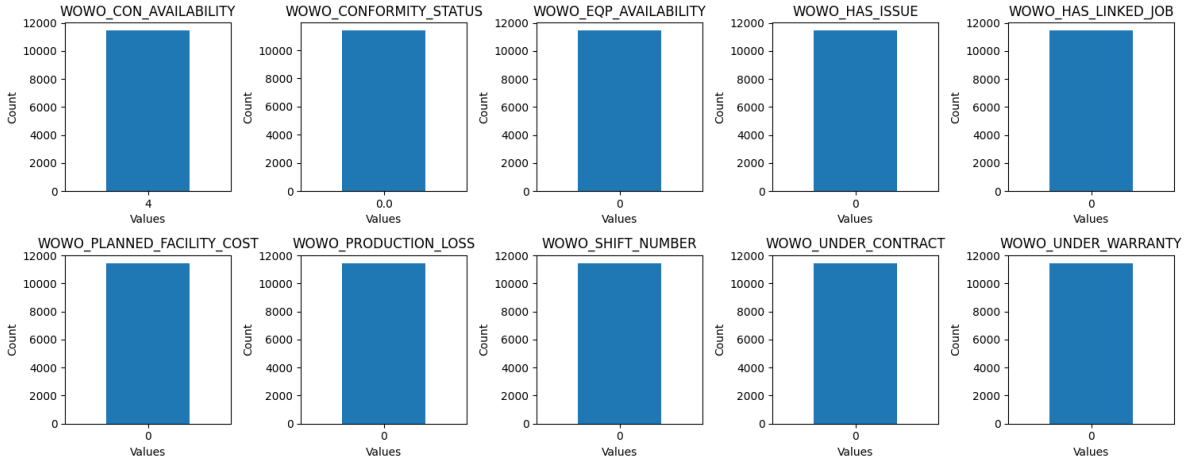


Figure 2.9: Redundant columns analysis

2.1.3.2 Feature selection

Following the data cleaning phase, we proceeded with the feature selection process, focusing on merging and refining the dataset to retain only the most relevant information for our predictive maintenance task. Initially, we identified the two tables with the most pertinent columns post-cleaning: Work Order and Intervention. Recognizing the significance of these tables in capturing maintenance activities, we merged them to create a consolidated dataset.

Upon merging, our attention turned to refining the dataset further. Given the diversity of maintenance activities, we opted to narrow our focus to interventions categorized as corrective maintenance, which are often indicative of imminent equipment failures or malfunctions. This targeted approach allowed us to extract insights specific to critical maintenance events, enhancing the predictive capabilities of our models.

Subsequently, we filtered the dataset to concentrate solely on interventions related to a particular equipment type: "Convoyeur". This deliberate focus enabled us to delve deeply into the maintenance history and performance of this specific equipment, facilitating more nuanced analysis and prediction.

As a final step in the feature selection phase, we curated the dataset to include only the most informative columns essential for our predictive modeling task. These included:

- **Textual Data:** Intervention description and equipment description, providing contextual information about the maintenance activities and the equipment involved.
- **Date:** Intervention date, enabling temporal analysis and trend identification.
- **Location:** Geographic location of the intervention, providing spatial context to maintenance events.
- **Cost:** We also incorporated repair costs attribute, to capture the financial implications of maintenance activities.

By streamlining the dataset to include these key attributes, we ensured that our subsequent analysis would be focused, efficient, and aligned with the objectives of our text mining for predictive maintenance endeavors. Additionally, to facilitate a chronological understanding of maintenance events, we sorted the dataset by date, enabling sequential analysis of maintenance activities and their impact on equipment performance and reliability. All of this steps are summarized in the diagram illustrated in figure 2.10.

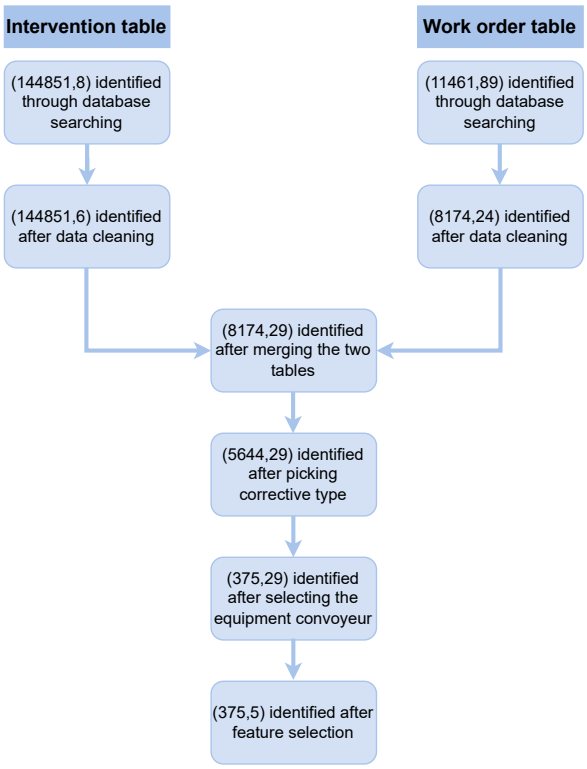


Figure 2.10: Dataset study flowchart

2.1.3.3 Data transformation

Once the data has been cleaned, the next crucial step is data transformation. This process involves converting the cleaned data into a suitable format for analysis and modeling. One significant aspect of data transformation in our study is the handling of the date column. We parsed the dates into a uniform format to standardize the dataset.

2.1.3.4 Text preprocessing

With the dataset now refined and enriched with textual information, our attention is directed towards harnessing the inherent value embedded within textual data for predictive maintenance analysis. Textual data represents a rich source of contextual information, offering nuanced insights into maintenance activities and equipment performance. To unlock the latent predictive potential contained within these textual descriptions, we will embark on a systematic text preprocessing endeavor.

In this step, we focus on preprocessing the text in the MDJB_DESCRIPTIONS column, which contains descriptions of interventions before the maintenance occurred. This preprocessing involves transforming the textual data through techniques such as tokenization, lemmatization, and stopwords removal, preparing the text for clustering tasks. Once preprocessed, the text is ready to be converted into numerical representations suitable for further analysis and modeling.

In our case we skipped using lemmatization, stemming and stop words removal to avoid changing the semantic of the description when using the BERTopic, ex : "conveyor engine was not functional" after removing stop words will be "conveyor engine was functional" which changed the whole meaning of the sentence.

the preprocessing techniques used in our case are :

- **Lower casing:** this operation is performed for better visualization only and does not affect the performance of the model.
- **Spell correction:** a lot of words in the text were abbreviated or misspelled, so it is essential to correct these words to help the model recognize the semantic in each one.
- **Punctuations removal:** punctuations are not used in catching the semantic in descriptions so we preferred to remove them to simplify the preprocessing

As illustrated in Figure 2.11, the preprocessing techniques are applied systematically to the maintenance descriptions. The figure showcases an example of an original description and the steps involved in transforming it for analysis, highlighting the importance of each preprocessing step.

	MDJB_DESCRIPTION	numbers_removed	punctuation_removed	lowercas	spell correction
0	Changement trois rouleau supérieure de tapis 3015	changement trois rouleau supérieure de tapis	changement trois rouleau supérieure de tapis	changement trois rouleau supérieure de tapis 3015	changement trois rouleau supérieure de tapis
1	Ecrasement bouteilles sur la TDC	ecrasement bouteilles sur la tdc	écrasement bouteilles sur la tdc	ecrasement bouteilles sur la tdc	écrasement bouteilles sur la tdc
2	HEUFTE DEFAILLANT EJECTION BOUTEILLES CONFORME	heufte defaillant ejection bouteilles conforme	heufte defaillant ejection bouteilles conforme	heufte defaillant ejection bouteilles conforme	heufte defaillant ejection bouteilles conforme
3	Rupture chaine convoyeur C03 sortie bandroleuse	rupture chaine convoyeur c sortie bandroleuse	rupture chaine convoyeur c sortie bandroleuse	rupture chaine convoyeur c03 sortie bandroleuse	rupture chaine convoyeur c sortie bandroleuse
4	Arret du convoyeur sortie salle blanche	arret du convoyeur sortie salle blanche	arrêt du convoyeur sortie salle blanche	arret du convoyeur sortie salle blanche	arrêt du convoyeur sortie salle blanche
5	Écrasement bouteilles	écrasement bouteilles	écrasement bouteilles	écrasement bouteilles	écrasement bouteilles
6	Blocage des packs a l'entrée	blocage des packs a l'entrée	blocage des packs a l'entrée	blocage des packs a l'entrée	blocage des packs a l'entrée
7	Cisaillement convoyeur entree pasto	cisaillement convoyeur entree pasto	cisaillement convoyeur entree pasto	cisaillement convoyeur entree pasto	cisaillement convoyeur entree pasto

Figure 2.11: Illustration of each stage of text preprocessing

By meticulously cleaning and preprocessing the data, we ensured that the dataset was well-suited for the clustering algorithm, thereby enhancing the accuracy and efficacy of our model.

2.1.4 Text Clustering

Following the text preprocessing step, we proceed to the clustering process. By applying clustering algorithm, we aimed to identify distinct clusters that correspond to the conveyor's behavior at different points in its operational lifespan. Clustering allows us to group similar maintenance descriptions, facilitating a deeper understanding of common patterns and anomalies in the conveyor's operation.

In our proposed methodology, we introduce an advanced clustering approach that leverages BERTopic combined with an autoencoder and K-means. BERTopic is a powerful tool for topic modeling that creates dense embeddings of textual data, capturing nuanced semantic relationships. To enhance these embeddings, we employ an autoencoder, a neural network designed for efficient dimensionality reduction. The autoencoder processes the high-dimensional embeddings generated by BERTopic, compressing them into a lower-dimensional space while preserving their essential features. This step is crucial for reducing computational complexity and improving the clustering performance.

After obtaining the reduced-dimensional features from the autoencoder, we apply K-means clustering to effectively group the intervention descriptions. K-means is a widely used clustering algorithm that partitions data into distinct clusters based on their similarity. By applying

K-means to the autoencoder-reduced embeddings, we ensure that the clustering process is both efficient and accurate, allowing us to uncover meaningful patterns within the maintenance data. This whole process is summrized in algorithm 6.

By identifying these distinct clusters, we gained valuable insights into the operational dynamics of the conveyor. This information is instrumental in developing predictive maintenance models that can anticipate future maintenance needs based on historical patterns, ultimately enhancing the reliability and efficiency of the conveyor system.

Algorithm 6 Text Clustering with BERTopic, Autoencoder, and K-means

1: **Input:** Maintenance descriptions dataset D
2: **Output:** Clustered maintenance descriptions
3: **Step 1: Text Preprocessing**
4: Perform text preprocessing on dataset D to clean and prepare maintenance descriptions.
5: **Step 2: Embedding with BERTopic**
6: Use BERTopic to create dense embeddings of the preprocessed maintenance descriptions:
- BERTopic generates high-dimensional embeddings capturing semantic relationships.

7: **Step 3: Autoencoder for Dimensionality Reduction**
8: Employ an autoencoder to compress the high-dimensional embeddings into a lower-dimensional space:
- Train an autoencoder neural network to reduce computational complexity and preserve essential features.

9: **Step 4: K-means Clustering**
10: Apply K-means clustering to the reduced-dimensional embeddings from the autoencoder:
- Partition the maintenance descriptions into distinct clusters based on similarity.

2.1.5 Bayesian Network Model

Following the clustering process, the next step in our methodology involves the creation of a predictive model using a Bayesian network. Bayesian networks are probabilistic graphical models that represent a set of variables and their conditional dependencies via a DAG. This approach is particularly well-suited for predictive maintenance as it can effectively handle the uncertainty and variability inherent in maintenance data.

Algorithm 7 summrize the main steps to model a Bayesian Network for Predictive Maintenance.

Algorithm 7 Bayesian Network Model for Predictive Maintenance

Input: Clustered data, work order dates
Output: Predictive Bayesian network
Step 1: Identify Variables
Define *current_state*, *time_since_last_intervention*, and *next_state*.
Step 2: Determine Network Structure
Construct directed acyclic graph (DAG) representing the conditional dependencies
Step 3: Parameter Learning
for each variable in the network **do**
    Estimate conditional probability distributions based on observed data
    Calculate likelihood of outcomes given the states of parent variables
end for
Step 4: Build Bayesian Network
Construct the Bayesian network using the defined structure and estimated parameters
Encapsulate probabilistic relationships between different factors affecting conveyor performance
Step 5: Model Validation and Use
Validate the model with historical data
Use the model to predict *next_state* based on *current_state* and *time_since_last_intervention*
Step 6: Maintenance Decision Support
Estimate likelihood of future degradation states and maintenance needs
Define a critical state where the conveyor system is at significant risk of failure.
Set a threshold probability in the CPD of *next_state*. When this threshold is reached, trigger proactive measures to avert potential failures.
Schedule proactive maintenance activities to minimize downtime and optimize efficiency

To build our Bayesian network, we first identify the key variables that influence the degradation states and maintenance needs of the conveyor system. These variables include the current state based on the clusters, time since the last intervention derived from work order dates, and the next state, which we aim to predict. A critical state is defined based on the clusters analysis, where the system is deemed at significant risk of failure. We establish a threshold probability in the CPD of the next state. When this threshold is reached, proactive measures are automatically triggered to prevent potential failures.

The structure of the Bayesian network is then determined by establishing the conditional dependencies between these variables, which is achieved through both domain expertise and data-driven techniques. Once the network structure is defined, we proceed to parameter

learning, where the conditional probability distributions for each variable are estimated based on the observed data. This involves calculating the likelihood of various outcomes given the states of their parent variables in the network. The resulting Bayesian network provides a comprehensive model that encapsulates the probabilistic relationships between different factors affecting the conveyor's performance.

The predictive capabilities of the Bayesian network allow us to estimate the likelihood of future degradation states and maintenance needs based on current observations and historical data. This enables the anticipation of potential failures and the proactive scheduling of maintenance activities, thereby minimizing downtime and optimizing operational efficiency. Additionally, the interpretability of the Bayesian network offers valuable insights into the causal relationships within the maintenance data, supporting more informed decision-making.

2.2 Conclusion

In this chapter, we have detailed a thorough approach to predictive maintenance analysis using advanced data preprocessing, clustering, and predictive modeling techniques. We began by refining our dataset, concentrating on the textual descriptions of maintenance interventions.

After preprocessing, we implemented a sophisticated clustering method that combined BERTopic, an autoencoder, and K-means clustering. This method allowed us to create dense textual embeddings, reduce dimensionality, and accurately group intervention descriptions into distinct clusters, revealing important insights into the conveyor system's degradation states throughout its lifecycle.

Finally, we developed a predictive model using a Bayesian network. By incorporating the clusters, this model effectively predicted future degradation states and maintenance needs, facilitating proactive maintenance scheduling and enhancing operational reliability.

In the next chapter, we will take our research to the next level by implementing and rigorously evaluating our approach in the context of text mining for predictive maintenance. Furthermore, we will present the tools and development environment employed in our study

Chapter3

Experimental Study and Results

Contents

This chapter outlines the various methodologies we tested before identifying the optimal solution. We began with several hypotheses based on theory and preliminary data, guiding our selection of experimental approaches. Each method was rigorously tested to address specific aspects of our research problem.

Throughout this chapter, we present an overview of the tools utilized in our experimental journey to identify the optimal approach. We start by detailing our initial experiments in text clustering, followed by the application of Bayesian network models and hidden Markov models. The implementation and results of each tool are thoroughly discussed, culminating in a comparative analysis and concluding remarks. This chapter aims to illustrate our methodological progression, highlighting the significant role of text mining and Bayesian networks in achieving our research objectives.

3.1 Tools presentation

This section outlines the methodological framework used in the research, detailing the specific methods and techniques chosen. The selection of tools was guided by alignment with

research goals and proven effectiveness in similar studies, supported by a thorough literature review and comparative analysis. The chosen methods directly addressed the research question and facilitated achieving the objectives, providing a transparent and replicable roadmap for critical evaluation and future research.

3.1.1 Working environment

3.1.1.1 Google colab

This project heavily relies on Google Colab, a cloud-based platform offered by Google for executing and collaborating on Jupyter notebooks. Colab functions as a sophisticated virtual workspace, eliminating the need for local high-performance computing resources. It fosters research endeavors by enabling researchers to work from any location with an internet connection. Notably, Colab seamlessly integrates with Jupyter notebooks, a versatile document format that blends code, text, and visualizations. This unique combination makes Colab an ideal tool for data analysis, machine learning experimentation, and collaborative research efforts. Furthermore, Colab boasts a pre-installed arsenal of popular machine learning and data science libraries, readily available for immediate use. Perhaps most compellingly, Colab offers free access to powerful hardware resources like GPUs and TPUs . These resources significantly accelerate deep learning tasks, making Colab an indispensable asset for research involving complex computational models.

3.1.2 Programming language

In this section, we explore the programming languages used in this research, with a special emphasis on Python and its extensive libraries. Python was selected due to its flexibility, readability, and comprehensive ecosystem, making it suitable for a variety of tasks, including data preprocessing, analysis, and the implementation of sophisticated machine learning models. We will discuss the specific libraries and tools that were utilized to streamline development and enhance application functionality. This overview will illustrate why Python was chosen and how its features and libraries supported our research efforts.

3.1.2.1 Python

Python is a versatile programming language known for its high-level, user-friendly, and dynamically interpreted nature. It started as a personal project by its creator, who wanted

to create a language that was both elegant and easy to read, featuring indentation to define code blocks instead of using curly braces. Although Python initially didn't gain widespread popularity compared to other languages, it has become highly regarded in the fields of Machine Learning and Artificial Intelligence because it enhances productivity and simplifies complex tasks. This increased utility is largely due to the powerful computing resources available today, which facilitate more efficient workflows even when development is time-consuming [47].

3.1.2.2 Python libraries

In this section, we delve into the Python libraries central to our research in ML and AI. Python's extensive library ecosystem has greatly facilitated our work. NumPy and pandas are essential for data manipulation, while Scikit-learn provides efficient algorithms for predictive analytics. TensorFlow and PyTorch stand out for deep learning tasks, enabling the construction and training of complex neural networks. Additionally, NLTK and spaCy play key roles in natural language processing. These libraries collectively empower us to address intricate ML and AI challenges efficiently and ethically.

- **Numpy :** Numerical Python (Numpy) is a foundational library for scientific computing in Python. It excels at efficient array manipulation, providing multidimensional arrays (NumPy arrays) that significantly outperform built-in Python lists for large datasets. Optimized functions and operators specifically designed for NumPy arrays further accelerate computations. This makes NumPy a natural choice for data analysis, empowering researchers to manipulate and analyze large datasets with ease. Notably, NumPy serves as the bedrock for popular libraries like SciPy and scikit-learn, which leverage NumPy's core functionalities for advanced scientific computing. In some cases, NumPy, when combined with libraries like SciPy and Matplotlib, can even be a viable alternative to MATLAB [48].

- **Pandas :** Pandas is a Python library specifically designed for data analysis. It provides powerful data structures, like DataFrames (similar to spreadsheets), and various functions to simplify and expedite working with structured data. Pandas aims to make data analysis tasks faster, easier, and more expressive by offering efficient tools for data manipulation, cleaning, and analysis [49].

- **Scikit learn :** Scikit-learn (or sklearn) is a widely adopted open-source software library

for Python that empowers researchers and practitioners in the field of machine learning. It offers a comprehensive suite of well-documented and user-friendly algorithms encompassing various machine learning tasks. These tasks include classification (categorizing data points), regression (predicting continuous values), dimensionality reduction (transforming high-dimensional data for efficient processing), and clustering (grouping similar data points). Additionally, scikit-learn provides modules for feature extraction (identifying relevant characteristics from data), data preprocessing (preparing data for machine learning algorithms), and model evaluation (assessing the performance of trained models). This extensive functionality, coupled with its user-friendly API, makes scikit-learn a versatile and popular choice for machine learning in Python [50].

- **Matplotlib :** Matplotlib stands as a cornerstone library in the Python data science ecosystem. Renowned for its ability to generate high-quality visualizations, it empowers users to explore and understand complex datasets through the power of charts and graphs. Matplotlib offers a user-friendly interface, allowing researchers and data scientists of all experience levels to create informative plots with relative ease. Notably, it provides flexibility by supporting a wide range of plot types and output formats, ensuring compatibility with various presentation and analysis needs. This versatility makes Matplotlib a go-to tool for tasks like correlation analysis, visualizing confidence intervals, outlier detection, and data distribution exploration [51].

- **SpaCy :** spaCy is a powerful, open-source library tailored for advanced NLP in Python. It is built for production environments, enabling efficient processing of extensive text data. With support for over 75 languages, spaCy includes pretrained models and components for a range of NLP tasks such as named entity recognition, part-of-speech tagging, and dependency parsing. It is designed to integrate effortlessly with machine learning frameworks like PyTorch and TensorFlow, making it a versatile tool for developing and deploying NLP applications [52].

- **The Natural Language Toolkit :** The Natural Language Toolkit (NLTK) is a versatile Python library for handling human language data. It offers a wide array of tools for text processing, including tokenization, stemming, lemmatization, and part-of-speech tagging. NLTK also provides access to numerous corpora and lexical resources, which greatly aid in the preprocessing and analysis of textual data. Utilizing NLTK allowed

us to efficiently convert raw maintenance logs into structured data, an essential step for our predictive modeling efforts [53].

- **Network :** The network package is a library used for handling, analyzing, creating, manipulating, visualizing, and analyzing networks or graphs, which consist of nodes (vertices) and edges (connections) [18].

- **Pgmpy :** The pgmpy package, which stands for "Probabilistic Graphical Models in Python," is a specialized Python library designed for working with PGMs. PGMs are essential tools for representing and reasoning in uncertain domains, utilizing probabilities to manage uncertainty. These models are extensively used in machine learning, artificial intelligence, and statistics for tasks such as probabilistic inference, decision making, and pattern recognition. Pgmpy provides a comprehensive set of tools for creating, manipulating, and learning various types of probabilistic graphical models, including Bayesian Networks, Markov Networks, and Factor Graphs, and supports functionalities for inference, parameter learning, structure learning, and more [18].

3.1.2.3 Structured Query Language

Structured Query Language (SQL) is a specialized language for managing relational databases, offering a standardized way to query, modify, and structure data, crucial for efficient data management across platforms and applications [54].

Our data was extracted from the CMMS Coswin 8i, using the database management tool SQL Sever Management Studio.

- **SQL Server Management Studio :** SQL Server Management Studio (SSMS) is a comprehensive integrated environment for managing SQL Server databases. It provides tools for configuring, monitoring, and administering instances of SQL Server, enabling efficient database development and maintenance tasks [55].

3.2 Text clustering techniques

To develop the optimal clustering model, we systematically tested numerous topic modeling techniques in conjunction with several dimensionality reduction methods. Each approach was implemented using K-means clustering, and its effectiveness was subsequently evaluated

with the silhouette score. By analyzing the strengths and weaknesses of each technique, we offer a detailed account of our efforts to identify the optimal clustering model.

3.2.1 Term Frequency-Inverse Document Frequency

TF-IDF, a technique that assesses the importance of a word within a document relative to a corpus, is heavily dependent on term frequency. Therefore, word frequency-based preprocessing was applied to prepare the text for the subsequent steps in the process. After that, the preprocessed text data is converted into a numerical representation using TF-IDF.

3.2.1.1 Without dimensionality reduction

In this section, we applied TF-IDF for topic modeling without any dimensionality reduction techniques, assessing the model's ability to extract topics directly from the original dataset.

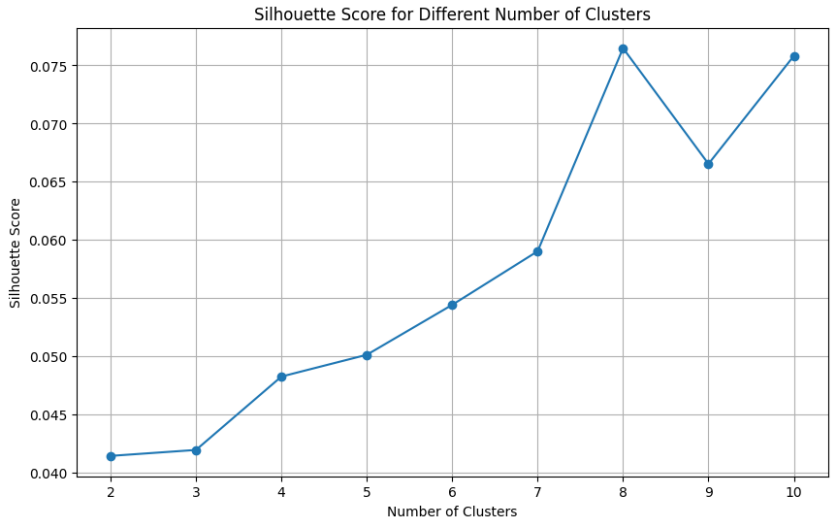


Figure 3.1: Silhouette score - TF-IDF without dimensionality reduction

The low silhouette score observed from the Figure 3.1 when using TF-IDF for clustering likely stems from the inherent sparsity of TF-IDF vectors. This sparsity arises because TF-IDF assigns high weights to only a few relevant terms in each document, resulting in high-dimensional vectors with most elements being zero. This phenomenon can be attributed to two factors in our case:

1. **Vocabulary Expansion :**Document descriptions naturally contain a wide variety of words, including potential misspellings. These unique terms contribute to a larger overall vocabulary.

2. **Limited Term Occurrence :**Each document typically uses only a small subset of all possible terms within this expanded vocabulary.

Consequently, TF-IDF vectors become sparse, containing numerous zero entries. This sparsity significantly impacts the effectiveness of distance-based clustering algorithms like K-means. In simpler terms, sparse vectors make it difficult for the algorithm to accurately measure the distance between data points, hindering its ability to identify distinct and well-separated clusters.

3.2.1.2 Using Umap

The Figure 3.2 shows the silhouette score for when we combined TF-IDF with Umap for dimensionality reduction, aiming to improve topic extraction by reducing the dataset's complexity while maintaining its structure.

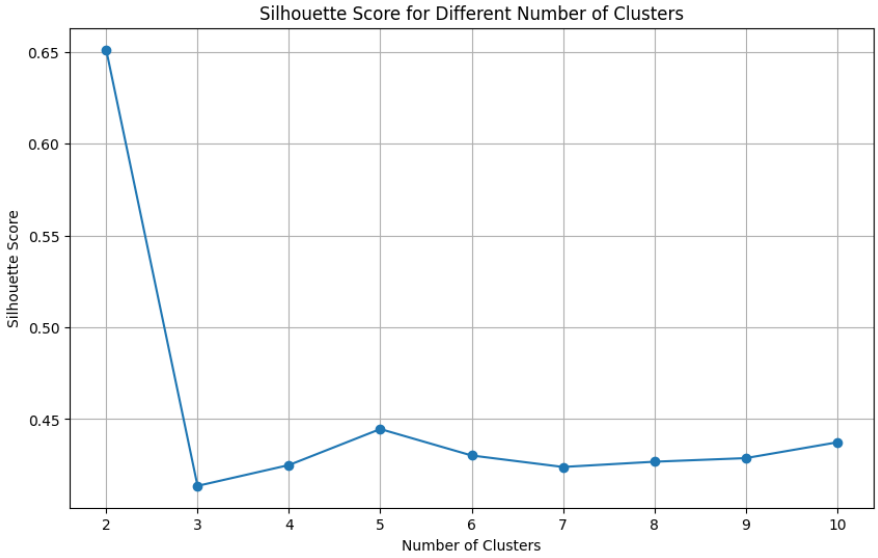


Figure 3.2: Silhouette score - TF-IDF with Umap

The highest silhouette score was achieved with the number of clusters set to 2.

3.2.1.3 Using T-sne

This section explores the use of TF-IDF with T-sne for dimensionality reduction, with the goal of enhancing the model's topic extraction capabilities by mapping the data to a lower-dimensional space. The different silhouette scores where calculated for different K configurations as shown in the Figure 3.3.

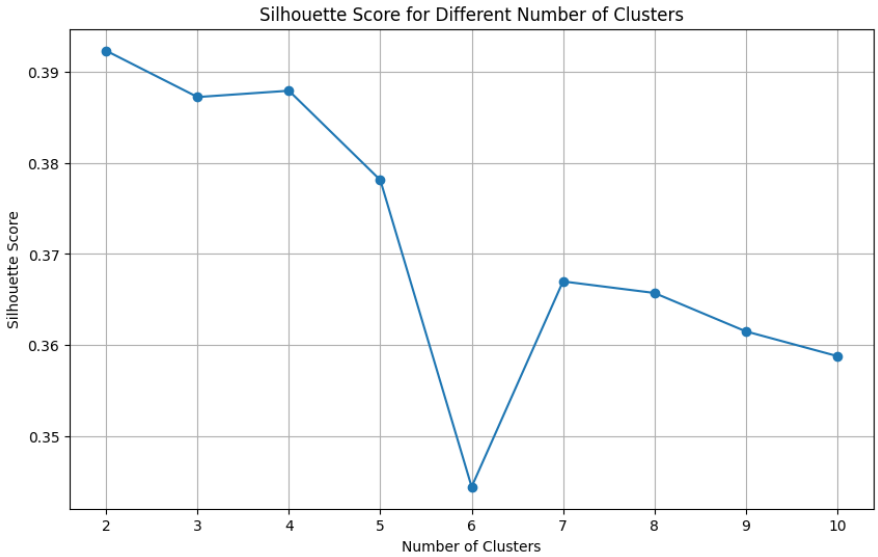


Figure 3.3: Silhouette score - TF-IDF with T-sne

3.2.1.4 Using Autoencoder

In this part, we integrated TF-IDF with an Autoencoder for dimensionality reduction, seeking to optimize the model's performance. The result is depicted in the following Figure 3.4.

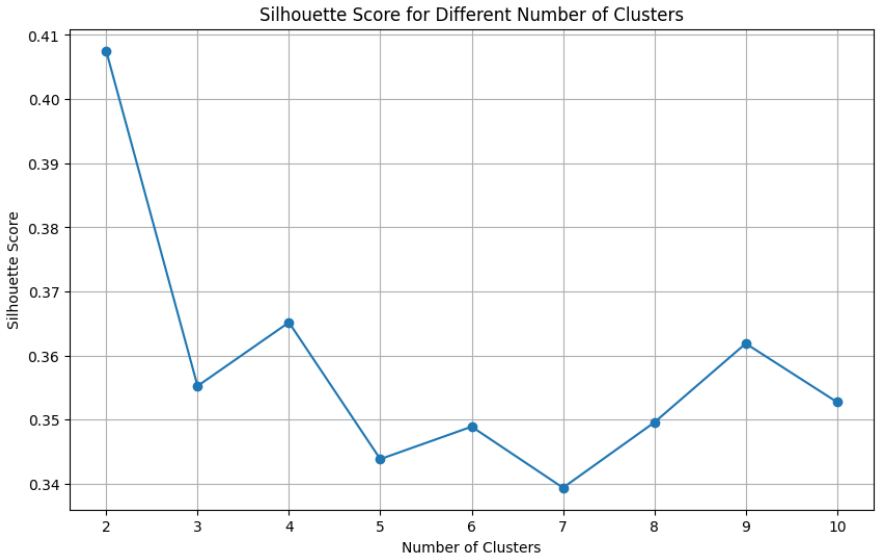


Figure 3.4: Silhouette score - TF-IDF with Autoencoder

3.2.1.5 Using PCA

In this section, we explored the use of TF-IDF for topic modeling in conjunction with PCA for dimensionality reduction.

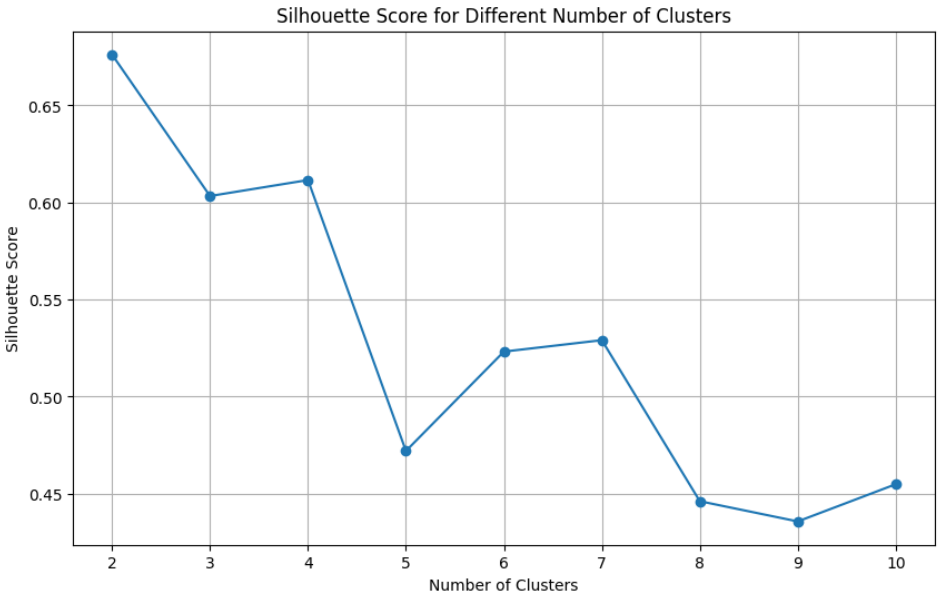


Figure 3.5: Silhouette score - TF-IDF with PCA

The clustering was executed with a range of potential cluster configurations from 2 to 10. Each configuration was evaluated based on the Silhouette Score, as illustrated in the Figure 3.5.

The following Table 3.1 displays the result of the experiments done with the TF-IDF.

Table 3.1: Silhouette scores for different dimensionality reduction's techniques with TF-IDF

Number of clusters	**Raw data**	**Umap**	**T-sne**	**PCA**	**Autoencoder**
2	0.041	**0.651**	**0.392**	**0.676**	**0.407**
3	0.041	0.413	0.387	0.603	0.355
4	0.048	0.424	0.387	0.611	0.365
5	0.050	0.444	0.378	0.472	0.343
6	0.054	0.430	0.344	0.523	0.348
7	0.059	0.423	0.366	0.529	0.339
8	**0.076**	0.426	0.365	0.446	0.349
9	0.066	0.428	0.361	0.435	0.361
10	0.075	0.437	0.358	0.454	0.352

3.2.2 Convolutional Neural Network (CNN)

When Convolutional Neural Networks (CNNs) are utilized for text processing, they do not directly depend on word frequency. Instead, they focus on capturing the semantic meaning of the text through the use of embeddings and convolutions. Consequently, a semantic-based preprocessing approach is applied to the text descriptions before they are fed into the CNN. To ensure uniformity in sequence length, padding is used.

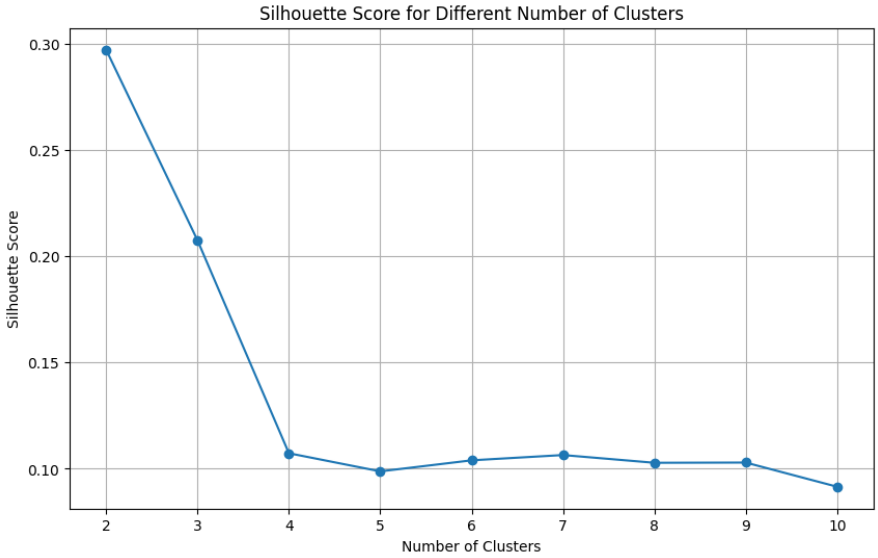


Figure 3.6: Silhouette score for different numbers of clusters - CNN

The highest silhouette score was captured for setting 2 clusters in the K-means algorithm and gave 0.29 as shown in the Figure 3.6

3.2.3 Latent Dirichlet Allocation (LDA)

While LDA does not directly incorporate word semantics, it implicitly captures semantic relationships by grouping frequently co-occurring words. Therefore, applying word frequency based preprocessing to clean and standardize text is crucial to accurately capture the word frequency patterns essential for effective topic modeling. After preprocessing, a range of dimensionality reduction techniques underwent evaluation to ascertain their compatibility with the topic modeling approach, utilizing LDA with 3 topics, and clustering with K-means. To validate the model's performance, we employed the cross-validation method with 5 splits.

3.2.3.1 Without dimensionality reduction

In this section, we examined the application of LDA without any dimensionality reduction

techniques. Our aim was to assess the model's performance in extracting topics directly from the original dataset. A silhouette score of 0.775 was captured when the k-means was trained with a number of clusters of 3 which gave the highest score, as shown in the following Figure 3.7.

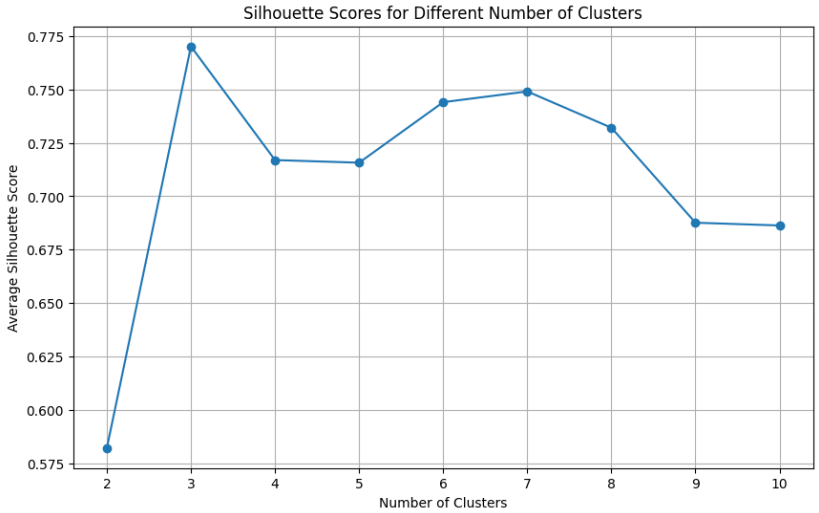


Figure 3.7: Silhouette score - LDA without dimensionality reduction

3.2.3.2 Using Umap

Here, we explored the integration of LDA with Umap for dimensionality reduction. By leveraging Umap, we aimed to enhance the model's ability to extract meaningful topics from the dataset. the Figure 3.8 shows the result of evaluating a range of cluster configurations (k) from 2 to 10 using the K-means algorithm.

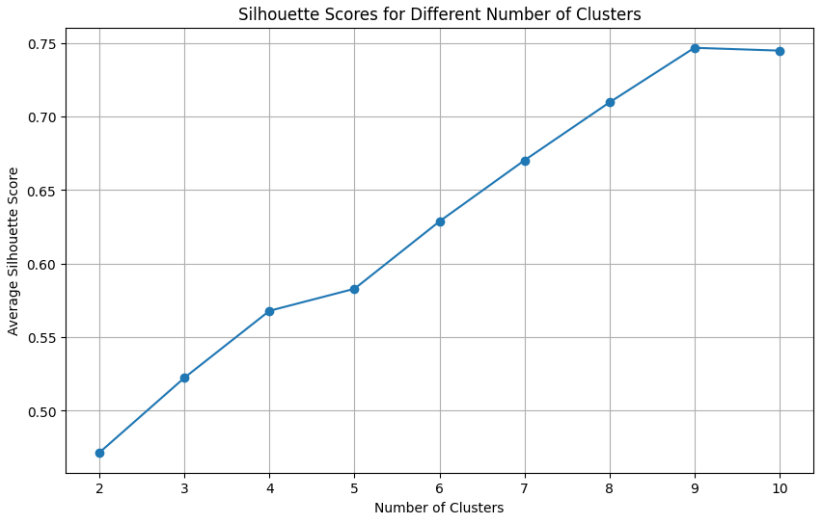


Figure 3.8: Silhouette score - LDA with Umap

3.2.3.3 Using T-sne

This section focused on employing LDA alongside t-SNE for dimensionality reduction. The silhouette score attained was 0.65 during the training of the k-means algorithm with a cluster count of 7, representing the maximum score achieved as shown in the Figure 3.9.

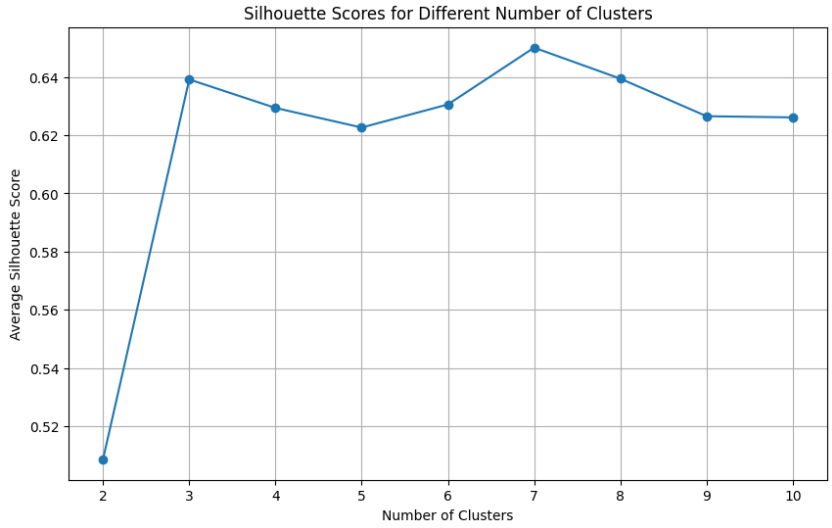


Figure 3.9: Silhouette score - LDA with T-sne

3.2.3.4 Using PCA

In this section, we investigated the application of LDA in combination with PCA for dimensionality reduction. The highest silhouette score was captured when setting the number of clusters for 3, as depicted in the following Figure 3.10.

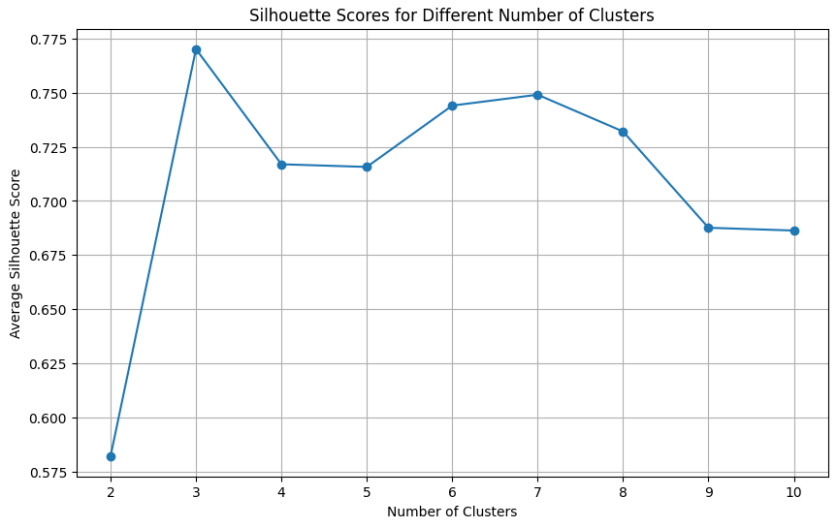


Figure 3.10: Silhouette score - LDA with PCA

3.2.3.5 Using Autoencoder

In this part, we investigated the use of LDA combined with autoencoder for dimensionality reduction. We evaluated a range of cluster configurations (k) from 2 to 10 using the K-means algorithm. The following section presents the captured results.

This one gave the highest silhouette score among the other techniques, with a silhouette score of 0.886 for 10 clusters as shown in the Figure 3.11.

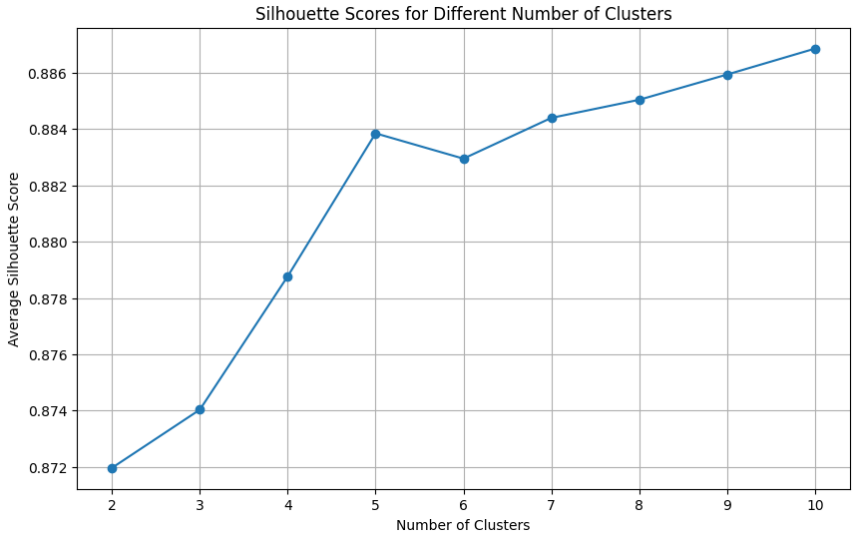


Figure 3.11: Silhouette score - LDA with Autoencoder

The Table 3.2 presents the various scores obtained using different dimensionality reduction techniques in conjunction with LDA and K-means for the description clustering task.

Table 3.2: Silhouette scores for different dimensionality reduction's techniques with LDA

Number of clusters	**Raw data**	**Umap**	**T-sne**	**PCA**	**Autoencoder**
2	0.582	0.471	0.508	0.582	0.871
3	**0.770**	0.522	0.639	**0.770**	0.874
4	0.716	0.567	0.629	0.716	0.878
5	0.715	0.582	0.622	0.715	0.883
6	0.744	0.628	0.630	0.744	0.882
7	0.749	0.670	**0.650**	0.749	0.884
8	0.732	0.709	0.639	0.732	0.885
9	0.687	**0.746**	0.626	0.687	0.885
10	0.686	0.744	0.626	0.686	**0.886**

3.2.4 Sentence BERT

Sentence-BERT (SBERT) is a transformer model designed to prioritize semantic similarity during text clustering tasks. To optimize the descriptions for clustering, a similarity based preprocessing is conducted to ensure they are thoroughly prepared.

3.2.4.1 Without dimensionality reduction

In this section, we applied SBERT for topic modeling without any dimensionality reduction techniques, assessing the model's performance in extracting topics directly from the high-dimensional dataset. The following Figure 3.12 shows the evaluation of this model.

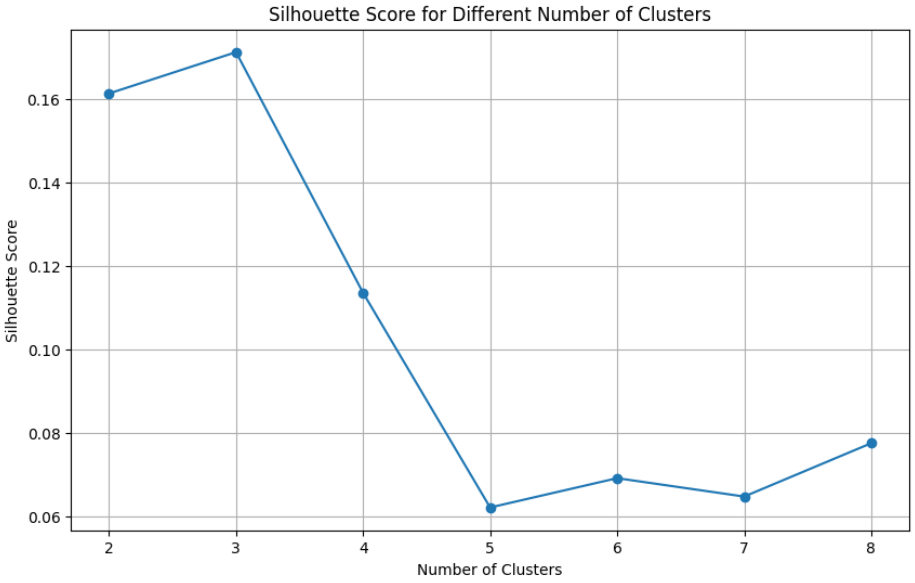


Figure 3.12: Silhouette score for different numbers of clusters - SBERT without dimensionality reduction

3.2.4.2 Using Umap

In this section, we investigated the use of SBERT combined with Umap for dimensionality reduction. The silhouette scores obtained are illustrated in the following Figure 3.13.

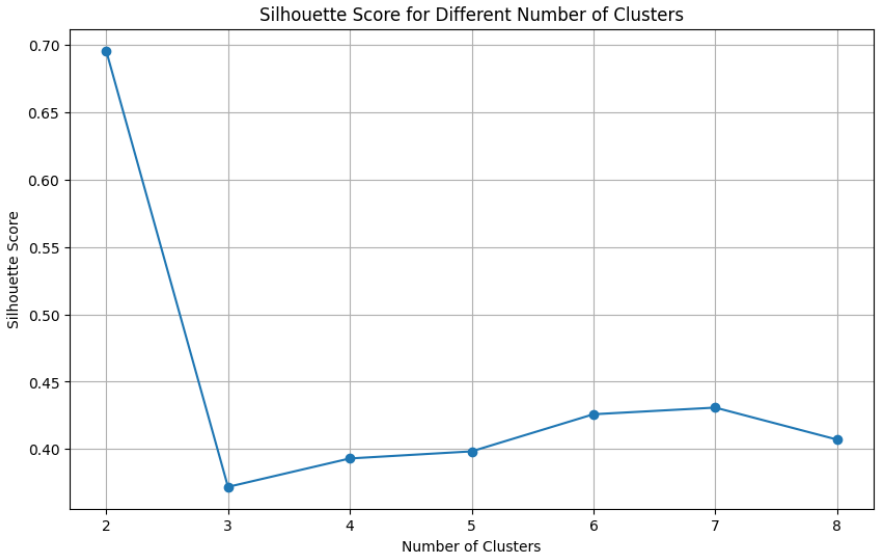


Figure 3.13: Silhouette score for different numbers of clusters - SBERT with Umap

3.2.4.3 Using PCA

In this section, we examined the use of SBERT in conjunction with PCA for dimensionality reduction. The silhouette scores yeild are displayed in the next Figure 3.14.

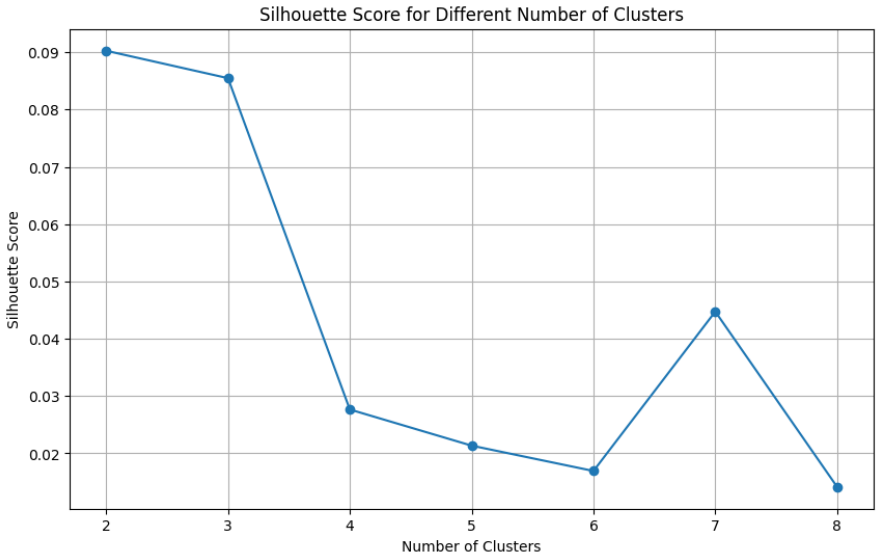


Figure 3.14: Silhouette score for different numbers of clusters - SBERT with PCA

3.2.4.4 Using T-sne

This section explores the use of SBERT combined with t-SNE for dimensionality reduction, aiming to enhance the model's topic extraction capabilities by mapping the data into

a lower-dimensional space that maintains its inherent relationships, as demonstrated in the following Figure 3.15.

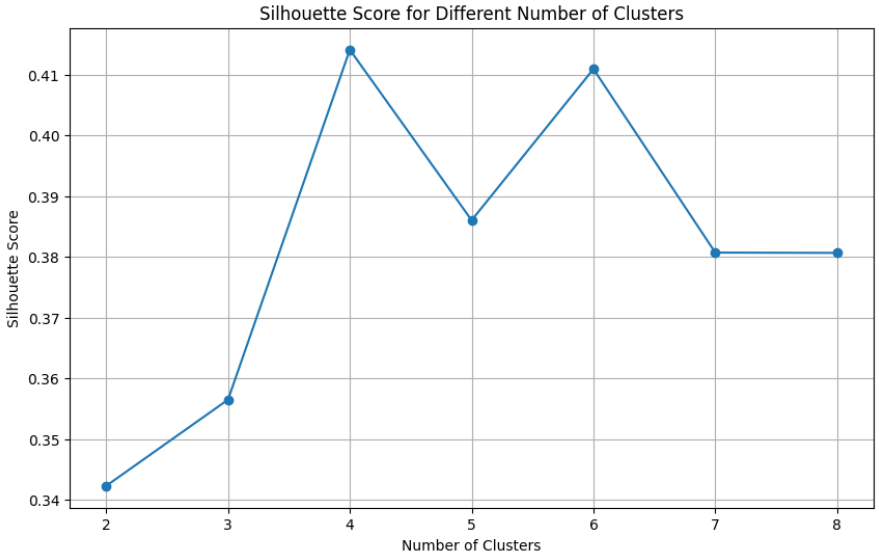


Figure 3.15: Silhouette score for different numbers of clusters - SBERT with T-sne

3.2.4.5 Using Autoencoder

In this part, we combined SBERT with an autoencoder for dimensionality reduction. The evaluation of this method is shown in the figure 3.16.

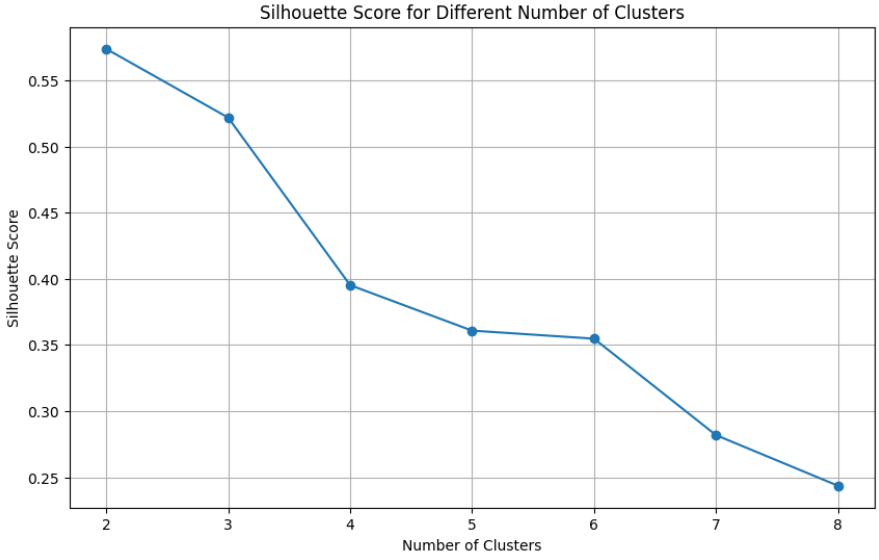


Figure 3.16: Silhouette score for different numbers of clusters - SBERT with Autoencoder

Table 3.3 illustrates the scores for different dimensionality reduction techniques applied to SBERT with K-means for the description clustering task.

Table 3.3: Silhouette scores for different dimensionality reduction's techniques with SBERT

Number of clusters	**Raw data**	**Umap**	**T-sne**	**PCA**	**Autoencoder**
2	0.161	**0.695**	0.342	**0.090**	**0.573**
3	**0.171**	0.372	0.372	0.085	0.521
4	0.113	0.393	**0.414**	0.027	0.395
5	0.062	0.398	0.386	0.021	0.360
6	0.069	0.425	0.410	0.016	0.354
7	0.064	0.430	0.380	0.044	0.281
8	0.775	0.407	0.380	0.014	0.243

3.2.5 BERTopic

In this experiment, we applied BERTopic with several dimensionality reduction techniques, followed by integration with K-means, to identify the method that most effectively captures features and delivers the highest accuracy.

BERTopic is a model designed to capture semantic information, so we will apply semantic-based preprocessing to our data before inputting it into the model.

3.2.5.1 Without dimensionality reduction

We explored the application of BERTopic without any dimensionality reduction techniques. By using the raw data, we aimed to assess the model's performance in capturing semantic features directly from the original dataset.

The highest Silhouette Score was achieved with 2 clusters, as demonstrated in the following Figure 3.17.

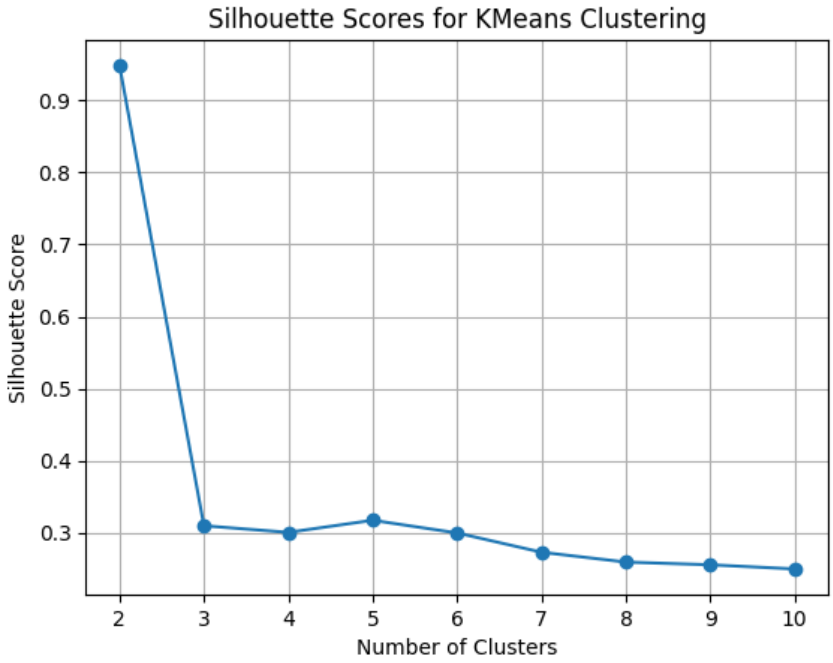


Figure 3.17: Silhouette score for different numbers of clusters - BERT without dimensionality reduction

3.2.5.2 With Umap

This time, we have investigated the use of BERTopic combined with UMAP for dimensionality reduction. The yielded results are shown in Figure 3.18.

Figure 3.18: Silhouette score for different numbers of clusters - BERT with Umap

3.2.5.3 With T-sne

In this section, we examined the application of BERTopic in conjunction with t-SNE for dimensionality reduction. By utilizing t-SNE, we aimed to enhance the model's performance in capturing semantic features by projecting the high-dimensional data into a lower-dimensional space while preserving its important structures, as demonstrated in the following Figure 3.19.

The best silhouette score was captured for a number of clusters set at 10.

Figure 3.19: Silhouette score for different numbers of clusters - BERT with T-sne

3.2.5.4 With PCA

In this section, we explored the application of BERTopic in conjunction with PCA for dimensionality reduction. By employing PCA, our goal was to improve the model's ability to capture semantic features by reducing the dimensionality of the data while maintaining its critical structures, as illustrated in the following Figure 3.20.

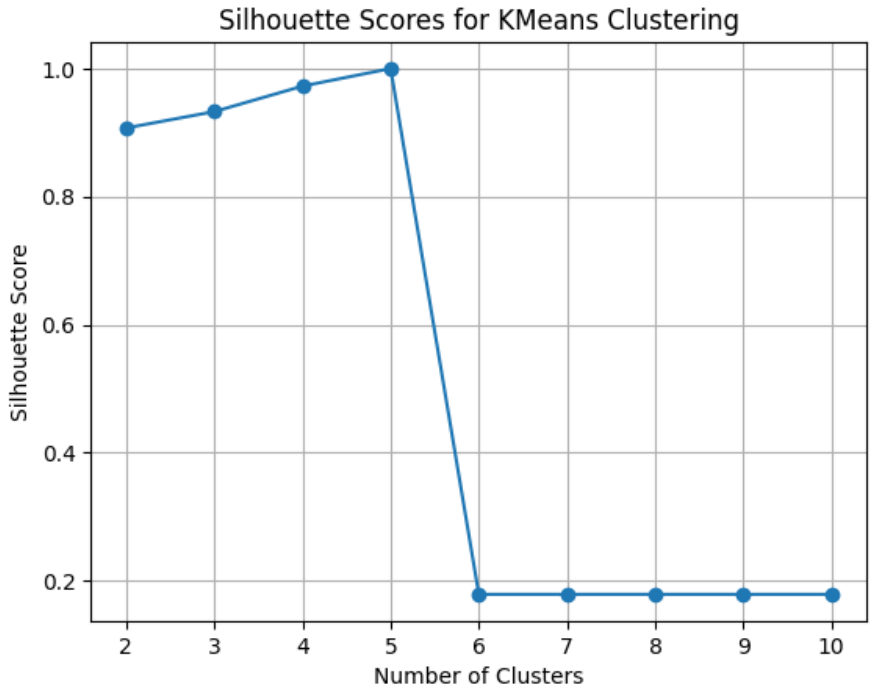


Figure 3.20: Silhouette score for different numbers of clusters - BERT with PCA

3.2.5.5 With Autoencoder

Finally, we tested BERTopic with an autoencoder for dimensionality reduction, aiming to enhance semantic feature extraction.

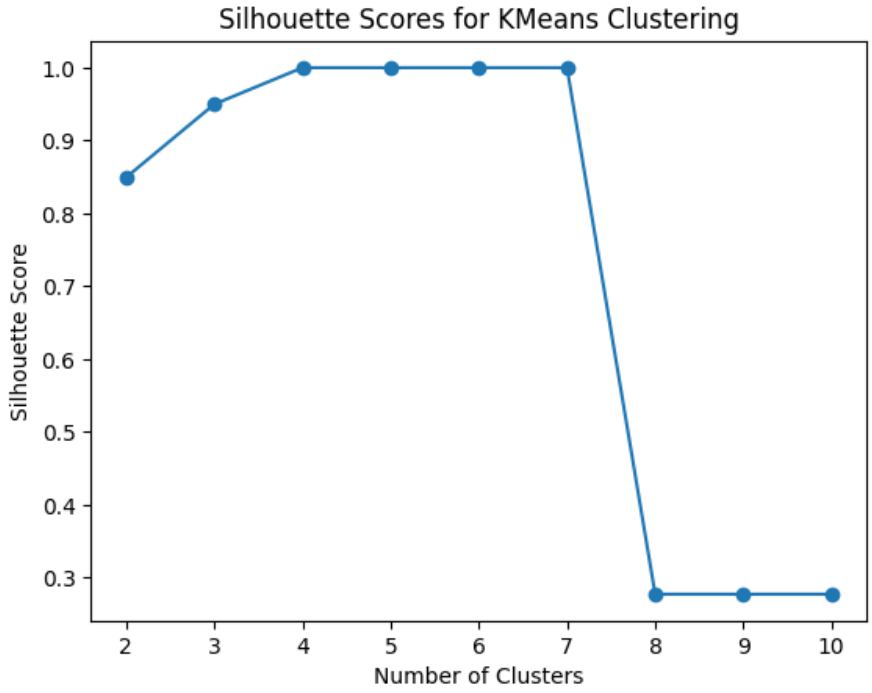


Figure 3.21: Silhouette score for different numbers of clusters - BERT with Autoencoder

This evaluation sheds light on the effectiveness of this approach for predictive maintenance. If we consider the best Silhouette Score different from 1, a number of clusters of 3 is the best fit in our clustering task with a score of 0.96, as demonstrated in the following figure 3.21.

In Table 3.4, the scores for various dimensionality reduction approaches applied to BERTopic with K-means for description clustering are displayed.

Table 3.4: Silhouette scores for different dimensionality reduction's techniques with BERT

Number of clusters	**Raw data**	**Umap**	**T-sne**	**PCA**	**Autoencoder**
2	**0.948**	**0.964**	0.896	0.907	0.824
3	0.309	0.314	0.691	0.932	**0.967**
4	0.300	0.301	0.696	**0.972**	1.0
5	0.317	0.324	0.767	1.0	1.0
6	0.299	0.297	0.859	0.178	1.0
7	0.273	0.272	0.887	0.178	1.0
8	0.259	0.265	0.906	0.178	0.174
9	0.255	0.251	0.889	0.178	0.174
10	0.249	0.262	**0.912**	0.178	0.174

3.3 Bayesian Network Model

3.3.1 Data Assignment and Preprocessing for Bayesian Network

Once the text data is clustered, we assign each description to its corresponding cluster and label it as a state. Additionally, we compute the time since the last intervention for each record by calculating the difference between the current and previous intervention dates. This time interval is discretized into meaningful categories such as '<1 week', '1 week - 1 month', '1-3 months', etc. This discretization aids in simplifying the temporal data and making it more suitable for probabilistic modeling. Each record is thus annotated with its current state, next state (shifted by one time step), and the discretized time interval since the last intervention.

3.3.2 Bayesian Network Creation and Fitting

The core of our predictive modeling involves creating a Bayesian Network, which is a probabilistic graphical model that represents a set of variables and their conditional dependencies

via a DAG. The network structure is defined with nodes representing the current state, next state, and the time since the last intervention, and edges indicating the dependencies between these variables. We fit this model using the Maximum Likelihood Estimator provided by the pgmpy library, which learns the CPDs from the data. This fitted model can then be used to infer future states based on current observations and historical data.

3.3.3 Visualization and Conditional Probability Distributions

Visualization of the Bayesian Network is performed using the networkx library, providing a clear graphical representation of the model's structure as illustrated in Figure 3.22. This visualization helps in understanding the dependencies and interactions between different states and time intervals. Additionally, we display the CPDs for each node, which detail the probability distributions of each state given its parents. These CPDs are crucial for interpreting the probabilistic relationships captured by the model, offering insights into the likelihood of transitions between different states.

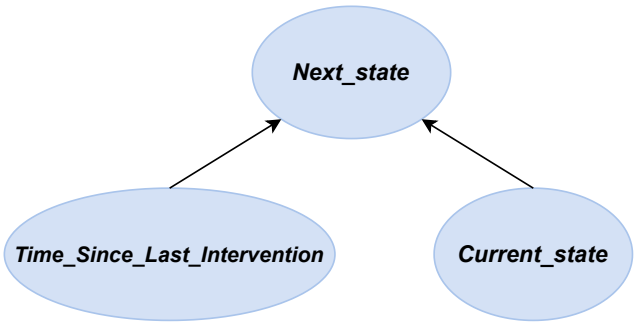


Figure 3.22: Bayesian Network diagram

3.3.4 Application and Prediction

The final part of our analysis involves applying the trained models to predict the next state for new data points. Given a new maintenance description and its date, we preprocess the text, predict its cluster, and use the Bayesian Network to forecast the subsequent state. This predictive capability is invaluable for anticipating future incidents based on historical patterns, aiding in proactive maintenance and decision-making. Additionally, we incorporate a critical state defined by the clusters analysis, where the system is deemed at significant risk of failure. We set a threshold probability in the CPD of the next state, specifically for $State\_2$. When this threshold is reached during prediction, proactive measures are automatically triggered to

avert potential failures. *State*_0 is determined as a slight breakdown, *State*_1 as an average breakdown, and *State*_2 as a critical failure. The entire workflow, from data loading to state prediction, is designed to be robust and scalable, ensuring that our approach can handle diverse and evolving datasets effectively.

3.3.5 Model Evaluation and Cross-Validation

To ensure the robustness and reliability of our Bayesian Network model, we perform rigorous evaluation using log-likelihood scoring and cross-validation techniques. This process involves several critical steps aimed at quantifying the model's predictive performance and its generalizability to unseen data.

First, we calculate the log-likelihood of the model, a metric that measures the probability of the observed data given the model. This is achieved using the log_likelihood_score function from the pgmpy library. A higher log-likelihood score indicates that the model better fits the data, reflecting its capability to capture the underlying probabilistic structure accurately.

Next, we assess the prediction accuracy of the model. This involves using the VariableElimination inference algorithm to predict the next state based on the current state and the time since the last intervention. By comparing the predicted states to the actual observed states, we can calculate the proportion of correct predictions, providing a direct measure of the model's accuracy. This step ensures that our model is not only theoretically sound but also practically useful in making accurate predictions.

To further validate the model, we perform cross-validation using the K-Fold technique with five splits. Cross-validation involves partitioning the dataset into multiple folds, training the model on a subset of these folds (training set), and testing it on the remaining fold (test set). This process is repeated multiple times, each time with a different fold as the test set. By doing so, we mitigate the risk of overfitting and obtain a more reliable estimate of the model's performance. Specifically, we calculate the average log-likelihood and prediction accuracy across all folds, providing a comprehensive evaluation of the model's effectiveness.

The results from the cross-validation process, including the average log-likelihood and accuracy, offer critical insights into the model's performance. These metrics help us understand the model's strengths and weaknesses, guiding further refinements and ensuring that our Bayesian Network is robust, accurate, and generalizable. This rigorous evaluation framework is inte-

gral to establishing the credibility of our predictive model and its potential applications in real-world scenarios.

3.3.6 Results

In this section, we present the outcomes of applying Bayesian Network models combined with text clustering techniques, specifically LDA and BERTopic, to predict state transitions within our dataset. We analyze the CPDs for various states and evaluate the models' predictive accuracy through metrics such as confusion matrices and log-likelihood scores. These evaluations provide insights into the models' effectiveness and highlight areas for potential improvement.

3.3.6.1 Bayesian Network with LDA

The figure 3.23 shows the results of bayesian network model with a text clustering using LDA.

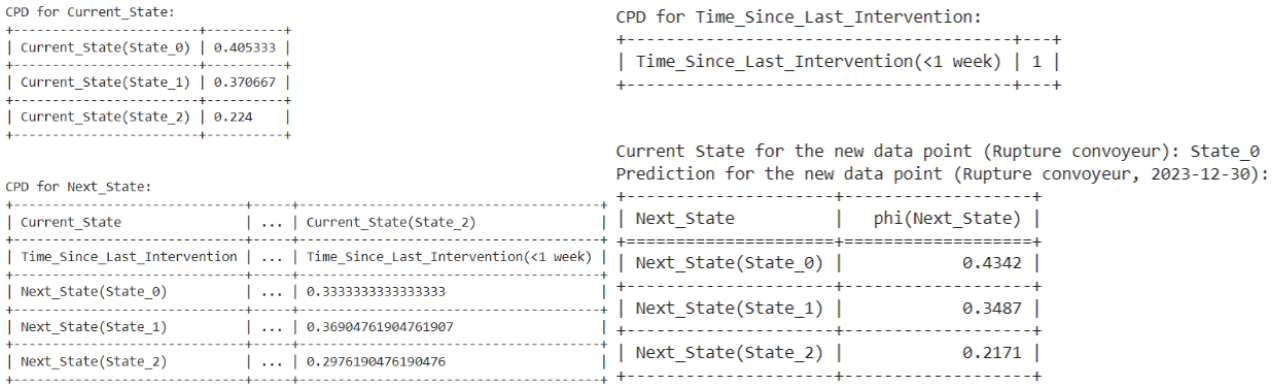


Figure 3.23: Bayesian Network with LDA

The CPD for *Current_State* shows the likelihood of the system being in each of the three states without considering other variables. This provides a baseline understanding of the system's state distribution, indicating which states are more or less common.

The CPD for *Time_Since_Last_Intervention* shows that this variable is deterministic in the current model, always being treated as less than a week. This could reflect the nature of the dataset or a modeling simplification, impacting the interpretation of the model's predictions.

The CPD for *Next_State* given *Current_State* and *Time_Since_Last_Intervention* details the probabilities of the system transitioning to each state based on its current state and the elapsed time since the last intervention. For example, when the current state is *State_2*

and less than a week has passed since the last intervention, the system is most likely to transition to *State*_1 and least likely to remain in *State*_2. This helps in understanding the system's behavior over time and can inform maintenance strategies.

For the new data point described as "Rupture convoyeur" with the date "2023-12-30", the model classifies the current state as *State*_0. Based on this state and considering the time since the last intervention as less than a week, the model predicts the probabilities for transitioning to the next state. The system is most likely to remain in *State*_0, with significant chances of transitioning to *State*_1.

The evaluation results show that the model has limited predictive accuracy, with an overall accuracy of 35.73%. The confusion matrix A reveals that while the model can somewhat predict *State*_0, it frequently misclassifies *State*_1 and *State*_2, indicating poor performance in distinguishing between these states. The log-likelihood score of -161.98 further suggests that the model's predictions do not closely match the observed data, highlighting a need for further refinement and improvement.

$$A = \begin{bmatrix} 99 & 52 & 0 \\ 105 & 35 & 0 \\ 53 & 31 & 0 \end{bmatrix}$$

3.3.6.2 Bayesian Network with BERTopic

In this section, we present the results of utilizing Bayesian Network with BERTopic for predictive maintenance, employing three different dimensionality reduction techniques: autoencoder, PCA, and no dimensionality reduction. Remarkably, all three approaches yielded identical results, as illustrated in the figure 3.24.

```
CPD for Current_State:
+-------------------------+-----------+
| Current_State(State_0) | 0.834667  |
+-------------------------+-----------+
| Current_State(State_1) | 0.106667  |
+-------------------------+-----------+
| Current_State(State_2) | 0.0586667 |
+-------------------------+-----------+

CPD for Next_State:
+-----------------------------+-----+-----------------------------------------+
| Current_State               | ... | Current_State(State_2)                  |
+-----------------------------+-----+-----------------------------------------+
| Time_Since_Last_Intervention | ... | Time_Since_Last_Intervention(<1 week) |
+-----------------------------+-----+-----------------------------------------+
| Next_State(State_0)         | ... | 0.7727272727272727                      |
+-----------------------------+-----+-----------------------------------------+
| Next_State(State_1)         | ... | 0.09090909090909091                     |
+-----------------------------+-----+-----------------------------------------+
| Next_State(State_2)         | ... | 0.13636363636363635                     |
+-----------------------------+-----+-----------------------------------------+

CPD for Time_Since_Last_Intervention:
+---------------------------------------+---+
| Time_Since_Last_Intervention(<1 week) | 1 |
+---------------------------------------+---+


1/1 [==============================] - 0s 21ms/step
Current State for the new data point (Rupture convoyeur): State_0
Prediction for the new data point (Rupture convoyeur, 2023-12-30):
+---------------------+--------------------+
| Next_State          |   phi(Next_State) |
+=====================+====================+
| Next_State(State_0) |             0.8466 |
+---------------------+--------------------+
| Next_State(State_1) |             0.0990 |
+---------------------+--------------------+
| Next_State(State_2) |             0.0543 |
+---------------------+--------------------+
```

Figure 3.24: Bayesian Network with BERTopic

The CPD for $Current\_State$ indicates that the system predominantly exists in $State\_0$, with much lower probabilities for being in $State\_1$ or $State\_2$. This suggests that the system is most often found in $State\_0$ under normal circumstances.

The CPD for $Time\_Since\_Last\_Intervention$ shows that this variable is deterministic, always considered to be less than a week. This simplifies the model by focusing on short-term transitions.

The CPD for $Next\_State$, given the current state and the time since the last intervention, shows a high probability that the system will remain in $State\_0$ if it is currently in $State\_2$ and less than a week has passed since the last intervention. This highlights a strong tendency for the system to stay in or return to $State\_0$.

For the new data point described as "Rupture convoyeur" with the date "2023-12-30", the model classifies the current state as $State\_0$. Given this state and considering that less than a week has passed since the last intervention, the model predicts a high probability that the system will remain in $State\_0$. There is a much lower probability of transitioning to $State\_1$ or $State\_2$.

In summary, the CPD tables reveal that the system is most often in $State\_0$ and tends to stay in or return to this state. For new data, the model predicts a high likelihood of remaining in $State\_0$, reflecting the patterns learned from the historical data. These insights are useful for understanding the system's behavior and for planning maintenance interventions.

The evaluation results show that the model performs exceptionally well in predicting $State\_0$, achieving a high accuracy of 83.2%. However, it fails entirely to predict $State\_1$ and $State\_2$, as indicated by the confusion matrix B, where all instances of these states are incorrectly classified as $State\_0$. The log-likelihood score being -inf further underscores the model's inability to capture the underlying patterns in the data, suggesting significant deficiencies in its predictive capabilities beyond $State\_0$. Overall, while effective for one state, the model requires substantial improvement to generalize across all states accurately.

$$B = \begin{bmatrix} 312 & 0 & 0 \\ 40 & 0 & 0 \\ 23 & 0 & 0 \end{bmatrix}$$

3.4 Hidden Markov Model Application

In this section, we use a Hidden Markov Model (HMM) to model and predict state transitions in our dataset. An HMM is suitable for sequential data with hidden states, as it assumes the system follows a Markov process.

We start by converting the *Current_State* column to numeric values, required by the hmmlearn library. Sequences from this column are then used to train the HMM. The model is designed to identify transition probabilities between three hidden states, based on preliminary analysis indicating this captures the system's dynamics. The training data is reshaped to estimate the transition and emission probabilities.

The model's performance is evaluated using prediction accuracy, which compares the HMM's predicted states against the actual states to determine the proportion of correct predictions. However, the results show an accuracy of just 10.7%, highlighting the need for significant improvement in predicting state transitions accurately.

3.5 Comparison and discussion

Based on the experiments conducted, we observed that the highest Silhouette Scores were achieved with LDA using an autoencoder for dimensionality reduction and with BERTopic both without dimensionality reduction and with dimensionality reduction using an autoencoder and PCA, as shown in Table 3.5. Despite the slight differences in Silhouette Scores, the integration of LDA results with the Bayesian network model led to poor performance. Similarly, the BERTopic model did not show significant improvement when integrated with the Bayesian network model. This is likely due to the limited amount of data, which caused overfitting and resulted in weak performance in the Bayesian model.

Method	**Dimensionality Reduction**	**Silhouette Score**
LDA	Autoencoder	0.886
BERTopic	None	0.948
BERTopic	Autoencoder	0.967
BERTopic	PCA	0.972

Table 3.5: Silhouette Scores for Different Methods and Dimensionality Reduction Techniques

Given the significantly higher silhouette scores obtained from BERTopic when utilizing autoencoder and PCA for dimensionality reduction, our preference leans towards autoencoder in our approach for the following reasons:

- Autoencoders have the capability to learn intricate non-linear relationships within data, thereby potentially enhancing their effectiveness in capturing complex patterns present in text embeddings.
- Autoencoders can learn hierarchical representations of data, which may facilitate the extraction of more meaningful features essential for clustering tasks, compared to PCA.
- While PCA has a slightly higher silhouette score, the difference is not significant (1% difference). In practice, this small difference might not lead to a noticeable change in clustering quality.
- PCA primarily aims to maximize variance and may overlook critical features necessary for clustering, particularly when the underlying data exhibits non-linear structures that are better handled by autoencoders.

3.6 Conclusion

In this chapter, we conducted an experimental study using various tools and techniques for predictive maintenance. We utilized Google Colab, Python, and SQL for computational tasks and explored multiple text clustering techniques, including TF-IDF, CNN, LDA, SBERT, and BERTopic, with and without dimensionality reduction methods.

Furthermore, we developed and evaluated a Bayesian Network model and applied a HMM for predictive maintenance. Our comparative analysis and discussion highlighted the strengths and limitations of some approaches, providing insights into their applicability in real-world predictive maintenance scenarios.

Overall, this chapter demonstrated the effectiveness of various text clustering and probabilistic modeling techniques in predictive maintenance. The results underscore the importance of choosing the right methodology based on the specific requirements and constraints of the maintenance task at hand.

In the next chapter, we will present the general conclusion of the thesis by discussing our contribution in the problem of predictive maintenance and the future perspective.

General Conclusion

In the face of an increasingly competitive industrial landscape, the pursuit of operational excellence through effective maintenance management has become paramount. This thesis has demonstrated the significant potential of leveraging Industry 4.0 advancements, particularly the integration of data acquisition and analytics, to transform maintenance strategies from reactive to predictive. The crux of this transformation lies in the deployment of Business Intelligence (BI) systems, which convert raw data into actionable insights, thereby facilitating informed decision-making and strategic planning.

The core methodology adopted in this research involved advanced data preprocessing, clustering, and predictive modeling techniques. By refining textual descriptions of maintenance interventions and employing a sophisticated clustering approach, we were able to uncover critical insights into equipment degradation patterns. This process was enhanced by the use of BERTopic, an autoencoder, and K-means clustering, which collectively enabled the creation of dense textual embeddings, dimensionality reduction, and accurate clustering of intervention descriptions.

The development of a Bayesian network-based predictive model marked a significant milestone in our research. By incorporating the clusters derived from our preprocessing techniques, the model proficiently forecasted future degradation states and maintenance requirements, thereby enabling proactive maintenance scheduling. This not only enhances operational reliability but also optimizes resource allocation and minimizes downtime.

However, this study is not without its limitations. One significant challenge encountered was the small amount of data available, which led to overfitting in the clustering method. Additionally, while the use of an autoencoder improved the Bayesian model's score, it also resulted in slightly imbalanced clusters. When the autoencoder was not used, the clusters were balanced, but the model still suffered from overfitting due to the limited training data.

Looking forward, several recommendations can be made for future research. Enhancing data quality and increasing the volume of data used for training the models will likely mitigate the overfitting issue and lead to more robust and generalizable results.

In conclusion, this dissertation has illustrated the transformative impact of predictive maintenance powered by advanced data analytics and machine learning. The integration of these technologies into maintenance management practices can significantly reduce the inefficiencies and high costs associated with traditional maintenance approaches. By anticipating and mitigating potential failures, industries can ensure timely and precise maintenance actions, thereby maximizing equipment uptime, production efficiency, and overall profitability. This research not only contributes to the academic discourse on predictive maintenance but also provides a practical framework for industries aiming to enhance their operational excellence in the era of Industry 4.0.

Bibliography

[1] A. Popovič, R. Hackney, P. S. Coelho, and J. Jaklič, "Towards business intelligence systems success: Effects of maturity and culture on analytical decision making," *Decision support systems*, vol. 54, no. 1, pp. 729–739, 2012.

[2] W. Yeoh and A. Koronios, "Critical success factors for business intelligence systems," *Journal of computer information systems*, vol. 50, no. 3, pp. 23–32, 2010.

[3] R. T. Ng, P. C. Arocena, D. Barbosa, and G. Carenini, *Perspectives on Business Intelligence*. Morgan & Claypool Publishers, 2013.

[4] K. Zhong, T. Jackson, A. West, and G. Cosma, "Natural language processing approaches in industrial maintenance: A systematic literature review," *Procedia Computer Science*, vol. 232, pp. 2082–2097, 2024.

[5] F. Longo, L. Nicoletti, and A. Padovano, "Smart operators in industry 4.0: A human-centered approach to enhance operators' capabilities and competencies within the new smart factory context," *Computers & industrial engineering*, vol. 113, pp. 144–159, 2017.

[6] A. Sahli, R. Evans, and A. Manohar, "Predictive maintenance in industry 4.0: Current themes," *Procedia CIRP*, vol. 104, pp. 1948–1953, 2021.

[7] Y. Wang, C. Deng, J. Wu, Y. Wang, and Y. Xiong, "A corrective maintenance scheme for engineering equipment," *Engineering Failure Analysis*, vol. 36, pp. 269–283, 2014.

[8] B. De Jonge and P. A. Scarf, "A review on maintenance optimization," *European journal of operational research*, vol. 285, no. 3, pp. 805–824, 2020.

[9] Y. Zhang, *New advances in machine learning*. BoD–Books on Demand, 2010.

[10] B. Abdelbari, "Healthcare big data warehouse integration," 2023.

[11] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[12] S. Raschka and V. Mirjalili, "Python machine learning packt publishing ltd," 2015.

[13] M. Mohammed, M. B. Khan, and E. B. M. Bashier, *Machine learning: algorithms and applications*. Crc Press, 2016.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "t-distributed stochastic neighbor embedding (t-sne)," 2024. Accessed: 2024-06-13.

[15] 2024. Accessed: 2024-06-13.

[16] X.-S. Yang, *Introduction to algorithms for data mining and machine learning*. Academic press, 2019.

[17] E. Alpaydın, "Introduction to machine learning," 2010.

[18] A. Ankan and A. Panda, *Mastering probabilistic graphical models using python*. Packt Publishing Ltd, 2015.

[19] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, no. 1, p. 6086, 2024.

[20] D. Sarkar, *Text analytics with Python: a practitioner's guide to natural language processing*. Springer, 2019.

[21] M. Shutaywi and N. N. Kachouie, "Silhouette analysis for performance evaluation in machine learning with applications to clustering," *Entropy*, vol. 23, no. 6, 2021.

[22] P. Goyal, S. Pandey, and K. Jain, "Deep learning for natural language processing," *New York: Apress*, 2018.

[23] T. Beysolow, "Applied natural language processing with python," 2018.

[24] T. Beysolow, "Applied natural language processing with python," 2018.

[25] H. Hapke, C. Howard, and H. Lane, *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Simon and Schuster, 2019.

[26] S. Vajjala, B. Majumder, A. Gupta, and H. Surana, *Practical natural language processing: a comprehensive guide to building real-world NLP systems*. O'Reilly Media, 2020.

[27] H. Zhou, T. A. L. Genez, A. Brintrup, and A. K. Parlikad, "A hybrid-learning decomposition algorithm for competing risk identification within fleets of complex engineering systems," *Reliability Engineering & System Safety*, vol. 217, p. 107992, 2022.

[28] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[29] H. Jelodar, Y. Wang, C. Yuan, and X. Feng, "Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey," 11 2017.

[30] T. Beysolow, "Applied natural language processing with python," 2018.

[31] A. Abuzayed and H. Al-Khalifa, "Bert for arabic topic modeling: An experimental study on bertopic technique," *Procedia computer science*, vol. 189, pp. 191–194, 2021.

[32] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[33] T. Abbasi, K. H. Lim, N. Rosli, I. Ismail, and R. Ibrahim, "Development of predictive maintenance interface using multiple linear regression," in *2018 International Conference on Intelligent and Advanced System (ICIAS)*, pp. 1–5, IEEE, 2018.

[34] Q. Wang, S. Bu, and Z. He, "Achieving predictive and proactive maintenance for high-speed railway power equipment with lstm-rnn," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6509–6517, 2020.

[35] J. Tang, D. You, F. Li, and Y. Cheng, "Development of predictive maintenance system for nuclear power turbine unit," in *2023 2nd International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*, pp. 1–4, IEEE, 2023.

[36] S. M. R. Naqvi, M. Ghufran, C. Varnier, J.-M. Nicod, K. Javed, and N. Zerhouni, "Unlocking maintenance insights in industrial text through semantic search," *Computers in Industry*, vol. 157, p. 104083, 2024.

[37] F. Ansari, L. Kohl, J. Giner, and H. Meier, "Text mining for ai enhanced failure detection and availability optimization in production systems," *CIRP Annals*, vol. 70, no. 1, pp. 373–376, 2021.

[38] R. Sala, F. Pirola, G. Pezzotta, and S. Cavalieri, "Nlp-based insights discovery for industrial asset and service improvement: an analysis of maintenance reports," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 522–527, 2022.

[39] F. Ansari, "Cost-based text understanding to improve maintenance knowledge intelligence in manufacturing enterprises," *Computers & Industrial Engineering*, vol. 141, p. 106319, 2020.

[40] Z. Liang and A. Parlikad, "A markovian model for power transformer maintenance," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 175–182, 2018.

[41] Z. Chen, Y. Li, T. Xia, and E. Pan, "Hidden markov model with auto-correlated observations for remaining useful life prediction and optimal maintenance policy," *Reliability Engineering & System Safety*, vol. 184, pp. 123–136, 2019.

[42] N. Burmeister, R. D. Frederiksen, H. Esben, P. Nielsen, *et al.*, "Exploration of production data for predictive maintenance of industrial equipment: A case study," *IEEE Access*, 2023.

[43] W. Zhong, J. Cai, Y. Song, T. Liang, J. Zhang, and Z. Gao, "Risk evolution of crude oil pipeline under periodic maintenance based on dynamic bayesian network," *Journal of Loss Prevention in the Process Industries*, vol. 87, p. 105229, 2024.

[44] Z. Yang, P. Baraldi, and E. Zio, "A novel method for maintenance record clustering and its application to a case study of maintenance optimization," *Reliability Engineering & System Safety*, vol. 203, p. 107103, 2020.

[45] A. Ahadh, G. V. Binish, and R. Srinivasan, "Text mining of accident reports using semi-supervised keyword extraction and topic modeling," *Process safety and environmental protection*, vol. 155, pp. 455–465, 2021.

[46] D. Valcamonico, P. Baraldi, E. Zio, L. Decarli, A. Crivellari, and L. La Rosa, "Combining natural language processing and bayesian networks for the probabilistic estimation of the severity of process safety events in hydrocarbon production assets," *Reliability Engineering & System Safety*, vol. 241, p. 109638, 2024.

[47] A. Rawat, "A review on python programming," *International Journal of Research in Engineering, Science and Management*, vol. 3, no. 12, pp. 8–11, 2020.

[48] S. Saabith, V. Thangarajah, and M. Fareez, "A review on python libraries and ides for data science," 11 2021.

[49] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[50] G. Hackeling, *Mastering Machine Learning with scikit-learn.* Packt Publishing Ltd, 2017.

[51] S. Saabith, T. Vinothraj, and M. Fareez, "A review on python libraries and ides for data science," *Int. J. Res. Eng. Sci*, vol. 9, no. 11, pp. 36–53, 2021.

[52] "SpaCy." https://spacy.io/. Accessed: 2024-06-09.

[53] "Natural Language Toolkit (NLTK)." https://www.nltk.org/. Accessed: 2024-06-18.

[54] "SQL." https://www.journaldunet.fr/web-tech/guide-du-big-data/1203603-sql-structured-query-language-definition-traduction-et-acteurs/.

[55] "SSMS." https://learn.microsoft.com/fr-fr/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16.