

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Task Prediction and Planning Tool for Complex Engineering Tasks

Afonso Maria Rebordão Caiado de Sousa



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

Supervisor at FEUP: João Carlos Pascoal Faria

Second Supervisor at FEUP: João Pedro Carvalho Leal Mendes Moreira

Supervisor at CERN: Fernando Baltasar dos Santos Pedrosa

Second Supervisor at CERN: Rodrigo Lanza Herrero

July 20, 2023

Task Prediction and Planning Tool for Complex Engineering Tasks

Afonso Maria Rebordão Caiado de Sousa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Ana Paiva

External Examiner: Prof. Alberto Silva

Supervisor: Prof. João Carlos Pascoal Faria

Co-Supervisor: Prof. João Mendes Moreira

July 20, 2023

Resumo

A predição e o planeamento de futuras tarefas complexas de engenharia numa organização são passos essenciais para o planeamento dos recursos humanos e físicos. Isto é particularmente importante numa era de gestão de stocks 'just-in-time' e de recursos humanos qualificados escassos.

Com base na realidade do CERN, a Organização Europeia para Pesquisa Nuclear, e a sua atual plataforma de planeamento de tarefas, PLAN, o presente trabalho aborda os problemas que podem surgir durante o planeamento de tarefas futuras. Aproveitando e tirando partido dos dados históricos existentes de tarefas planeadas e executadas, pretende-se contribuir para a plataforma acima mencionada através da aplicação de algoritmos de aprendizagem computacional (machine learning). Pretende-se também provar a viabilidade de realizar e executar projetos de aprendizagem computacional em situações com uma quantidade de dados limitada no CERN.

A primeira parte da otimização tem como base a análise da similaridade de dados entre uma nova tarefa (atividade) e os dados históricos existentes. O modelo preditivo tem como alvo uma lista ordenada de atividades semelhantes para sugerir ao utilizador para a atividade atual.

A segunda e última parte do trabalho apoia-se em dados relacionados com contribuições comuns de equipas de trabalho em atividades passadas para produzir uma lista de contribuições desejáveis para a nova tarefa.

Esta dissertação acompanha o percurso percorrido para atingir estes objetivos, começando com uma pesquisa aprofundada sobre os tópicos em questão, para uma compreensão profunda do problema, o processo de desenvolvimento e todos os altos e baixos que o acompanham, culminando numa solução final baseada em técnicas como *k-Nearest Neighbors* e *Natural Language Processing*, oferecendo resultados muito satisfatórios na sugestão de itens aos utilizadores da plataforma.

Como este projeto foi realizado num cenário do mundo real, os resultados deste trabalho de dissertação contribuem para aprimorar a aplicação de software da estrutura organizacional em questão e contribuir para o progresso da sua gestão de recursos humanos e físicos, providenciando também um exemplo de adaptabilidade e trabalho em direção a uma solução bem-sucedida no campo da inteligência artificial e machine learning.

Keywords: Predição, Aprendizagem computacional, Quantidade de dados limitada, Otimização, Similaridade de dados

Abstract

Prediction and planning of future complex engineering tasks across an organization are essential for human and physical resources planning. This is particularly important in an age of 'just-in-time' stock management and scarce qualified human resources.

Based on the reality of CERN, the European Organization for Nuclear Research, and their existing task-planning platform, PLAN, the present work tackles the problems that may arise while planning future tasks. Taking advantage and leveraging existing historical records of planned and executed tasks, this Dissertation revolves around optimizing and applying different machine learning algorithms to said planning tool. It also intends to prove the viability of carrying out and executing machine learning projects in low-data situations at CERN.

The first part of the optimization is based on the analysis of data similarity between a new task (activity) and the existing historical data. The predictive model has as its target a ranked list of similar activities to suggest to the user for the current activity, so that content can be reused.

The second and final part of the work leans on data regarding the common contributions from work groups in past activities to produce a list of desirable contributions for the new task.

This dissertation expands on the journey endured to accomplish these goals, starting with a thorough research on the topics at hand, to a deep understanding of the problem at hand, the development process and all the ups and downs that came with it, leading to a final solution that uses k-Nearest Neighbor and Natural Language Processing techniques and provides very satisfying results in suggesting new items to PLAN users.

As this project was carried out in a real-world scenario, the results from this thesis will enhance the software application of the organizational structure at hand and contribute to the progress of its human and physical resources management while also providing an example of adapting and working towards a successful solution in the field of AI and machine learning.

Keywords: Prediction, Machine learning, Low-data, Optimization, Data similarity

ACM Classification: CCS - Computing Methodologies - Machine Learning - Machine Learning Algorithms, CCS - Applied Computing - Operations research - Decision analysis

Acknowledgements

First of all, I want to thank my dissertation supervisors, Prof. João Carlos Pascoal Faria and Prof. João Pedro Leal Mendes Moreira, for the continuous support, help, and availability shown throughout the project.

Secondly, I would like to thank the entirety of CERN's EN-ACE-OSS group for welcoming me with open arms for the entire duration of my internship. A special thank you to my supervisor Fernando Baltasar dos Santos Pedrosa who guided me throughout my stay, and to Rodrigo Lanza Herrero, who helped me with the more technical part of the project.

Finally, I would like to express my deepest gratitude to my family, who was always there for me for the past two years and offered me the possibility of living abroad. Special thanks to Laura and to my friends, without whom this dissertation would not have been possible.

Afonso Maria Rebordão Caiado de Sousa

“Homme libre, toujours tu chériras la mer!”

Charles Baudelaire

Contents

1	Introduction	1
1.1	Context	1
1.2	Problem Statement	2
1.3	Research Objectives	2
1.4	Research Questions	3
1.5	Document Structure	5
2	State of the Art	6
2.1	Data Preprocessing	7
2.2	Supervised Learning Techniques	8
2.2.1	Decision Trees	9
2.2.2	Random Forest	9
2.2.3	Nearest Neighbor Algorithms	10
2.2.4	Performance Evaluation	12
2.3	Unsupervised Learning Techniques	15
2.3.1	Clustering	15
2.3.2	Association Rules	18
2.3.3	Performance Evaluation	19
2.4	Natural Language Processing Techniques	21
2.4.1	Text Representation and Feature Extraction	21
2.4.2	Text Similarity Measures	22
2.5	Dealing with Low-data Situations	23
2.6	Content-Based Recommendation Systems	23
2.7	Task Management and Resource Planning	24
2.8	Machine Learning in Task Management and Resource Planning	26
2.9	Summary and Identified Gaps	27
3	Problem Identification	29
3.1	Overview of PLAN Platform	29
3.1.1	Brief Description and Purpose	30
3.1.2	Key Components and Workflow	30
3.1.3	Model and Architecture	32
3.2	Use Cases Understanding	34
3.2.1	Activity Suggestion	34
3.2.2	Contribution Suggestion	36
3.3	Summary	37

4	Solution Development	38
4.1	Data Preparation	38
4.1.1	Data Extraction	39
4.1.2	Activity Dataset	40
4.1.3	Contribution Dataset	41
4.2	Initial Approach	42
4.2.1	Clustering Methods	43
4.2.2	Supervised Learning Techniques	44
4.2.3	Evaluation Method	45
4.2.4	Performance Evaluation	46
4.2.5	Challenges and Lessons Learned	48
4.3	Final Algorithm Development and Optimization	49
4.3.1	Similar activity grouping	50
4.3.2	Suggestion Generation	52
4.3.3	Evaluation Method	57
4.3.4	Model Tuning	60
4.3.5	Final Model Performance	61
4.4	Summary	63
5	Practical Implementation and Deployment	64
5.1	API implementation and performance	64
5.2	Integration and Interface Design	67
5.3	Practical Impact	69
5.4	Maintenance and Future Improvements	69
5.5	Summary	70
6	Conclusions	72
6.1	Summary of Findings	72
6.2	Limitations and Future Work	73
6.3	Implications and Contributions	75
6.4	Final Thoughts and Conclusions	76
	References	78

List of Figures

2.1	Decision tree example for vehicle types	9
2.2	Simple KNN representation - Source [13]	11
2.3	Silhouette Plot Example with 4 clusters - Source [5]	20
2.4	Aggravated Matrix - Source [8]	25
3.1	PLAN activity process diagram	31
3.2	PLAN Home Page	32
3.3	PLAN Model Diagram	33
3.4	PLAN Architecture Diagram	34
3.5	Activity Creation Main Form	35
3.6	Contribution Creation Tab	36
4.1	PLAN Table access example	40
4.2	Activity Dataset	41
4.3	Contribution Dataset	42
4.4	Clustering Solution's Silhouette Plot	47
4.5	Activity Creation Example	53
4.6	Returned activities	53
4.7	Suggested Activity Fields	54
4.8	Suggested Contributions for Preparation Phase	55
4.9	Association Rule Example	57
4.10	Similarity Score by Number of Neighbors	60
4.11	Similarity Score by k-NN weight	61
5.1	API Request Example	65
5.2	API Response Example	66
5.3	PLAN Interface with Suggestions	68
5.4	PLAN Activity Suggested Interface	68

List of Tables

2.1	Confusion Matrix Example	13
4.1	Suggestion System Performance	62
5.1	API Performance	67

Abbreviations

AC	Accuracy
AI	Artificial Intelligence
API	Application Programming Interface
CERN	Conseil Européen pour la Recherche Nucléaire - European Organization for Nuclear Research
DBI	Davies-Bouldin Index
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FN	False Negative
FP	False Positive
FNR	False Negative Rate
FPR	False Positive Rate
GA	Genetic Algorithm
IBL	Instance-Based Learning
k-NN	K-Nearest Neighbors
LHC	Large Hadron Collider
LS	Long Shutdowns
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Networks
P	Precision
POC	Proof of Concept
RI	Rand Index
RQ	Research Question
SEN	Sensitivity
SPE	Specificity
SVM	Support Vector Machine
TF-IDF	Term-Frequency Inverse Document Frequency
TN	True Negative
TP	True Positive

Chapter 1

Introduction

This chapter serves as an introduction to the dissertation, setting the stage for the research conducted and the results obtained.

Section 1.1 provides the context of the dissertation, explaining the collaboration with CERN and the need for efficient task planning and resource allocation.

Section 1.2 presents the problem statement, identifying the main difficulties in the PLAN platform and the project's primary goal.

The research objectives are outlined in section 1.3.

Section 1.4 establishes the research questions to be answered during this work.

Lastly, the document's structure is detailed in section 1.5, guiding the reader through the subsequent chapters of the dissertation.

1.1 Context

This dissertation and project are done in close collaboration with the engineering department at CERN in Switzerland. CERN, the European Organization for Nuclear Research, is a leading research organization in Geneva. It is one of the world's largest and most respected centers for scientific research, known for its particle physics experiments.

CERN is a large organization with a diverse international community of researchers, engineers, and support staff. CERN has over 2,500 full-time employees and hosts approximately 12,000 visiting scientists and engineers each year [6]. It also encompasses a large physical site with several installations and facilities. The main campus covers an area of approximately 540 hectares (about 1,320 acres), including several large buildings and underground experimental areas.

As for any organization or company, especially for large-scale ones, task management, and resource planning are critical elements of project management and help to ensure that projects are completed on time, within budget, and to the satisfaction of stakeholders. Task management involves creating a plan for how tasks will be completed, assigning those tasks to team members, and tracking progress. Resource planning consists in identifying and allocating the necessary resources, such as personnel, equipment, and materials, to complete the project.

Research has shown that organizations that effectively manage tasks and resources have a higher rate of project completion within budget and on schedule.

Thus, CERN has developed its own task planning platform called PLAN. PLAN allows a user to create tasks that must be done, named activities. Each activity has information associated with it, like the title, the type of the activity, the location, the facilities, and the person responsible, among others. Requests can be made to other working groups for their contribution to said activity.

1.2 Problem Statement

There are two main problems or difficulties identified in the context of the PLAN platform that need to be addressed:

- **PLAN Platform Limitations:** The PLAN platform currently requires users to manually input information for activity creation. This process can be time-consuming and error-prone, leading to inefficiencies in the task management process at CERN. The risk of errors in data entry can result in inaccurate or incomplete task planning, which negatively impacts the overall efficiency of project management.
- **Difficulty in Generating Meaningful Suggestions for Task Assignments:** Developing a system that generates helpful and relevant suggestions for task assignments is a complex challenge. Suggestions must be tailor-made for the specific system they are working on, in our case, for PLAN, considering factors such as available resources, task requirements, and historical data of similar tasks. This complexity makes it challenging to create a system that provides meaningful and actionable recommendations to the users.

These two main problems lead to the overarching goal of this work and project: Developing an intelligent system to enhance macro-scale activity creation and task planning within the PLAN platform, with the use of machine learning techniques. By addressing these challenges, the proposed system aims once again to improve the efficiency, accuracy, and overall effectiveness of the task management process at CERN.

1.3 Research Objectives

The main objectives of this research and project have been established to address the challenges and limitations identified in the problem statement. These objectives are as follows:

- **Develop a machine learning-based system to generate suggestions:** The main objective is to design and implement a system that utilizes machine learning techniques to generate meaningful suggestions for task assignments. This system should take into account the available data from the PLAN platform and provide accurate and relevant recommendations to enhance the task planning process.

- **Explore the viability of machine learning in low-data situations:** Another crucial objective of this research is to investigate the potential of applying machine learning techniques to the PLAN system, given its limited historical data. This exploration will involve identifying suitable approaches and techniques for dealing with low-data scenarios, which could potentially benefit other similar applications in the field.
- **Optimize the user experience and interface design:** Enhancing the user experience of the PLAN platform is also an important objective. By incorporating the intelligent suggestion system and improving the interface design, the goal is to ensure seamless integration and ease of use for users, ultimately leading to more efficient task management and resource allocation.
- **Evaluate the performance and impact of the proposed system:** Conducting a thorough evaluation of the proposed system's performance is essential. This evaluation will focus on accuracy, efficiency, and usability, providing insights into the system's effectiveness in improving task planning and resource allocation at CERN.
- **Identify potential areas for future improvements and research:** Lastly, this research aims to identify possible directions for future research and improvements in the field of task management and resource planning using machine learning techniques. By analyzing the findings and lessons learned from this project, the goal is to contribute to the ongoing development of innovative solutions for complex task planning and resource allocation challenges.

The establishment of the main objectives leads us to a more concise definition of what questions we want to answer with the completion of this project. These will be outlined in section 1.4.

1.4 Research Questions

As we said, this dissertation essentially aims to optimize CERN's task-planning tool, PLAN, by making suggestions and predictions to improve platform interaction and help users.

In an organization as large and complex as CERN, efficient and accurate task management is crucial to ensure the success of projects and the overall productivity of the organization. Manual input of task information in the PLAN system can be time-consuming and error-prone, leading to inefficiencies and decreased productivity. It is essential to explore ways to utilize the available data from PLAN databases effectively to improve platform interaction and help PLAN users. This leads to the following research question:

RQ1: "How can we use machine learning algorithms to improve platform interaction and to provide assistance to the user?"

Another important aspect is investigating the potential for applying Machine Learning techniques to the PLAN system, given its low quantity of historical data. This poses a challenge in developing robust algorithms capable of providing accurate suggestions and predictions. Consequently, we ask:

RQ2: “Can machine learning be used to improve task management and resource planning in PLAN, considering the low quantity of historical data available?”

As a direct result of the second research question comes a logical third research question where we aim to discover which are the best machine learning techniques to utilize, if it is indeed possible to improve task management and resource planning in PLAN.

RQ3: “Which machine learning techniques are most appropriate and efficient in low-data situations”

Addressing the questions outlined in RQ1, RQ2 and RQ3 has significant implications for the field of task planning and resource allocation, as well as the broader application of machine learning in low-data situations. By improving the efficiency, productivity, and success of projects, organizations can better meet their goals and objectives. Enhancing task planning and resource allocation can lead to higher efficiency, reduced costs, and improved project and product quality.

Furthermore, the investigation into the potential for applying machine learning techniques to the PLAN system, considering its low quantity of historical data, could pave the way for similar advancements in other industries and organizations facing similar data limitations. This analysis of viability could help researchers and practitioners better understand the possibilities and limitations of machine learning in low-data situations, leading to the development of more effective techniques and tools for these scenarios.

Addressing these challenges in the context of CERN has the potential to extend beyond the field of task planning and resource allocation and positively impact society as a whole. Improving task planning and resource allocation can contribute to the efficiency, quality, and success of various industries such as construction, manufacturing, and healthcare. As a result, organizations like CERN can increase their efficiency and productivity, leading to improved products, services, and outcomes for society. This, in turn, can contribute to economic growth, job creation, and overall quality of life.

Lastly, effective task planning and resource allocation can also lead to better decision-making support for users in situations where they are uncertain about how to complete tasks or whose contributions to request. The system can provide intelligent suggestions by leveraging machine learning techniques, helping users make informed decisions, and improving overall task management. This not only enhances the efficiency and productivity of the organization but also reduces the time spent on manual decision-making and increases the likelihood of successful task completion. Exploring the viability and the specificities of applying machine learning in low-data situations, as outlined in RQ1, RQ2 and RQ3, plays a crucial role in unlocking these benefits and driving advancements in the field of task planning and resource allocation.

1.5 Document Structure

This dissertation project is informed by the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, a widely accepted process model that, although initially created for data mining tasks, is commonly employed across different types of data analysis and machine learning projects. The structure of the project and the dissertation mirrors the stages of the CRISP-DM, commencing from the Problem Identification chapter.

This dissertation is organized into six chapters, including the introduction. The chapters are structured as follows:

1. **Introduction:** The current chapter, as we've seen, provides the context, motivation, problem statement, and research objectives of the dissertation.
 2. **State of the Art:** Chapter 2 presents a review of the relevant literature, covering data preprocessing, supervised and unsupervised learning techniques, natural language processing, dealing with low-data situations, content-based recommendation systems, and machine learning in task management and resource planning. The chapter concludes with a summary and identification of gaps in the literature.
 3. **Problem Identification:** Chapter 3 corresponds to the "Business Understanding" and "Data Understanding" stages of CRISP-DM. It dives deeper into the problem at hand, exploring the task planning platform, PLAN, and understanding both of the use cases that will be addressed in this work.
 4. **Solution Development:** Chapter 4 aligns with the "Data Preparation", "Modeling", and "Evaluation" stages of CRISP-DM. It details the journey and approach taken to get to a final solution to the problem, including data preparation, our initial approach, and final algorithm development and optimization. This chapter also discusses initial results, performance evaluation, and the challenges and lessons learned from the process.
 5. **Practical Implementation and Deployment:** Chapter 5 corresponds to the "Deployment" stage of CRISP-DM. It focuses on the implementation of the API for CERN deployment, on the integration of the developed system with the PLAN platform, practical impact, and maintenance and future improvements.
 6. **Conclusion:** Chapter 6 summarizes the findings, implications, and contributions of the research. It also discusses the limitations of the study and outlines potential avenues for future work, ending the thesis with some final thoughts and conclusions.
- **References:** A list of references cited throughout the dissertation is provided at the end of the document.

Through the lens of the CRISP-DM methodology, this project ensures a systematic and comprehensive approach to the problem at hand, showcasing a balance between theoretical understanding and practical application in the field of machine learning.

Chapter 2

State of the Art

This chapter provides a comprehensive review of the state of the art in Machine Learning, focusing on its application in task management and resource planning, a crucial component of the task prediction system developed for this thesis. Each section offers crucial insights into the techniques and challenges encountered during the development of our algorithm.

Section 2.1 explains the process that data must go through before being ready to be used in any kind of machine learning algorithm.

Section 2.2 dives into supervised learning techniques, emphasizing their utility in data classification and prediction. These techniques, including Decision Trees, Random Forests, and Nearest Neighbor Algorithms, have foundational implications in task prediction algorithms, whose essence lies in classifying and predicting tasks based on certain attributes.

Section 2.3 introduces unsupervised learning techniques such as clustering methods, and association rules. These techniques are instrumental in identifying inherent groupings and associations within data, a capability crucial for the efficient operation of our algorithm.

Section 2.4 presents Natural Language Processing (NLP) techniques such as text representation, feature extraction, and text similarity measures. These techniques were crucial for enabling our algorithm to effectively analyze and understand free text fields.

Section 2.5 offers insights into dealing with low-data situations, a significant challenge in this research. The strategies explored are detailed here as solutions to enhance the performance of the algorithm despite limited data.

Section 2.6 provides an overview of recommendation systems, focusing on content-based recommendation systems. These systems are explored for their potential to accurately suggest tasks based on the items to suggest and on the user, thereby forming an integral part of the methodology adopted in this thesis.

Sections 2.7 and 2.8 provide the necessary context for the problem addressed by the algorithm. They offer a thorough review of task management and resource planning, discussing existing strategies, frameworks, and their limitations. Furthermore, they explore the current state of machine learning applications in these domains, thus positioning our work within a broader research landscape.

Finally, Section 2.9 synthesizes the discussions from the previous sections, summarizing the key points and identifying gaps in the existing literature. This section sets the stage for presenting our work, which aims to address those gaps.

2.1 Data Preprocessing

Before the application of any machine learning techniques, data must undergo a crucial process known as data preprocessing. This is an essential step that primarily involves cleaning the data, handling missing values, and transforming data into a suitable form that can be later consumed by machine learning algorithms.

Data cleaning, which involves removing inconsistencies and errors in the data, is an integral part of data preprocessing. This is because the quality of the data significantly affects the performance of machine learning models. Furthermore, noisy, inconsistent, or missing data can lead to misleading outcomes, affecting the overall validity of the results [22]. Some common data cleaning techniques include:

- **Noise Identification and Reduction:** Noise in the data can distort the underlying patterns and relationships, leading to wrong conclusions. Techniques like smoothing, aggregation, binning, and regression can be employed to identify and reduce noise [22].
- **Outlier Detection:** Outliers, or extreme values, can significantly skew the analysis. Statistical methods like the Z-score, IQR (Interquartile Range), and robust statistical metrics, alongside data mining techniques, can be used to detect and manage outliers.
- **Duplicate Removal:** Duplicate data entries can artificially inflate the apparent significance of certain patterns. Various methods exist to detect and remove duplicates, depending on the nature and complexity of the data.
- **Handling Inconsistent Data:** Inconsistencies may arise due to various reasons, including data integration from disparate sources, human errors, or system glitches. Techniques like data transformation, entity resolution, and the use of rule-based systems can help to rectify inconsistencies.

Another fundamental aspect of data preprocessing is dealing with missing values. Missing data is a common occurrence and can be due to various reasons, including errors during data collection or issues during data transfer. There are several techniques available to handle missing values before applying any machine learning algorithm, which include:

- **Removal:** This technique consists in simply removing the entire data point whenever it contains a missing value.
- **Imputation:** Imputation replaces the missing values with a certain value. Be it the median, the mean, or any other values determined by the developer.

- **Adding a missing value indicator:** This method involves creating a new feature that determines whether or not there is a missing value. This could be useful in situations where the presence of missing values is informative.

Moreover, the encoding of categorical variables is another essential step in data preprocessing. Machine learning models generally require numerical input; hence categorical variables in the data need to be encoded in numerical form. Two main encoding techniques were considered and tested:

- **One-hot encoding:** This technique converts every existing category into a new column and assigns a 0 or a 1 to the column.
- **Label encoding:** This method replaces each category with a different integer value.

The chosen data preprocessing steps heavily depend on the specificities of the data at hand and the subsequent machine learning techniques to be applied. The techniques outlined above have been widely used in the field of machine learning and have shown their effectiveness in different scenarios. It is important to carefully choose the appropriate preprocessing techniques as they have a direct impact on the performance and interpretability of the resulting models.

Building on these foundational data preprocessing steps, we transition into the domain of supervised learning. The following section (Section 2.2) explores various techniques within the realm of supervised learning, such as Decision Trees, Random Forest, and Nearest Neighbors algorithms. These techniques, each with its unique strengths, will be explored in detail, offering insights into how they can contribute to our broader goal of task management and resource planning.

2.2 Supervised Learning Techniques

Supervised learning, one of the most commonly used branches of machine learning, is a method where an algorithm learns from labeled training data, allowing it to make accurate predictions. It is renowned for its ability to discover complex data patterns and has been applied in various domains with considerable success [21].

Supervised learning algorithms use labeled data, that is, input-output pairs, during training with the aim of learning a function that can accurately map inputs to their corresponding outputs. Once this function is accurately defined, the algorithm can predict outputs for new, unseen inputs.

This category of machine learning algorithms typically falls into two main categories: regression, for predicting continuous output values, and classification, for predicting discrete output categories. Widely used supervised learning algorithms include Decision Trees, Random Forests, and Nearest Neighbor algorithms, each offering unique advantages and mechanisms [35].

Performance evaluation is also an important aspect of supervised learning. Metrics such as accuracy, precision, recall, and F1-score are frequently used to evaluate the performance and efficiency of these learning algorithms.

In the following subsections, we will dive into further detail on some of the mentioned supervised learning techniques.

2.2.1 Decision Trees

A decision tree is a type of algorithm used in machine learning and statistics for both classification and regression tasks. It is a tree-based model that starts with a single node, called the root node, and branches out into multiple decision points, called internal nodes, and ends in leaf nodes, which represent the final decision or prediction.

The tree is built by using a set of rules, called decision rules, that are based on the features of the data. The decision rules are chosen in a way that they can best split the data into different groups, called branches, that correspond to different classes or outcomes.

For example, consider a simple decision tree that is used to classify types of vehicles based on their characteristics, in figure 2.1. The first decision point is "Number of wheels", where a value of 4 leads the tree to predict the vehicle type is "Car". If the vehicle has 2 wheels, it will move to the next decision point, "Has motor", where a value of "No" leads the model to predict the vehicle type is "Bicycle". If the vehicle has a motor, the tree will predict the vehicle type is "Bike".

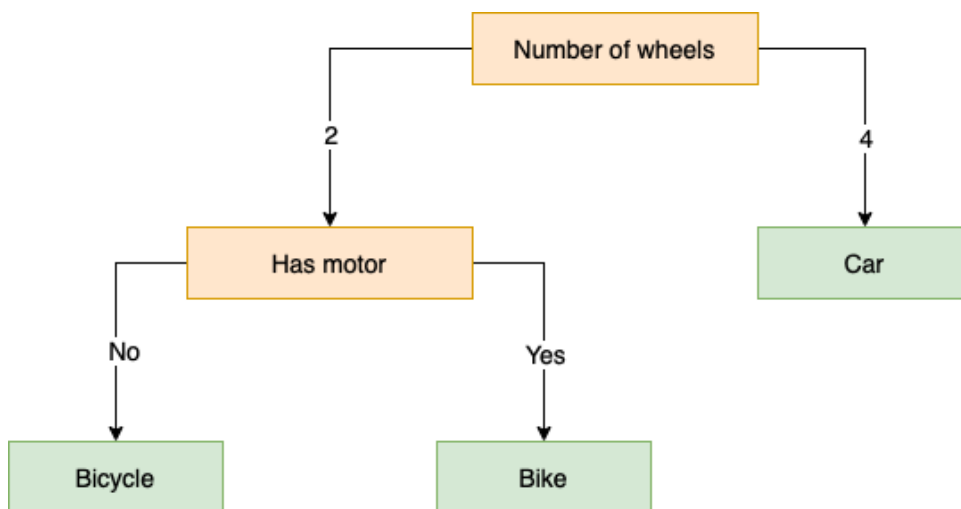


Figure 2.1: Decision tree example for vehicle types

Decision trees are simple to understand and interpret, even for non-experts, and they can handle both categorical and numerical data. They are also relatively easy to implement and they can handle missing data. However, decision trees can be very sensitive to small changes in the data and they can easily overfit to the training data, especially when they are deep and complex.

2.2.2 Random Forest

Random Forest is an ensemble machine learning algorithm that is used for classification and regression tasks. It is a type of decision tree algorithm that is composed of multiple decision trees, seen in the previous section, 2.2.1.

The basic idea behind Random Forest is to combine the predictions of multiple decision trees to improve the overall performance and reduce overfitting. The decision trees are constructed using a process called bootstrap aggregating, or bagging, where the algorithm randomly selects a subset of the data to train each tree. This helps to reduce the variance of the model, as each tree is trained on a different subset of the data and is less likely to be overfitted to the training data.

Once all of the decision trees have been trained, the Random Forest algorithm makes a final prediction by combining the predictions of all the trees. For a classification problem, this is done by taking a majority vote among the predictions of all the trees, and for a regression problem, this is done by taking the average of the predictions.

Random Forest has several important advantages over other machine learning algorithms. It is relatively simple to implement and interpret, it can handle both categorical and numerical data, and it is not sensitive to the scale of the data. It also has the ability to handle missing data, and it can be used for feature selection.

However, Random Forest can be computationally expensive, especially when working with large datasets and when many trees are used. Additionally, Random Forest can be less effective when the number of features is much larger than the number of observations.

2.2.3 Nearest Neighbor Algorithms

Nearest neighbor algorithms are a class of supervised machine learning algorithms that function based on the premise of instance-based learning, further explained in section 2.5. Unlike other types of algorithms, where a model is trained and used to make predictions, nearest neighbor algorithms instead rely on memorizing the training dataset. When it comes to making a prediction for a new instance, these algorithms look at the instances from the training dataset that are nearest to the new instance and use them to predict the label of the new instance.

Nearest neighbor algorithms can be particularly effective in scenarios where the quantity of available data is limited. Instead of requiring a vast amount of data to train a comprehensive model, these algorithms rely on local approximations using the nearest instances in the dataset, thus allowing for meaningful predictions even with smaller datasets.

These algorithms can be used for both classification and regression tasks, and despite their simplicity, they have been proven to perform well on a variety of tasks and has been successfully applied in a wide range of fields, including computer vision, speech recognition, and, more useful for this work, recommendation systems.

A famous and widely used variation of nearest neighbor algorithms is the k-Nearest Neighbors (k-NN), which we review in the following section (2.2.3.1). After that, we explore the existing distance metrics that can be used for similarity measuring in nearest neighbor algorithms (2.2.3.2).

2.2.3.1 k-Nearest Neighbors (k-NN)

The k-nearest neighbors (k-NN) algorithm is a specific implementation of the nearest neighbor algorithm, where k represents the number of nearest neighbors used for making the prediction. The

value of 'k' is a hyperparameter that can be tuned to achieve the best performance for a particular dataset.

The steps involved in the k-nearest neighbors (k-NN) algorithm are as follows [33]:

1. Store the training set
2. Calculate distance: The next step is to calculate the distance between the input sample and all the samples in the training set. This can be done using various distance measures, such as Euclidean distance, Manhattan distance, or Cosine similarity, depending on the type of data. For example, the Euclidean Distance is defined as follows:

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.1)$$

3. Find the k-nearest neighbors: After calculating the distances, the k-NN algorithm then selects the samples in the training set that are closest to the input sample.
4. Make a prediction: The prediction is made based on the class labels or target values of the nearest neighbors. Assign the class containing the maximum number of nearest neighbors.

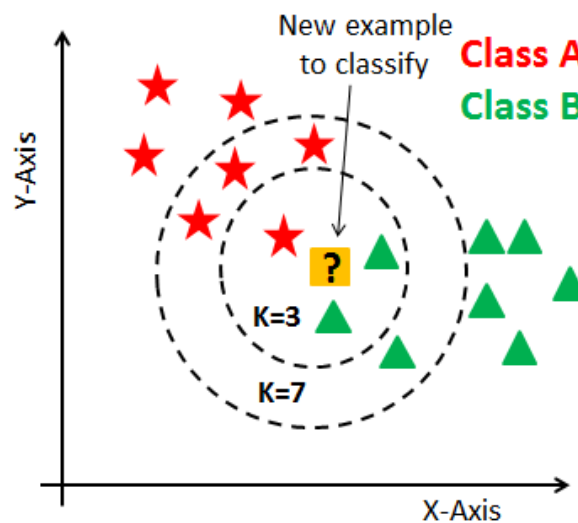


Figure 2.2: Simple KNN representation - Source [13]

In figure 2.2, we can observe the importance of the 'k' value. If we used $k = 3$, the new example would be assigned to class B, and if we instead chose $k = 7$, it would be class A.

2.2.3.2 Distance Metrics and Similarity Measures

Similarity measures and distance metrics are crucial for the operation of nearest neighbor algorithms, as they quantify the closeness between data points. Three important metrics that are used

in a variety of fields, including multi-attribute decision making, are the Gower distance, Euclidean distance, and Jaccard Dissimilarity distance.

- **Gower distance:** The Gower distance is a measure of dissimilarity between two data points. It is made so that it can handle both numerical and categorical variables. Gower calculates the distances between records based on the sum of the weighted absolute differences between the values in the columns. The result of this calculation ranges from 0 to 1, meaning identical and completely dissimilar, respectively.
- **Euclidean Distance:** The Euclidean distance, originating from the Pythagorean theorem, is a commonly used metric for numerical data. It computes the straight-line distance between two points in an n-dimensional space. This method is suitable for continuous numerical data and is sensitive to changes in the magnitudes or scales of the variables. It is mathematically defined as:

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.2)$$

Where n is the number of dimensions (attributes or features), and x_i and y_i are the values of objects ([30]).

- **Jaccard Dissimilarity distance:** The Jaccard coefficient is a method used for comparing the similarity and diversity of sample sets. It measures the similarity between the sets and is defined by the size of the intersection divided by the size of the union of the sample sets. The Jaccard Dissimilarity is one minus the Jaccard coefficient. It is defined as:

$$1 - \frac{|A \cap B|}{|A \cup B|} \quad (2.3)$$

Where $|A \cap B|$ represents the intersection of cluster or sets A and B, the elements common to both, and $|A \cup B|$ represents the union of sets A and B, all elements that are either in A, B or in both. The Jaccard Dissimilarity is a measure of how dissimilar two sets are. Thus a value of 0 indicates that they are identical, and 1 indicates they have nothing in common.

2.2.4 Performance Evaluation

Evaluation metrics play a crucial role in assessing the performance of a machine learning model after it has been trained. These metrics provide a quantitative measure of the model's ability to accurately predict outcomes based on the training data. By using evaluation metrics, data scientists and machine learning engineers can identify areas for improvement, tune the model's parameters, and optimize its performance. There are various evaluation metrics available, each designed to measure specific aspects of a model's performance.

In this section, we will explore some of the most common evaluation metrics used in machine learning, especially for supervised learning techniques. We will also discuss the advantages and

limitations of each metric, as well as when and how they should be used to assess the performance of a machine learning model.

The simplest evaluation metric is accuracy, which is calculated by dividing the number of correct predictions by the total number of predictions made by the model [25].

$$Accuracy = \frac{\text{Number of correct results}}{\text{Total number of predictions}} \quad (2.4)$$

Classification accuracy works well if there is an equal number of samples belonging to each class. For instance, imagine that the training set consists of 98% samples of class A and 2% of class B, the model can achieve 98% training accuracy by merely classifying every single sample as class A [25]. If for this particular model, the number of samples between classes was equal, as it should be, the accuracy of the model would drop significantly.

For binary classification problems, confusion matrixes give us a matrix as output and describe the complete performance of the model. Assume we have samples belonging to two classes: "YES" or "NO" and that we have a model that predicts a class for a given input. We have the corresponding confusion matrix in table 2.1.

Table 2.1: Confusion Matrix Example

		Actual class	
		YES	NO
Predicted class	YES	True Positive	False Positive
	NO	False Negative	True Negative

We find four different terms in the matrix:

- **True Positive (TP)** - The situation where we predict YES and the actual output is also YES.
- **True Negative (TN)** - The situation where we predict NO and the actual output is also NO.
- **False Positive (FP)** - The situation where we predict YES and the actual output is also NO.
- **False Negative (FN)** - The situation where we predict NO and the actual output is also YES.

The performance of a predicting model can be then given by the analysis of four common evaluation measures [14]:

- **Accuracy (AC)** - the percentage of the total number of predictions that were correct. Calculated as follows:

$$AC = \frac{TP + TN}{TP + FP + FN + TN} \quad (2.5)$$

- **Precision (P)** - the percentage of predicted positive cases that were actually positive [14]. Calculated as follows:

$$P = \frac{TP}{TP + FP} \quad (2.6)$$

- **Recall (R)** - the percentage of positive cases that were correctly predicted [14]. Also known as Sensitivity (SEN) or True-Positive Rate (TPR). Calculated as follows:

$$R = \frac{TP}{TP + FN} \quad (2.7)$$

- **Specificity (SPE)** - the percentage of negative cases that were correctly predicted [14]. Also known as True-Negative Rate (TNR). Calculated as follows:

$$SPE = \frac{TN}{TN + FP} \quad (2.8)$$

Additionally, we have two more measures similar to the True-Positive Rate and the True-Negative Rate:

- **False Positive Rate (FPR)** - the percentage of negative cases that were incorrectly predicted and considered as positives [25]. Calculated as follows:

$$FPR = \frac{FP}{TN + FP} \quad (2.9)$$

- **False Negative Rate (FNR)** - the percentage of positive cases that were incorrectly predicted and considered as negatives. Calculated as follows:

$$FNR = \frac{FN}{FN + TP} \quad (2.10)$$

Finally, particularly when dealing with imbalanced datasets, F1-score is a popular metric in the field of machine learning.

F1-score is often more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar costs. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall or the F1-score.

- **F1-score (F1)** - the harmonic mean of precision and recall. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is calculated as follows:

$$F1 = \frac{2 * (P * R)}{P + R} \quad (2.11)$$

Evaluating the performance of supervised learning models via metrics like accuracy, precision, recall, and F1-score is fundamental in fine-tuning these models. However, our exploration of machine learning doesn't stop here.

In the next section, we will explore the domain of unsupervised learning, where models must decipher patterns without labeled data and the assessment of their performance calls for a whole different set of techniques.

2.3 Unsupervised Learning Techniques

Unsupervised learning is one of the main types of machine learning methods, focusing on uncovering hidden patterns and structures from unlabeled data [26]. Kevin P. Murphy, in "Machine Learning: A Probabilistic Perspective", highlights the utility of these algorithms in analyzing data and extracting meaningful insights without requiring pre-existing labels. Unsupervised learning is often characterized as Knowledge Discovery.

One of the central techniques within unsupervised learning is clustering. This method, explored in section 2.3.1, aims to separate data into distinct groups or clusters based on their inherent characteristics and differences. Clustering has found utility in numerous applications relevant to task management and resource planning, where grouping similar tasks can lead to improved efficiency and productivity.

Another significant technique under unsupervised learning is association rule learning. This approach, detailed in section 2.3.2, is designed to extract interesting and frequent correlations, patterns, or associations among sets of items in the data. The implications of this method for task management are promising, as it could uncover meaningful associations between tasks and resources, leading to insightful strategies for resource allocation and utilization.

Each of these techniques has its performance metrics, which will be detailed later in section 2.3.3. Understanding these metrics is crucial for evaluating the effectiveness of an unsupervised learning algorithm and its suitability for specific applications, such as in the realm of task management and resource planning.

The subsequent sections will explain the specifics of these methods, exploring their strengths, limitations, and practical applications. This discussion aims to provide an understanding of the state-of-the-art in unsupervised learning, setting the stage for its exploration in the context of task management and resource planning.

2.3.1 Clustering

Clustering is a fundamental unsupervised learning technique in machine learning that aims to group similar data points together. Each group, or cluster, represents a collection of data points that share common characteristics or features. By identifying these patterns within the data, clustering can reveal valuable insights that might not be immediately evident otherwise.

Clustering algorithms function by using various metrics or distance measures, which assess the degree of similarity between data points, similar to the ones described in section 2.2.3.2. These measures form the basis of how clusters are formed and are key to the successful application of these techniques.

The process of clustering is useful in a variety of domains. In the context of this work, clustering will be instrumental in task management and resource planning by enabling the grouping of similar tasks together, fostering better understanding and efficiency in process management.

There are many existing clustering algorithms, each with its own strengths and weaknesses. In this subsection, we will describe three commonly used algorithms: K-means, K-modes, and DBSCAN.

2.3.1.1 K-means

The k-means algorithm is an unsupervised learning algorithm from the family of clustering algorithms. The goal of the k-means algorithm is to divide a set of n observations into k clusters, where each observation belongs to the cluster with the nearest mean. The k-means algorithm consists of the following steps:

1. **Initialization:** The first step is to initialize the centroids, which are the centers of the k clusters. This can be done randomly or using some other method, such as the K-Means++ algorithm.
2. **Assignment:** In this step, each observation is assigned to the nearest centroid based on the Euclidean distance between the observation and the centroids.
3. **Recalculation:** The next step is to recalculate the new centroids as the mean of all the observations in each cluster.
4. **Repeat:** The assignment and recalculation steps are repeated until the centroids no longer change or a maximum number of iterations is reached.
5. **Output:** The final output of the k-means algorithm is the cluster assignments for each observation, along with the final locations of the centroids.

For the current work, the activities would be separated into k clusters and any newly created activity would be assigned to the cluster it is closest to, so that PLAN would then be able to make suggestions with common information among the data points of that cluster.

The k-means clustering algorithm is widely recognized as a powerful and widely used tool in data mining. Despite its widespread use, the algorithm does have some limitations, particularly with regards to the random initialization of centroids, which can result in unexpected convergence [3].

2.3.1.2 K-modes

The K-modes algorithm is an extension of the K-means algorithm, designed to handle categorical data. In a dataset consisting of categorical variables, the conventional K-means algorithm is not suitable due to its reliance on numerical metrics, such as Euclidean distance. To resolve this issue, the K-modes algorithm was introduced, which modifies the computational aspect of the K-means

algorithm to handle and cluster categorical data [16]. The K-modes algorithm follows these main steps:

1. Initialization: Similar to the K-means algorithm, the first step is to initialize the modes, which are the centers of the k clusters. This can be done randomly or using other initialization methods.
2. Assignment: Each observation is assigned to the nearest mode based on the similarity measure, often, the simple matching dissimilarity measure is used, which counts mismatches in categorical attributes.
3. Recalculation: The modes of the clusters are then recalculated. The new mode is the most frequent category value of each attribute across the data points within the cluster.
4. Repeat: The assignment and recalculation steps are iterated until the modes no longer change or a maximum number of iterations is reached.
5. Output: The final output of the K-modes algorithm is the cluster assignments for each observation, along with the final locations of the modes.

The k-modes algorithm was first proposed by Zhexue Huang in his seminal paper, where he laid out the groundwork for handling categorical data through clustering [16]. Huang's work provides a detailed exploration of the algorithm and emphasizes its utility in the context of large datasets. His contributions to this area have greatly influenced our understanding and usage of the k-modes algorithm, particularly in how we perceive its strengths and limitations. This paper serves as the foundation for the steps of the k-modes algorithm detailed above and informs our discussion on the method's capabilities and potential drawbacks.

In the context of this work, the K-modes algorithm could be used to cluster activities based on their categorical attributes. Any new activity would be assigned to the cluster whose mode is most similar, providing a basis for task suggestion and planning within PLAN.

While K-modes is an effective tool for categorical data clustering, it should be noted that, similar to K-means, the initial selection of modes can significantly affect the algorithm's performance and final clustering results. It also assumes that each cluster is spherical and doesn't account well for clusters of different shapes.

2.3.1.3 DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm, widely used due to its efficacy in discovering clusters of various shapes and sizes, and its ability to identify noise [11]. The DBSCAN algorithm operates by defining a cluster as a high-density region separated by low-density regions. Unlike partitioning and hierarchical clustering methods, it does not require the user to define the number of clusters a priori. The main steps of the DBSCAN algorithm are as follows:

1. For each point in the dataset, the algorithm starts by exploring its neighborhood to find at least a minimum number of points (MinPts) within a certain radius (eps). If the condition is met, a new cluster is created, and the point becomes a core point. Otherwise, the point is labeled as noise.
2. For each new core point discovered in the previous step, the points within its eps distance are added to the cluster. If they also have at least MinPts within their eps distance, they also become core points.
3. The process continues until no new point can be added to any cluster.

DBSCAN's ability to handle noise and find clusters of different shapes and sizes can be exceptionally beneficial for the field of task planning. By identifying groups of activities that are closely related (dense clusters), while also separating out less related activities (noise), the quality of task planning can potentially be improved. Such enhanced grouping can facilitate more nuanced and personalized planning strategies, accounting for individual patterns, preferences, and habits. The robustness of DBSCAN in distinguishing noise from significant clusters also supports more reliable decision-making in planning, potentially leading to more efficient and effective task execution.

However, DBSCAN also has some limitations. The algorithm relies heavily on the eps and MinPts parameters, and choosing appropriate values can be challenging. Furthermore, DBSCAN can struggle with datasets that have varying densities.

2.3.2 Association Rules

Association rule learning is a critical component of unsupervised machine learning that focuses on discovering interesting relationships or associations among a set of items in large databases. This process can be seen as identifying rules that adequately predict the occurrence of an item based on the occurrences of other items.

The Apriori algorithm, introduced by Agrawal and Srikant [1], is among the most famous algorithms used for mining frequent itemsets for boolean association rules. It uses a "bottom-up" approach, where frequent subsets are extended one item at a time, and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

The algorithm has many benefits. First of all, it reduces the computational demand by only considering itemsets that are likely to be frequent. It also provides a level of adaptability as it allows the user to define the minimum support and confidence, which can then be tailored based on the specific needs of the dataset and problem. This adaptability means that the user can control the number and quality of the rules generated by manipulating these parameters.

In the context of planning and organizing projects, association rule learning can play a crucial role. If we can generate a set of contributions that often occur together within a project, we can suggest these associations to users when they are developing new contributions, thereby using past patterns to inform future work. This use could lead to significant efficiency gains, as it can ease

and facilitate the decision-making process by using established patterns, and therefore enhance the overall planning process.

Nevertheless, it is worth noting that the Apriori algorithm has its downfalls. It may be inefficient in scenarios where datasets have numerous frequent itemsets, long patterns, or where the minimum support threshold is set very low. However, its simplicity, clarity of rules, and ease of implementation have ensured its position as a cornerstone in association rule learning.

2.3.3 Performance Evaluation

Correctly measuring the performance of unsupervised learning algorithms, namely clustering algorithms, is critical. This is especially true as clusters are often manually and qualitatively inspected to determine whether the results are meaningful [36].

To start us off, the Silhouette Score measures how close each point in a cluster is to points in the neighboring clusters. This metric has a range of [-1, 1] and is an excellent visual tool for examining similarities within clusters and differences between clusters. It uses the compactness of individual clusters (intra cluster distance) and separation amongst clusters (inter cluster distance) to measure an overall representative score of how well our clustering algorithm has performed [18]. The Silhouette score for a given datapoint \mathbf{i} is as follows:

$$S_i = \frac{b_i - a_i}{\max(b_i, a_i)} \quad (2.12)$$

In equation 2.12, b_i is the inter cluster distance, defined as the average distance to the closest cluster of datapoint \mathbf{i} except for which it's a part of:

$$b_i = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \quad (2.13)$$

Similarly, in equation 2.12, a_i is the intra cluster distance, defined as the average distance to all other points in the cluster which \mathbf{i} belongs to [18]:

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \quad (2.14)$$

A Silhouette Plot represents the cluster label on the y-axis, while the actual Silhouette Score is on the x-axis. The size of the silhouettes is proportional to the number of samples inside that cluster. Here is an example with four clusters:

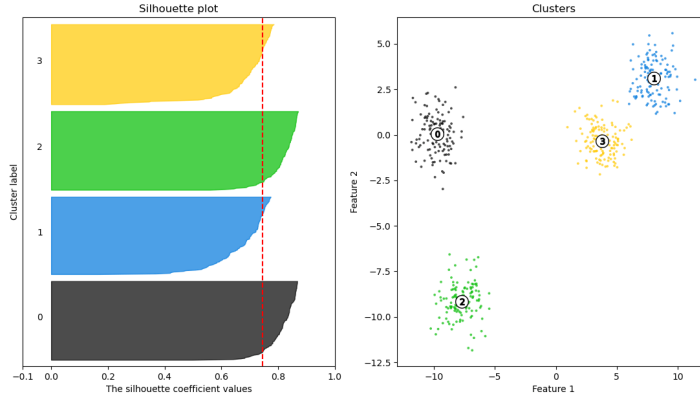


Figure 2.3: Silhouette Plot Example with 4 clusters - Source [5]

The Rand Index is another widely used metric. It calculates a similarity between two clusters by examining all sample pairs and determining the number of pairs that are assigned to the same or different clusters in both the predicted and actual clusterings [36]. The formula for the Rand Index is:

$$RI = \frac{\text{Number of Agreeing Pairs}}{\text{Number of Pairs}} \quad (2.15)$$

The drawback of the Rand Index is that it requires the availability of ground-truth cluster labels for comparison with the model's performance. For CERN's particular case, this metric is much less useful than the Silhouette Score as it is unlikely that we are able to obtain cluster labels for activities or contributions. We are dealing with a pure, unsupervised learning task.

Another measure to evaluate non-labeled clustering problems is the Calinski-Harabasz Index, also known as the Variance Ratio Criterion. It is defined as the ratio between the within-cluster dispersion ($\text{tr}(B_k)$) and the between-cluster dispersion ($\text{tr}(W_k)$). The higher the Index (S) is, the better the model's performance is.

$$S = \frac{\text{tr}(B_k)}{\text{tr}(W_k)} * \frac{n_E - k}{k - 1} \quad (2.16)$$

where we have:

$$B_k = \sum_{q=1}^k n_q (C_q - C_E)(C_q - C_E)^T \quad (2.17)$$

$$W_k = \sum_{q=1}^k \sum_{x \in C_q} (x - C_q)(x - C_q)^T \quad (2.18)$$

Building on the above-mentioned performance measures, it is also beneficial to consider the Davies-Bouldin Index. This index provides an evaluation metric for clustering algorithms by quantifying the average 'similarity' between clusters [10]. The similarity in this context is a measure

that combines the distance between clusters and the size of the clusters themselves.

The Davies-Bouldin Index (DBI) is calculated as follows:

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{i \neq j} (R_{ij}) \quad (2.19)$$

In this equation, R_{ij} is the ratio of the sum of within-cluster distances in cluster i and cluster j to the distance between the cluster centers. Essentially, a lower Davies-Bouldin Index indicates a model with better separation between the clusters.

Like the Silhouette Score, the Davies-Bouldin Index does not require the availability of ground truth labels, making it an apt measure for unsupervised learning scenarios. However, it's important to note that DBI focuses on the similarity between a cluster and its nearest neighbor, so it may not capture the overall structure as effectively if the dataset has complex multi-level clustering.

In conclusion, for non-labeled clustering problems, selecting the appropriate measure to assess model performance is dependent on the specifics of the data and the goals of the analysis. Indices such as the Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index offer viable solutions, each providing different perspectives on the quality of the clustering.

2.4 Natural Language Processing Techniques

Natural Language Processing (NLP) is an evolving field within artificial intelligence that emphasizes the interaction between computers and humans using natural language. As Chowdhary describes in his book "Fundamentals of AI", NLP is fundamentally a set of computational techniques for automatic analysis and representation of human languages [9].

Natural languages, due to their complexity and infinite set of sentences, present significant challenges. Ambiguity is one such challenge: words can have multiple meanings, and the interpretation of sentences can change based on the context [9].

The current NLP algorithms have limitations in dealing with these complexities. While they are efficient at retrieving texts, spell checking, and word-level analysis, interpreting sentences and extracting meaningful information remain challenging tasks [9]. This is largely due to the algorithms' inability to consider background information - the contextual knowledge that humans naturally bring to language comprehension.

Despite the challenges, the potential and relevance of NLP in today's digital age is immense. It is becoming increasingly essential in fields such as online information retrieval, text summarization, question-answering systems, and improving human-machine communication [9].

In the following subsections, we will describe specific NLP techniques, such as text representation and feature extraction, and text similarity measures.

2.4.1 Text Representation and Feature Extraction

To process and understand natural language, we need to convert it into a format that is understandable by a machine. This process involves the extraction of relevant features from the text,

which can be used in various NLP tasks such as sentiment analysis, document clustering, or topic modeling. Two popular methods of text representation and feature extraction are Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings.

TF-IDF is a statistical measure used to evaluate the importance of a word in a document, part of a corpus. This technique transforms text into a meaningful representation of numbers which is used to analyze the document. It consists of two parts: Term Frequency (TF) and Inverse Document Frequency (IDF).

TF refers to the frequency of a word in a particular document. IDF, on the other hand, diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. Thus, TF-IDF tends to filter out common terms such as 'is', 'of', and 'that', in favor of terms that are more useful for distinguishing between documents [24].

Word embeddings are the modern way of representing words as vectors. The goal of word embeddings is to redefine the high-dimensional word features into low-dimensional feature vectors by preserving the contextual similarity in the corpus. They are capable of capturing the context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

2.4.2 Text Similarity Measures

A fundamental task in many NLP applications, including information retrieval, text mining, and machine translation, is the measurement of similarity between texts [12]. Cosine similarity is a traditional similarity measurement metric that is often combined with TF-IDF [23].

Cosine similarity is a metric that calculates the cosine of the angle between two non-zero vectors. In NLP, each vector usually represents a text, where each dimension corresponds to a distinct word (or sometimes n-gram or concept), and the value in each dimension corresponds to the weight of that word (often a TF-IDF weight or a count of occurrences) in the text.

Cosine similarity depends on the angle between the vectors, rather than their magnitudes. Therefore, even if two documents have vastly different lengths, if they express the same topics (i.e., they use the same words with similar proportions), their cosine similarity will be high.

Cosine similarity is calculated using the following formula [19]:

$$\cos(A, B) = \frac{A \cdot B}{\|A\| \|B\|} \quad (2.20)$$

Overall, cosine similarity is a very valuable tool in NLP, useful in a wide variety of tasks and scenarios, including document classification, clustering, information retrieval, and recommendation systems. It's also worth noting that cosine similarity is not the only text similarity measure. Others include the Jaccard index (2.2.3.2), Euclidean distance (2.2.3.2), and Manhattan distance, each of which may be more appropriate in certain situations. However, cosine similarity is arguably the most common and versatile one, and therefore the focus of this section.

2.5 Dealing with Low-data Situations

Machine learning models typically need large amounts of data to learn patterns and make accurate predictions or suggestions. However, in many real-world cases, the available data can be limited. This presents significant challenges in training robust and reliable models. Several strategies have been developed to deal with low-data situations, a few of which are discussed in this section.

Data augmentation is a widely employed strategy in many machine learning applications where data is sparse. It involves creating new synthetic training samples by applying transformations that preserve the label information [31]. Despite its popularity, this approach may not always be suitable, particularly in cases where synthetic data can't be used.

Real-time similarity computation is another method that can be beneficial in low-data situations. By calculating similarity measures like cosine similarity (section 2.4.2) or Gower distance (section 2.2.3.2) in real-time, new data points can be compared with existing ones to generate predictions. This approach sidesteps the need for a traditional training phase, making it suitable for scenarios where historical data is limited.

In the context of sparse data, instance-based learning (IBL) becomes valuable. This learning paradigm, often associated with methods such as k-nearest neighbors (section 2.2.3.1), relies on specific instances or data points rather than a general model learned from the data. This approach is particularly useful when dealing with limited data, as it also doesn't require a separate training phase. However, IBL is often computationally expensive and sensitive to the choice of the algorithm's similarity function [2].

Additionally, feature extraction techniques can help uncover useful information from a limited amount of data. By transforming the input data into a set of new features that capture the essential characteristics of the data, these techniques can enhance the performance of machine learning models.

Lastly, transfer learning is an approach where pre-trained models on large datasets are adapted for a different but related task [31]. By leveraging the learned representations from a related task, transfer learning can mitigate the effects of limited data availability, offering a promising approach to dealing with low-data situations.

In summary, while limited data presents significant challenges in machine learning, there are many different strategies to navigate such situations. Understanding these techniques and their applicability to specific tasks is vital for effective machine learning application in real-world, low-data situations.

2.6 Content-Based Recommendation Systems

Recommendation systems are systems that are designed to recommend things to the user based on many different factors. A recommendation system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user's preference and interest. It finds out the match between user and

item and imputes the similarities between users and items for recommendation. Both the users and the services provided have benefited from these kinds of systems. The quality and decision-making process has also improved through these kinds of systems.

Content-based recommendation systems, as detailed in Michael Pazzani's chapter in "The Adaptive Web," focus on the recommendation of items to a user by leveraging a description of the item and a profile of the user's interests [27]. These systems have wide applicability, extending to various domains such as recommending web pages, news articles, restaurants, television programs, and items for sale. Fundamentally, content-based recommendation systems incorporate a method for item description, a mechanism for creating a user profile, and a means of comparing these items to the user profile to determine recommendations.

In web applications, which is a prevalent setting for modern recommendation systems, the user interacts with a summarized list of items. The user can then select among these items for more detailed information or further interaction. As the total number of items often exceeds what can be comfortably displayed on a webpage, it becomes essential to carefully select a subset of items or determine a specific order of display for the user. Content-based recommendation systems tackle this issue by analyzing item descriptions to identify those that align closely with the user's interests [27].

However, as per Pazzani's work, it's worth noting that content-based recommendation systems are not without limitations [27]. For example, the performance of such systems might suffer when the content lacks sufficient information to discriminate between items the user likes and dislikes.

In summary, content-based recommendation systems provide a robust and flexible method of personalizing recommendations to individual users based on their preferences and interests. This ability to tailor suggestions becomes particularly crucial in environments like CERN's task planning platform, where the goal is to aid users by recommending relevant tasks based on the item to be suggested, the activity and its information. However, it's essential to keep in mind the limitations of such systems, such as their dependence on the content's sufficiency to discriminate user preferences effectively. In the next section, we will dive into more detail on existing task management and resource planning platforms.

2.7 Task Management and Resource Planning

Task management is the process of defining, planning, tracking, and organizing tasks and activities that need to be completed as part of a project or work. It involves identifying the tasks to be done, setting priorities, allocating resources, and tracking progress to ensure that tasks are completed on time and within budget. Task management helps improve work processes' efficiency and effectiveness and provides a centralized platform for teams to collaborate and communicate.

Resource planning is the process of identifying, organizing, and allocating the resources (e.g., people, equipment, materials, and budget) needed to complete a project or work. It involves determining the availability of resources, ensuring they are allocated to the right tasks, and tracking their utilization over time. Resource planning helps organizations optimize their resources, reduce

waste, and ensure that projects are completed on time and within budget. By aligning resources with project requirements, organizations can ensure they have the right resources at the right time to achieve project goals.

Task management and resource planning are critical components of project management that have evolved over time to meet the changing needs of organizations and teams. The history of task management and resource planning can be traced back to manual methods such as to-do lists and calendars [20]. Over the years, technology has played an increasingly important role in facilitating these processes, leading to the development of sophisticated project management software.

In recent years, technological advances have led to the development of cloud-based project management software that offers increased flexibility and scalability. The increasing availability of these tools has made it easier for organizations and teams of all sizes to implement effective task management and resource planning processes.

There are several existing task management and resource planning platforms available in the market today, including Asana [4], Trello [34], Redmine [29], Jira [17], and MS Project [28], among others. These platforms vary in terms of their scope, complexity, and target audience, and can range from simple to-do list apps to comprehensive project management software.

In Cheema's "The choice of project management software by project managers" [8], the authors propose a study that aims to determine which project management software is the most used by project managers and which is the most appropriate for a company by comparing the five platforms mentioned above. In a sample of 101 qualified project managers, the most used software platforms were MS Project and Jira. Then, after setting seven different project management criteria and their importance: Tasks, Collaboration, Project Essentials, Portfolio, Management of resources, Platform, and Procurement, project managers were inquired about each platform regarding the criteria. In figure 2.4, we can see an aggravated matrix that considers the weights of each criterion and the survey results. The results inferred that JIRA is the best project management software currently available to manage IT projects.

	C1	C2	C3	C4	C5	C6	C7	SUM
	Task	Colab.	Projects	Portfolio	Resource Management	Platform	Procurement	
MS Project	-0.017	-0.032	-0.021	0.026	0.032	0	0.152	0.138
JIRA	0.095	0.011	0.023	0.035	0.033	0	0.166	0.365
Trello	-0.032	0.013	-0.016	0.031	0.016	0	0.140	0.152
Asana	-0.032	0.008	0.003	-0.041	-0.115	0	0.230	-0.408
Redmine	-0.011	-0.000	0.011	-0.051	0.033	0	0.228	-0.247
IDEAL	0.095	0.013	0.023	0.035	0.033	0	0.166	0.367
Anti-Ideal	-0.032	-0.032	-0.021	-0.051	-0.115	0	-0.230	0.040

Figure 2.4: Aggravated Matrix - Source [8]

Overall, the main benefits the existing platforms bring to the process of task management and resource planning within a company or an organization are the following:

- Centralized platform for managing tasks and resources - Task management and resource planning platforms provide a centralized platform for organizing and tracking tasks and resources, improving collaboration and communication among team members.
- Improved efficiency and productivity - By automating repetitive and time-consuming tasks, these platforms can help organizations to improve their efficiency and productivity.
- Increased visibility and control - Task management and resource planning platforms offer increased visibility into project progress and resource utilization, allowing organizations to make informed decisions and take corrective action when necessary.
- Enhanced resource utilization - By optimizing the allocation and utilization of resources, these platforms can help organizations to reduce waste and improve project planning and execution.

However, especially for large-scale organizations, they can have their drawbacks:

- Initial setup and customization - Implementing a task management and resource planning platform can be time-consuming and require significant effort to customize it to meet the specific needs of an organization.
- User adoption - Adopting a new task management and resource planning platform requires buy-in from team members and can be challenging, mainly if they are resistant to change or prefer to use manual methods.
- Integration with other tools and systems - Task management and resource planning platforms may need to be integrated with other tools and systems used by the organization, adding to the complexity of the implementation. This is particularly true for a complex organization with many technologies and systems like CERN.
- Cost - Task management and resource planning platforms can be expensive, particularly for comprehensive and feature-rich solutions, making it important for organizations to carefully evaluate their needs and budget before choosing a platform.

2.8 Machine Learning in Task Management and Resource Planning

The integration of machine learning into task management and resource planning has recently gained significant attention in the field of project management. With the growing amount of data generated by organizations and the increasing demand for efficient and effective project management, machine learning has the potential to revolutionize the way tasks and resources are managed. Machine learning algorithms can analyze large amounts of data, identify patterns, and make predictions about future behavior, which can inform decision-making and improve the overall efficiency and effectiveness of task management and resource planning.

Recent advancements in machine learning have enabled the development of sophisticated task management and resource planning platforms that use predictive analytics, automated resource allocation, and dynamic project scheduling to optimize project performance. The use of machine learning in task management and resource planning can help organizations improve project planning, reduce manual effort, and increase overall project success.

However, while the potential benefits of machine learning in task management and resource planning are significant, there are also challenges associated with the integration of these technologies. These challenges include the need for substantial investment in data collection and preparation, the potential for biased results, and the need for thorough testing and validation of machine learning models.

Ana Sá's report [32] focuses on the application of different machine learning approaches in recommending individuals for a specific task. The study utilizes two popular machine learning algorithms, Support Vector Machine (SVM) and Naive Bayes classifiers, to analyze the data and make recommendations. In each experiment, the classifier calculates the probability of a person performing the task based on various factors. The report aims to explore the effectiveness of these algorithms in the task recommendation process and provide insights into the strengths and limitations of each approach.

In [15], the focus is once again on the comparison of two machine learning models, a Neural Network model, and a Support Vector Machine model, in evaluating the risk level of software projects. The goal of the study was to determine which approach was more accurate in predicting the outcome of the project, which was defined as "successful", "failed", or "challenged". The models were developed using a set of relevant data and tested using a comprehensive evaluation process. The initial results showed that the SVM model had a higher accuracy rate of 80% compared to the NN method, which had an accuracy rate of 70%. The difference in accuracy was attributed to the NN's tendency to find the local optima, as reported in reference [19]. However, after optimizing the NN method using a Genetic Algorithm (GA), the NN-GA method showed an improvement in accuracy, surpassing the SVM model with a rate of 85% compared to 80%. The optimization of the NN method with GA reduced the search for a local optima, resulting in the improvement in accuracy.

The use of machine learning on task management and resource planning platforms can bring several benefits to organizations. It can be used for predictive task management, automated task assignment, predictive resource management, personalization, and collaboration. However, it is worth noting that the use of machine learning on said platforms is still in the early stages of development.

2.9 Summary and Identified Gaps

The present work explores the area of machine learning (ML), with a particular focus on its application in task management and resource planning. It explores different supervised learning techniques for data classification, unsupervised learning for detecting groups, Natural Language

Processing for text comprehension, and strategies for handling low-data situations. The function of ML in recommendation systems, specifically concerning task recommendations, is also thoroughly investigated.

The study critically assesses the state of current strategies in task management and resource planning, their limitations, and the existing applications of ML in these domains. This review offers a broader context for the developed algorithm, an integral part of this research.

Notably, this research uncovers gaps in the current field. ML techniques, while possessing significant potential in managing and predicting tasks, are often complicated in low-data situations. Although strategies like data augmentation, transfer learning, and pre-trained models offer some relief, there is a marked need for additional exploration of methods to augment the efficiency of ML under data-scarce conditions, not only at CERN but in the field of study.

Furthermore, the integration of ML in task management and resource planning presents considerable challenges. These range from the requirement for substantial data collection and preparation, the potential for biased outcomes, to the need for testing and validation of models.

While the current literature has dedicated significant effort to comparing different ML algorithms such as SVM, Naive Bayes, and NN in task recommendation or risk evaluation, there is always room for further research in different contexts. The goal is to consistently achieve higher accuracy rates, potentially through advanced optimization methods, despite some algorithms already demonstrating superior performance after optimization.

Lastly, it is crucial to note that the application of ML in task management and resource planning platforms remains in an early stage of development. This provides a wide berth for innovation and improvement, particularly in areas like predictive task management and automated task assignment.

In light of these identified gaps, the thesis is positioned to make a substantial contribution by advancing the integration and optimization of ML in the context of CERN, but also for task management in general.

Chapter 3

Problem Identification

In the current chapter, we introduce the problem that will be addressed in this thesis. We start off by providing a more in-depth overview of the PLAN platform, in section 3.1, starting off with a description of the platform and of its purpose.

We then describe the key components and workflow of the PLAN platform, highlighting its importance in facilitating task management and resource allocation at CERN. Understanding the functionalities and architecture of PLAN is crucial for developing an effective suggestion system that can seamlessly integrate with the existing platform and improve the overall efficiency of the activity planning process.

Then, we discuss the two use cases that the suggestion system aims to address, in section 3.2. The first use case focuses on generating recommendations for activities. The second use case revolves around suggesting contributions to activities, both automating and speeding up the process of activity creation in PLAN.

We will finish the chapter with a brief summary, taking us into the development of a solution for the problem at hand (section 3.3).

3.1 Overview of PLAN Platform

This section provides an overview of the PLAN platform, a comprehensive task-planning tool designed to facilitate the organization and management of activities at CERN, particularly during Long Shutdowns (LS) of the Large Hadron Collider (LHC) [7]. We begin with a brief description and purpose of the platform in section 3.1.1, followed by an explanation of its key components and workflow (section 3.1.2). Finally, we describe the model and the architecture of PLAN, shedding light on its underlying structure and organization, in section 3.1.3.

Understanding the PLAN platform's functionalities and architecture is essential for developing a robust and effective suggestion system that can seamlessly integrate with the existing platform, ultimately improving the overall efficiency of the activity planning process at CERN.

3.1.1 Brief Description and Purpose

As mentioned previously, PLAN is a comprehensive task-planning tool designed to facilitate the organization and management of activities at CERN, particularly during Long Shutdowns of the Large Hadron Collider. The LHC is a powerful particle accelerator that collides particles at nearly the speed of light, enabling scientists to study the fundamental constituents of matter and the forces that govern their interactions [7].

Long Shutdowns are scheduled periods of maintenance and upgrade work for the LHC and its related experiments, typically lasting around two years. These shutdowns are crucial for ensuring the efficient operation and continuous improvement of the LHC.

The primary goal of PLAN is to define the necessary work and prioritize potential options based on available resources and the importance of each activity during an LS.

As a platform, PLAN stores all activities for a specific period, facilitating the approval process and offering a coherent approach to task management. This unified platform allows decision-makers and support groups to obtain a clear overview of various requests and their potential impacts. By providing a clear and organized view of tasks, PLAN contributes to more efficient resource allocation and improved project management.

In the context of CERN, efficient task planning is of extreme importance due to the complex nature of the organization and the multitude of activities that need to be carried out during an LS. PLAN plays a crucial role in making sure that all tasks are effectively coordinated and resources are optimally utilized.

3.1.2 Key Components and Workflow

The PLAN platform is designed to handle the complex task of managing and scheduling activities within CERN, particularly those related to the LHC. The platform consists of several key components and follows a specific workflow that ensures a smooth and efficient activity planning process. Here, we briefly discuss the main components and the workflow of the PLAN platform.

The activity process in PLAN can be classified into several stages, as detailed below:

- **Created:** A group created an activity, added scheduling information, and requested contributions from other groups.
- **Prioritization:** The group submitted the activity for prioritization.
- **Resource Allocation:** The Plan coordinator then prioritizes and schedules all activities, taking into account their impact on the performance of the facilities, the resources needed from other groups, and the scheduling constraints.
- **Under Approval:** The Plan coordinator reviews the activity and checks the validity of contributions before submitting it to the approval process.
- **Approved:** The activity has been approved by the Plan coordinator and is ready to be executed.

- **Postponed:** The activity has been postponed due to unforeseen circumstances or changes in priorities
- **Approved If:** The activity is conditionally approved, pending the fulfillment of certain criteria or completion of other activities.
- **Partially Finished:** Some parts of the activity have been completed, while others are still in progress.
- **Canceled:** The activity has been canceled due to changes in priorities, lack of resources, or other reasons.
- **Finished:** The activity has been successfully completed, and all its objectives have been met.

A better understanding of the workflow and what makes activities change stage can be had with figure 3.1.

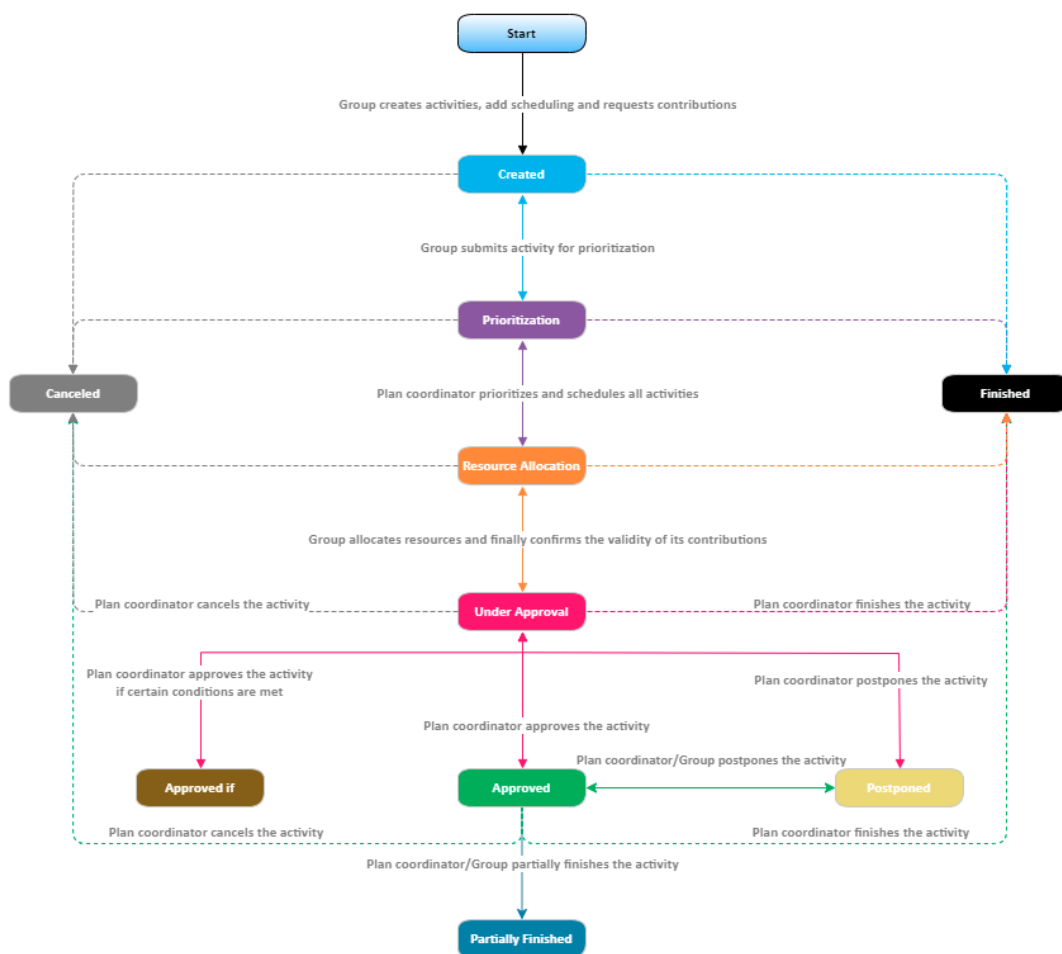


Figure 3.1: PLAN activity process diagram

It is worth noting that this diagram is used by CERN’s personnel to summarize the flow of activities in PLAN. As such, some transitions between stages, represented as arrows in Figure 3.1, are omitted not to overcomplicate the diagram.

The PLAN platform only includes activities that can have an impact on the performance of the facilities, need support from other groups in terms of resources (such as studies, designs, manufacture, tests, and infrastructure modification), or need to be scheduled by facility coordinators.

The platform’s user interface is designed to facilitate efficient communication and collaboration between different groups, providing a clear and accessible way to create, prioritize, and allocate resources for activities.

In order to provide a better understanding of the PLAN platform’s interface, a screenshot of its home page is presented in Figure 3.2. This figure showcases the layout and main components of the platform, allowing users to navigate through the various stages and features of the activity planning process.

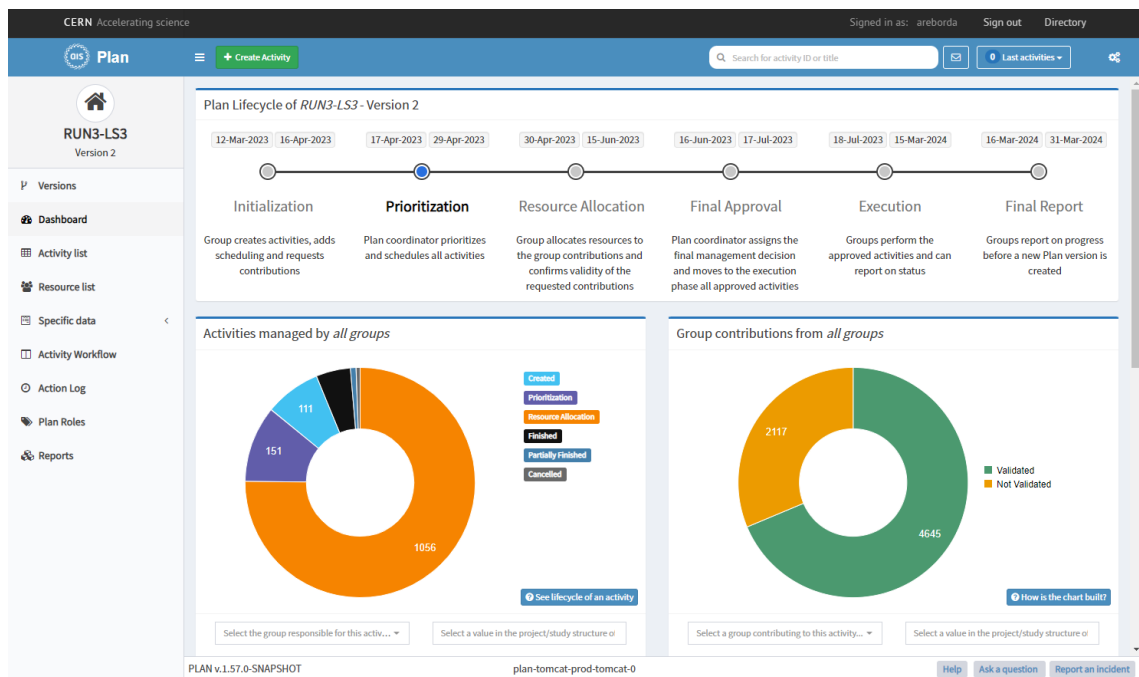


Figure 3.2: PLAN Home Page

The RUN3-LS3 Version 2 codification in Figure 3.2 indicates that we are working on the second version of PLAN regarding the third long shutdown of the four-year period that is the third run of the LHC.

3.1.3 Model and Architecture

In this section, we discuss the model and architecture of the PLAN platform, providing an overview of how it organizes and manages activities and contributions across different Long Shutdowns and

groups within CERN. The PLAN platform is a Java-based application that ensures a high level of performance and maintainability.

It uses a hierarchical model to organize and manage its activities and contributions. The top level consists of the PLAN Web tool, which encompasses the entire platform. Within the PLAN Web tool, there are separate instances for each Long Shutdown (e.g., PLAN LS1, LS2, etc.). Each Long Shutdown instance contains activities, which in turn include contributions. Figure 3.3 illustrates the hierarchical model of the PLAN platform.

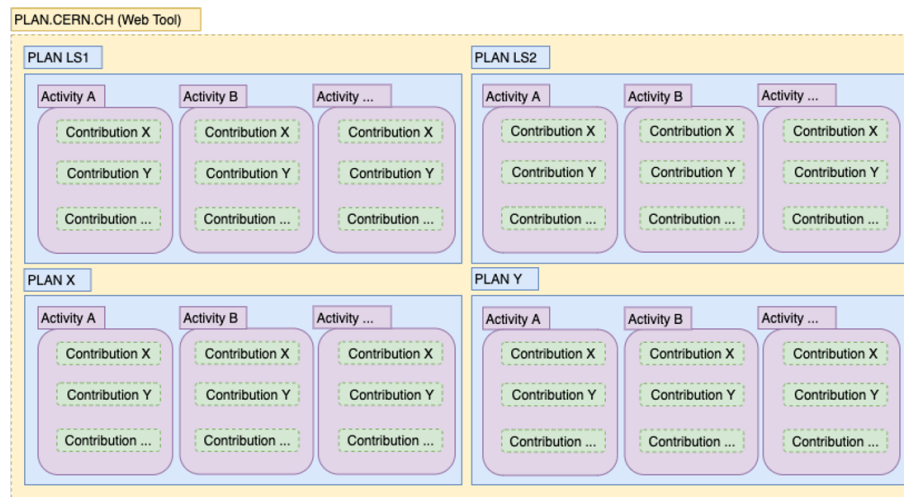


Figure 3.3: PLAN Model Diagram

The architecture of the PLAN platform is designed to ensure the efficient handling of data and improve the overall performance and maintainability of the platform. It consists of two major blocks: the write model (DDD) and the read model (CRUD).

DDD stands for Domain-Driven Design. This is a software development methodology that focuses on the core domain and domain logic of the system being built. Here, PLAN is using it since write operations are typically more complex and involve business rules and logic. CRUD, stands for Create, Read, Update, and Delete. In PLAN's context, the term CRUD is somewhat misleading as it normally also involves writing operations, but here is used to refer to a simpler part of the system focused on reading and displaying data.

The write model is responsible for handling the creation and modification of activities and contributions, while the read model deals with the retrieval and display of information.

These two blocks are connected through a publish/subscribe mechanism, which allows for a clear separation between writing and reading operations, further enhancing the platform's efficiency and scalability. Figure 3.4 presents the architectural diagram of the PLAN platform.

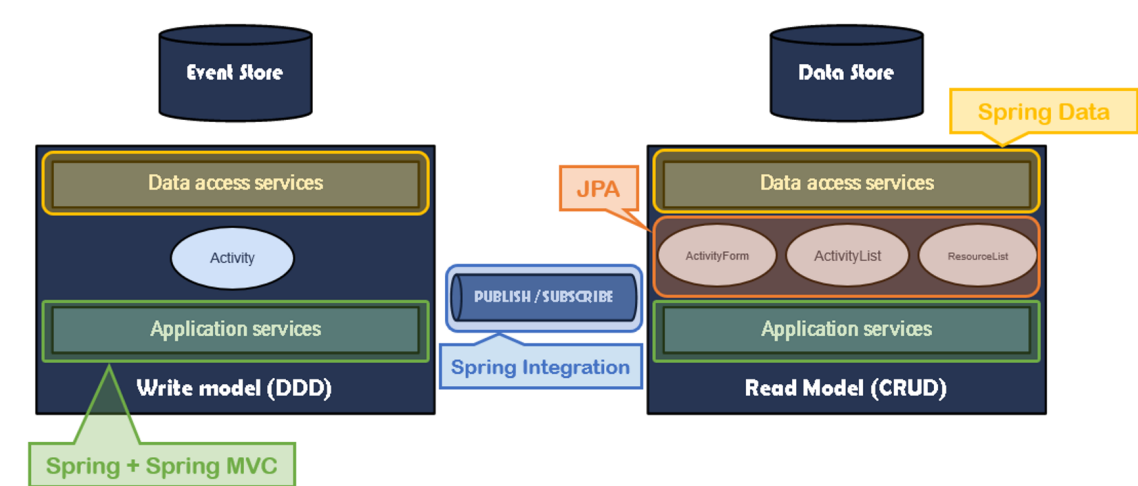


Figure 3.4: PLAN Architecture Diagram

3.2 Use Cases Understanding

In this section, we will discuss the two main features or uses cases that the tailor-made suggestion system aims to address, as identified by CERN. Both these use cases are designed to enhance the overall efficiency and effectiveness of the activity planning process within the PLAN platform by providing intelligent suggestions tailored to the specific requirements and constraints of the tasks at hand.

While both use cases address the need to make accurate and interesting suggestions to PLAN users in real-time, they differ in the focus of their suggestions. The first use case revolves around generating recommendations for activities, while the second one pertains to the contributions made towards those activities.

By addressing these distinct aspects of the planning process, the suggestion system aims to create a more streamlined and user-centric experience that ultimately enhances productivity and resource allocation during Long Shutdowns at CERN.

3.2.1 Activity Suggestion

When a user is logged into PLAN and wishes to create a new activity to be done by different CERN sections and groups, he is presented with the interface shown in Figure 3.5.

The screenshot shows a web-based form titled "Create Activity". The form is organized into several sections, each with a label and a corresponding input field. The sections are: "Title*" (text input), "Type" (dropdown menu), "Responsible group*" (dropdown menu), "Responsible*" (dropdown menu), "Project/Study*" (text input with a blue information icon), "Facilities" (text input), "Location" (text input with a green location pin icon), "Goal" (text input), and "Impact if not done / postponed" (text input). A blue "Create" button is located at the bottom right of the form. The form is set against a light blue background with a dark blue header bar.

Figure 3.5: Activity Creation Main Form

Fields like the title, location, goal, and impact if the activity is postponed can be manually described by the user as free text. The remaining values, such as type and responsible group, are organized into selectable categories.

Although only the information marked with a red asterisk is considered mandatory for creating an activity, all the information associated with an activity has to be manually entered by the user, either during activity creation (Figure 3.5) or later during editing.

The primary challenge of this use case is to simplify the activity creation process by leveraging historical data from previous PLAN activities. By analyzing and comparing this data, the suggestion system aims to identify similarities or even exact matches between the new activity and existing ones. If such similarities are detected, the system can then provide suggestions for filling out the activity information, reducing manual data entry and improving overall efficiency.

To achieve this, the suggestion system must be capable of accurately identifying relevant patterns and trends within the historical data. This requires the development of a robust and intelligent algorithm that can not only identify similar activities but also generate meaningful suggestions based on the available information. Additionally, the system should be designed to seamlessly integrate with the existing PLAN platform, ensuring a smooth and intuitive user experience.

In summary, this use case focuses on the development of a suggestion system that can facilitate and automate the process of entering activity information, ultimately improving the efficiency of the activity planning process at CERN.

3.2.2 Contribution Suggestion

Once an activity has been created in PLAN, additional information can be added, such as schedule, allocated resources, comments, and documents. The focus of the second use case is on the contributions that CERN teams make to a newly created activity, which can be entered in the resources tab of the activity. The user is presented with the interface shown in Figure 3.6.

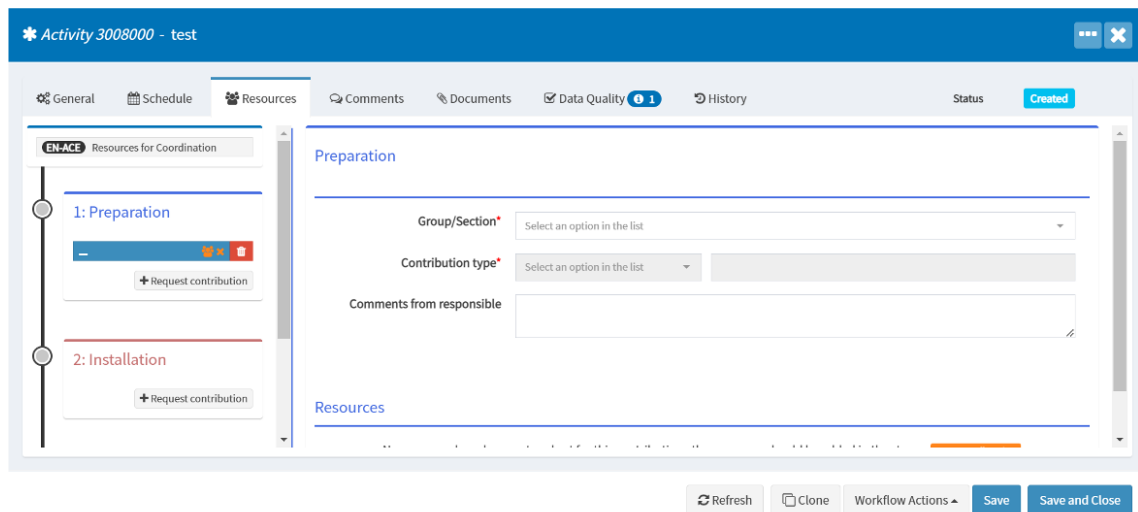


Figure 3.6: Contribution Creation Tab

As shown on the left side of Figure 3.6, contributions to an activity are assigned to one of three different activity phases: Preparation, Installation, and Commissioning. Contributions are mainly identified by their group or section (i.e., the CERN team that will make the contribution) and the contribution type.

The primary challenge of this use case is to facilitate the process of requesting contributions from other CERN teams by providing users with a list of relevant contributions that may be needed for the successful completion of the activity.

To achieve this, the suggestion system must also be capable of accurately identifying relevant patterns within the historical contribution data and generating meaningful suggestions that align with the specific requirements and information of the activity. This requires the development of a robust and intelligent algorithm that can analyze and compare past contributions to generate useful suggestions.

The potential benefits of this use case include not only automating and speeding up the process of requesting contributions but also helping users better understand what contributions may be needed for the successful completion of an activity. By providing users with tailored suggestions, the system can contribute to more efficient and effective resource allocation and overall project management at CERN.

As for the first use case, this use case focuses on the development of a system that can facilitate and automate the process of requesting contributions, also leading to an improvement in the efficiency of the activity planning process at CERN.

3.3 Summary

In the current chapter, the focus was on understanding the problem at hand. This goes through a deep understanding of the PLAN platform, which plays a pivotal role in managing tasks and allocating resources at CERN, especially during the Long Shutdowns of the LHC. PLAN's overall structure, workflow, and key components were elaborated upon.

The significance of PLAN lies in its ability to ensure efficient task planning and improve project management. The activities and their stages were explained, detailing the activity process in PLAN. Furthermore, the model and architecture of the PLAN platform were analyzed, indicating its Java-based application nature and its hierarchical data management structure.

To be able to come up with an optimal solution, it is essential to comprehend the two use cases that we want the suggestion system to tackle. The first use case focuses on generating recommendations for activities. The idea is to reduce manual data entry and improve efficiency by identifying similarities between the new and existing activities using historical data.

The second use case is about the suggestions for contributions to the activities. This case will automate the process of requesting contributions and help users understand the contributions needed for the successful completion of an activity, again using the historical data from previous contributions.

These two use cases aim to create a more streamlined and user-centric experience, ultimately enhancing productivity and resource allocation during an LS at CERN.

Chapter 4

Solution Development

Once the platform and the challenges related to this project have been established, the subsequent phase involves navigating through the process and methodologies employed to arrive at the ultimate solution for the identified problem.

In the current chapter, an understanding of the distinct stages within this process will be provided. The first step involves the comprehension and preparation of the data for utilization by machine learning algorithms. This will be explored in section 4.1, where the extraction of data from CERN's databases will be explained, alongside the generation of the final datasets.

Then, our focus shifts towards the development of our initial approach, describing the implementation of techniques accompanied by the motives behind their selection, an evaluation of their performance, and an analysis of the insights and conclusions derived from the outcomes. This is presented in section 4.2.

Thereafter, a transition is made towards a new and final approach to address the problem at hand. The development of this approach is described in detail, including what was done to optimize its potential and performance. Section 4.3 concludes with an evaluation of the performance of the final solution.

Finally, a summary of the chapter is provided, followed by a transition into the subsequent chapter, in section 4.4.

4.1 Data Preparation

The journey of constructing a machine learning model begins with data, which serves as the foundation for all other processes. This includes the extraction, preprocessing, and management of the data. The quality of data directly impacts the outcomes of the model and therefore requires rigorous attention and meticulous handling.

In this study, the data preparation and management process was designed to facilitate the application of machine learning techniques to the unique use cases of CERN's PLAN tool. This process, which is detailed in Sections 4.1.1, 4.1.2, and 4.1.3, was a multi-step procedure that

started with data extraction and ended in the creation of two meaningful datasets: the Activity Dataset and the Contribution Dataset.

Section 4.1.1 outlines the data extraction process, which began with establishing a connection to CERN's PLAN database and using an SQL client to extract relevant tables. This process was carefully designed to ensure a systematic and comprehensive extraction of the data, providing a strong foundation for the subsequent stages.

The next phase, detailed in Section 4.1.2, was the creation of the Activity Dataset. This dataset, intended to serve as the primary input for our machine learning algorithms, included historical data on past activities. The process involved carefully selecting relevant fields, merging and transforming original tables, and ultimately, creating a final dataset ready for analysis.

The Contribution Dataset, discussed in Section 4.1.3, was designed to cater to our second use case. This dataset focused on contributions made to past activities. A similar approach to the one used for the Activity Dataset was employed, with the necessary fields being identified, irrelevant columns being dropped, and the remaining datasets being merged to create the final Contribution Dataset.

Despite the unique challenges presented by the limited availability of data, the process was designed to maximally utilize the available data to address the study's objectives. The result was two comprehensive datasets, each specifically tailored to a distinct use case. These datasets provide the groundwork for the subsequent stages of the study, paving the way for the application of machine learning techniques to address the study's objectives.

4.1.1 Data Extraction

The initial stage in the data treatment process involved retrieving the pertinent tables from CERN's PLAN database. Upon receiving a comprehensive list of the available tables within the database from PLAN developers, a TNS (Transparent Network Substrate) connection was established to facilitate access to the database using an alias.

To visualize the tables, we employed an SQL client, specifically Oracle's SQL Developer. This enabled us to establish connections between our client and PLAN's database and subsequently utilize SQL queries to view the contents of each table at the time the query was executed.

Next, we capitalized on SQL Developer's functionality to manually extract query results, generating CSV files containing the most relevant tables for both of our use cases. For instance, a query such as *SELECT * FROM plan.rm_activity_list;* would return every entry in the Activity List table. Subsequently, a CSV file would be generated from these results for further treatment and preprocessing using Python. This process is illustrated in Figure 4.1.

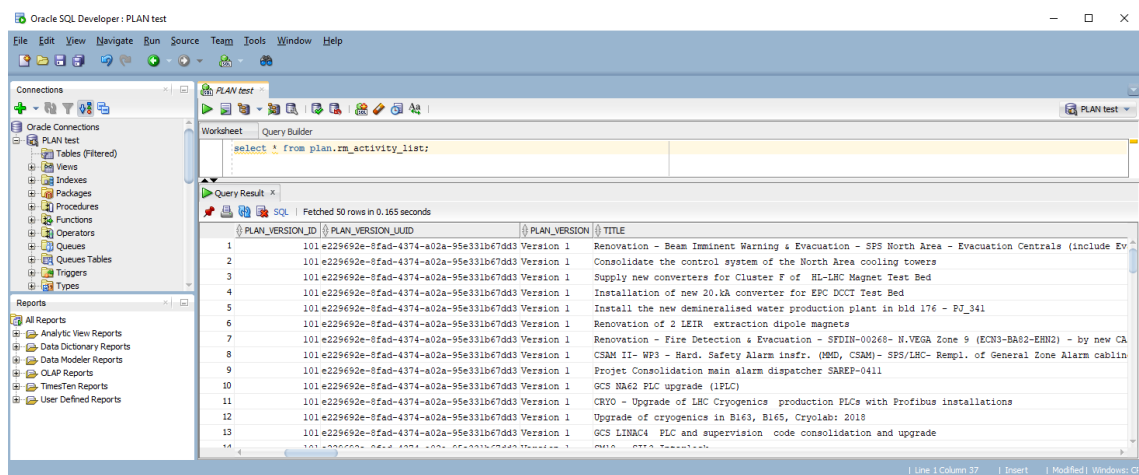


Figure 4.1: PLAN Table access example

By employing this approach, we ensured that the data extraction process was both systematic and comprehensive. This foundation enabled us to accurately preprocess and analyze the extracted data, ultimately leading to more reliable insights and conclusions in our study.

4.1.2 Activity Dataset

The primary data resource for our machine learning algorithms was historical data on past activities. This dataset was going to serve as the foundational basis for all the algorithms we developed.

To create an optimal CSV file for our activity data, it was necessary to determine the fields that would be most pertinent to our algorithms. This decision was made based on both arbitrary and logical criteria. The final dataset included every field displayed in the activity creation interface, as depicted in Figure 3.5.

The columns designated for our activity.csv file included:

- **ACTIVITY_UUID:** The unique identifier of the activity;
- **TITLE:** The title of the activity (Free text);
- **GROUP_RESPONSIBLE_NAME:** The CERN team responsible for the activity (Categorical);
- **RESPONSIBLE_WITH_DETAILS:** The individual responsible for the activity and their associated CERN team details (Categorical);
- **ACTIVITY_TYPE_EN:** The category of the activity, in English (Categorical);
- **WBS_NODE_CODE:** The project or study that the activity is associated with (Categorical);
- **FACILITY_NAMES:** The names of the facilities where the activity will take place (List of Categorical Values);

- **CREATOR_NAME:** The name of the individual who created the activity (Categorical);
- **CREATION_DATE:** The date when the activity was created (Date format string);
- **LOCATION_INFORMATION:** The location where the activity will occur (Free text);
- **GOAL:** The objective of the activity (Free text);
- **IMPACT_NOT_DONE:** The potential impact if the activity is not completed (Free text).

After identifying the necessary fields for our final dataset, we initiated the processing, merging, and transformation of the original tables.

We scrutinized every activity-related PLAN table, starting by eliminating columns that would not contribute to the creation of our final activity dataset. This could be because they contained only one unique value or because they were deemed logically irrelevant to the final dataset. Information regarding contributions and the duration of the activity were omitted, for example.

Then, after merging the resulting datasets based on their IDs, we obtained our final dataset. An extract of said dataset is depicted in Figure 4.2.

ACTIVITY_ID	TITLE	GROUP_RE	RESPONSIBLE_WI	ACTIVITY_TYPE_EN	WBS_NOD	FACILITY_NAM	CREATOR_NAM	CREATION	LOCATION_INFOR	GOAL	IMPACT_NOT_DONE
8033baff-7c	New liquefier in P	TE-CRG	LIONEL HERBLIN (TI	Consolidation & upgr	NONE	LHC Machine	SYSTEM ADMINI	19-NOV-21		Improve the man	Reduced flexibility in h
81578ada-9	Short Circuit Test	TE-MPE	MIRKO POJER (TE-N	Consolidation & upgr	NONE	LHC Machine	SYSTEM ADMINI	19-NOV-21	The tests will be pe	The objective of t	The bad cooling of the
8572a050-7	Profound mainte	SY-RF	SUITBERT RAMBER	Consolidation & upgr	ACCONS	ISOLDE	SYSTEM ADMINI	19-NOV-21	Amplifier room of	Work required to	The profound mainten
85798a58-4	SM18 integrate a	BE-ICS	LUKASZ JACEK GOR	Consolidation & upgr	OTHER	Buildings And O	SYSTEM ADMINI	19-NOV-21		SM18 - "Overvie	TBD
8828ec7b-3	Linac4 LEBT Redet	BE-ABP	ALESSANDRA LOMI	Consolidation & upgr	OTHER	LINAC 4	SYSTEM ADMINI	19-NOV-21		Optimise beam tr	Difficulty to maintain ir

Figure 4.2: Activity Dataset

At the conclusion of this process, our dataset encompassed 1842 activities, corresponding to the number of activities created since the beginning of RUN3-LS3. This reaffirmed CERN's assertion and one of the bigger challenges of this project: the available data was considerably limited for the application of machine learning techniques. As a result, techniques from section 2.5 were going to be necessitated for further exploration and application.

4.1.3 Contribution Dataset

In order to address our second use case (Section 3.2.2), we needed to create another dataset - this time focusing on contributions made to past activities.

Our methodology in the creation of this dataset paralleled the approach used in section 4.1.2. Initially, we had to identify the fields that would be of value for our contribution suggestion algorithms. This decision was informed by an examination of the PLAN tool's interface, particularly the resources tab, where contributions are recorded and characterized, as depicted in Figure 3.6. The two key defining characteristics of contributions were identified as being the Group / Section and the Contribution Type.

In addition to these, several other fields were considered to potentially hold value for future usage:

- **ID:** The unique identifier of the contribution;

- **ACTIVITY_UUID:** The unique identifier of the activity corresponding to that contribution;
- **DESCRIPTION:** The description of the contribution (Free text);
- **PHASE_NAME:** The name of the phase associated with that contribution: Preparation, Installation, Commissioning or None (Categorical);
- **ORG_UNIT_CODE:** The team that is making the contribution (Categorical);
- **ORG_UNIT_TYPE:** The type of the team making the contributions: Group or Section (Categorical);
- **STATUS:** The status of the contribution (Categorical);
- **CONTRIBUTION_TYPE:** The type of the contribution (Categorical).

After determining these parameters, we proceeded to discard the unnecessary columns from the extracted contribution-related PLAN tables. Subsequently, the remaining datasets were merged.

The final determination of the team name for inclusion in the final dataset required a more nuanced approach. We had to scrutinize the merged contribution dataset and, for each entry, check whether the contribution was made by a group or a section. Only then could we fetch the name of the corresponding team. Upon completion of all merges, we finalized our contributions dataset, as shown in Figure 4.3, where we see a sample of it.

ID	ACTIVITY_U	DESCRIPTION	PHASE_NAME	ORG_UNIT_C	ORG_UNIT_T	STATUS	CONTRI
6c2ad23e-7	8033baff-701			TE-CRG	GROUP	A	Other
4fbd3870-e	81578ada-99			TE-MPE	GROUP	A	Other
5a7e95fc-f	81578ada-99	Planning and gener	Preparation	EN-ACE-OSS	SECT	A	Other
20cca00-2	81578ada-99	Production of detec	Preparation	TE-MPE-EP	SECT	A	Other
a4333f16-a	81578ada-99	Supply of material	Preparation	TE-MPE-MP	SECT	A	Other

Figure 4.3: Contribution Dataset

Our resulting dataset comprised 11399 contributions, a figure significantly larger than the number of entries in our activity dataset. However, this greater quantity did not alleviate the limitations imposed by our initial low data problem. Despite this new dataset's focus on contributions, the foundational premise of our second use case was the comparison of a new activity with previous ones. Thus, the use of our smaller activity dataset remained an unavoidable necessity.

4.2 Initial Approach

When first confronted with the problem at hand, further described in section 3.2, the first technique that we came up with to develop the suggestion system for PLAN was a hybrid method, comprised of a clustering model followed up by a supervised learning algorithm.

Clustering methods would help divide and split our existing historical data on activities into clusters or "groups" by finding underlying patterns and structures in the data.

From there, supervised learning techniques with the Cluster label of each activity as the target value would help to accurately predict the group or cluster a new activity would belong to.

Finally, suggestions could be made from the assigned cluster by simply fetching the most common values from a certain variable within the cluster or by any other suitable approach.

In this present section, we will dive into more detail on the implementation of the clustering methods (section 4.2.1) to our data and to the training of a supervised learning model to predict the cluster label for a new activity in 4.2.2. We will end the explanation of our model by explaining how the values are suggested, in section 4.2.3. We will follow with an explanation on how this approach was evaluated in section 4.2.4 and the challenges and lessons learned in the implementation of the hybrid method (4.2.5).

4.2.1 Clustering Methods

The first step in this approach would be to try and identify patterns and underlying structures in the historical data on PLAN activities so that we can later use the formed groups to make suggestions or predictions.

This would be tackled by a clustering technique (more in section 2.3.1), which is designed to do exactly that, to separate the existing data points into different clusters.

In our case, to implement the clustering, we would have to perform some extra data preparation steps to the activity dataset described in section 4.1.2.

Most of the existing clustering algorithms do not work with missing values (NaNs), as this would make a distance-based algorithm, such as most of the clustering ones, unable to work properly.

Therefore, we need to handle these existing missing values in our relevant fields, the ones listed in section 4.1.2. There are several available ways to handle missing values before applying any clustering algorithm (section 2.1), namely removal, imputation, and adding a missing value indicator.

With regards to PLAN historical data, the removal technique cannot be applied, as one of the main constraints of the project is the fact that we are dealing with limited data, and using this technique would only aggravate this situation.

Also, the presence of missing values can actually mean something when it comes to CERN activities. For example, if the activity is not completed, the fact that that value is empty is actually informative to the system.

But, as this is important for each relevant field and not only for the presence of missing values in the data point in general, we will use an imputation method where we will replace every missing variable in the dataset with the "Unknown" value

Then, we need to determine which are the variables that will be used to characterize and try to group the data. The possible combinations of important variables will be later changed and tested, but to start us off, we decided to cluster the data based on the five existing categorical variables that

would best separate activities into different categories: 'GROUP_RESPONSIBLE_NAME', 'RESPONSIBLE_WITH_DETAILS', 'ACTIVITY_TYPE_EN', 'WBS_NODE_CODE', and finally, the 'CREATOR_NAME'.

From there, we need to encode these variables so that we can experiment with different clustering algorithms, some of which do not function properly with categorical data.

To encode our variables, two main encoding techniques were considered and tested, as described previously in section 2.1: one-hot encoding and label encoding.

In our case, one-hot encoding isn't suitable because some of the selected variables have a high number of categories. This would lead to a very high number of added columns, which wouldn't be ideal. As so, we apply label encoding to the selected relevant columns in the activity dataset.

From here, we can apply different clustering algorithms to determine which, if any, has the best performance and, consequently, will be the one used for our suggestion system.

We will try out two different algorithms: DBSCAN (2.3.1.3) and KModes (2.3.1.2), using Python's scikit-learn and kmodes libraries, respectively.

One focal point of our case is that since all of our initial relevant fields for the clustering are categorical, when experimenting with DBSCAN, the distance calculation used for the clustering cannot be the Euclidean distance since it would assume that the label-encoded variables are closer to each other when the number is closer, and that is not true.

We need a distance metric that can handle categorical values. Revisiting what we discussed in section 2.2.3.2, we decided to experiment with both Gower and Jaccard Dissimilarity distances.

Once we have applied the algorithms to the activity dataset, we have to assess their performance. Starting with comparing which clustering algorithm has the best performance and then trying to improve it, be it by testing what is the optimal number of clusters or testing out different combinations of relevant values. Finally, we have to determine if the clustering is actually good enough for future use in our PLAN suggestion system. This analysis will be carried out in section 4.2.4.

From there, we would use the best clustering algorithm to assign each row in the activity dataset to a cluster label, which will be a new column in the dataset, that will serve as the target in the supervised learning techniques that we will now describe in section 4.2.2.

4.2.2 Supervised Learning Techniques

Following the integration of an additional column in the activity dataset - which is dedicated to the cluster number each row has been assigned to during the clustering process - we transition to the subsequent step in our pipeline. This involves applying a supervised learning algorithm to the dataset. The goal of this step is to equip our model with the capability to assign a new activity to a particular cluster that we would then use for the suggestions.

To initiate the application of a supervised learning algorithm, we first define the features and the target variable. As indicated in previous discussions, our target variable is the cluster identifier that emerged from the clustering process. The relevant features for this process mirror those utilized in the clustering operation, which have been elaborated in section 4.2.1. Upon establishing

these variables, we proceed to partition the data into training and testing sets, adhering to an 80% - 20% split, respectively.

In our study, we experimented with two distinct classifiers, namely, Decision Tree (2.2.1) and Random Forest algorithms (2.2.2). For each of these classifiers, the prescribed method involves fitting the classifier to the training data, followed by its evaluation on the testing set. Our supervised learning model is now able to determine a cluster label for a new activity.

It is essential to underscore that, analogous to the clustering algorithms, supervised learning algorithms require comparative analysis and evaluation. This forms an integral part of our research, which we will further explore and discuss in section 4.2.4.

4.2.3 Evaluation Method

Having meticulously implemented and rigorously tested the two pillars of our hybrid method, we proceeded to the stage of making suggestions. To reiterate our earlier explanation, we would start by applying in real-time our supervised model to the new activity that is being created in the PLAN platform. Then, we would use the cluster group assigned to fetch the most common variables within the cluster for the fields that we want to make suggestions on.

We would go through every column in the new activity and, for each missing value, return a list of the 5 values that appear the most for that column in the assigned cluster. These five values are ranked based on their frequency of occurrence within the assigned cluster, thereby ensuring the most relevant suggestions are ranked higher, which ultimately improves user experience and the utility of our system.

To evaluate the effectiveness of our suggestion system, we have developed a specific evaluation mechanism. The function takes as input the test data, the original values of the variables, the complete dataset, and the columns for which we are making suggestions. For each iteration of the evaluation method, we create a test data set. This function generates a random sample from the data and then randomly removes values from the suggestion columns in each activity, which will be replaced with the "Unknown" label when making the cluster prediction. The randomness of the missing columns will certainly compromise to some extent the cluster label prediction, but this process is done to mirror the real-world scenario where some information might not be available when creating a new activity.

Then, we apply our suggestion system to this test set, which produces a set of suggested labels for the missing values. Our evaluation metrics, specifically precision, recall, and F1 score (section 2.2.4), are then calculated based on these suggested labels and the original values. The precision metric is weighted using a function that assigns higher importance to labels that are suggested first in the list, simulating the real-world usage of the suggestion system where earlier suggestions carry more weight.

This entire process provides a comprehensive evaluation of our suggestion system's performance, giving us insights into the accuracy and efficiency of the model. It is this evaluation mechanism that guides our refinement and optimization efforts, ultimately enhancing the quality and efficacy of our hybrid approach.

We will expand on the results of our suggestion system performance after analyzing the performance of each step in the hybrid approach, in the Performance Evaluation section (4.2.4).

4.2.4 Performance Evaluation

To better understand the performance of our final solution, we will start by analyzing the performance of the clustering model (4.2.4.1), then of our supervised learning classifier (4.2.4.2) and finally the overall performance of the suggestion system and its success suggesting useful values, in section 4.2.4.3.

4.2.4.1 Clustering Performance

To analyze the performance of our clustering algorithms, we used two different methods for evaluating the quality of a clustering solution: the Davies-Bouldin Index (DBI) and the Silhouette Score. Both of these methods are explained into detail in section 2.3.3.

When comparing the performance of the DBSCAN and the KModes algorithms, we quickly realized that the KModes algorithm was the more appropriate one for this problem. We will now dive into more detail on its performance.

The DBI measures the average similarity between each cluster and the cluster that is most similar to it. Lower DBI values indicate better data partitioning since they imply that the data points inside each cluster are close to one another and that the clusters are well-separated (i.e., there is little overlap between them). As mentioned, this is further explored in section 2.3.3.

Our model had a DBI score of 2.720. As 0 indicates perfect clustering, where the clusters are clearly separated between them and the data points inside each cluster are close to each other, a value of 2.720 isn't ideal. However, as there is no clear upper bound in this index, we can't draw any conclusion at this stage. As so, we will explore the silhouette score coefficient while keeping in mind that the DBI informed us that the clustering solution might not be the best.

Once again, as described in section 2.3.3, the silhouette score is another method for assessing the quality of a clustering solution. It quantifies how well each object lies within its cluster, and a higher silhouette score indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters.

It ranges from -1 to 1, where 1 indicates a good clustering solution, 0 shows us that clusters overlap, and -1 means that there are blatantly wrong cluster assignments in the solution developed. Our KModes model got a score of 0.142, which is really close to 0. This does once again indicate that the developed clustering model isn't very good. We also analyzed the silhouette plot of our 12-cluster model to verify this theory.

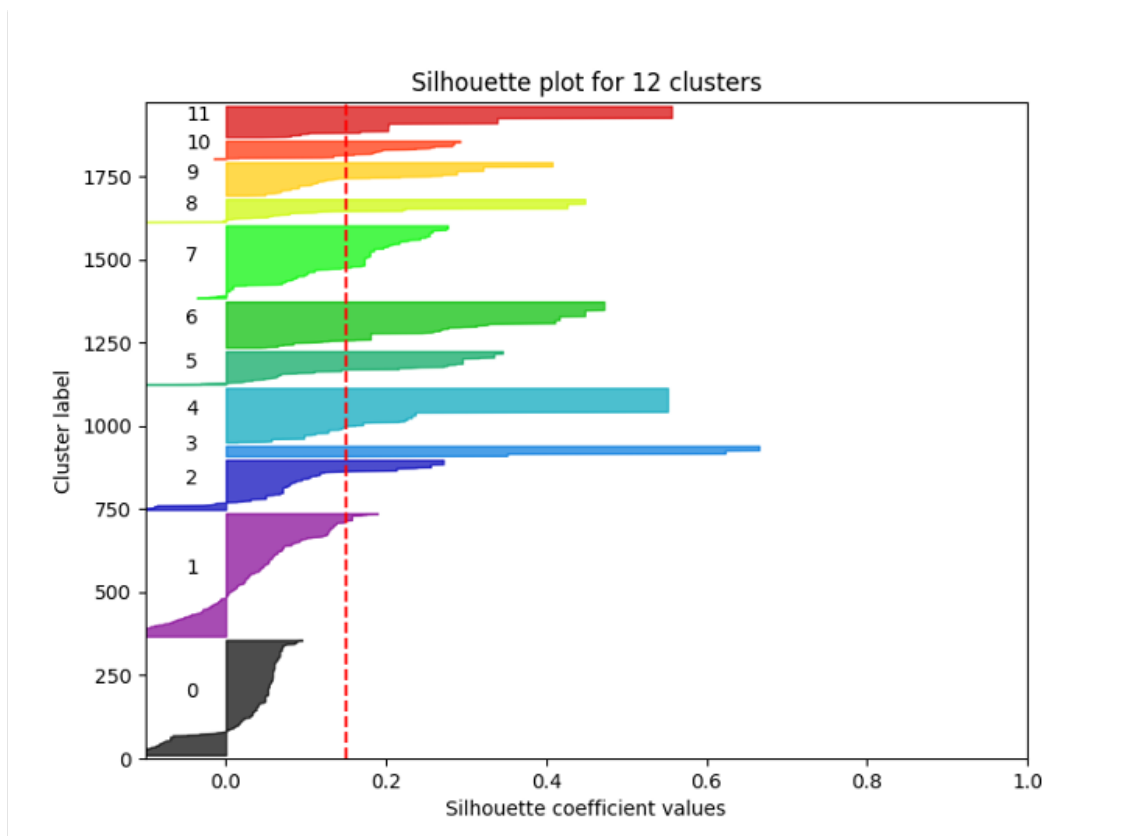


Figure 4.4: Clustering Solution's Silhouette Plot

As we can see and as it was explained in section 2.3.3, other than the fact that the mean value of the silhouette score for our model is low, the size of each cluster (this can be seen on the y-axis of the graph) is very different from cluster to cluster, which is not a good sign. Also, some clusters even have data points that have a silhouette score below 0.0 (this can be seen on the x-axis), which indicates that there are points that are simply on the wrong cluster.

In addition, the silhouette score kept getting better the fewer fields we used for the clustering and also kept getting exponentially better the bigger the number of clusters we chose. This may indicate that the activity data does not have clear groups of data, therefore no good clustering solution can be achieved.

Overall, we can conclude that the model's performance is not good enough for use and that another solution will probably have to be found in the future.

4.2.4.2 Supervised Learning Performance

Even though the clustering model seems to be underperforming, we still decided to implement the hybrid method until the suggestion phase. As so, the next step in the process was the supervised learning techniques.

We used the three metrics described in section 2.2.4: Precision, Recall, and F1-score to evaluate our two models. When using the decision tree classifier, we got, respectively: 0.924, 0.921,

0.920. With random forest, we had the values of 0.814, 0.802, and 0.792, making the decision tree algorithm the best suited for our supervised learning part of the method.

It is understandable that the metrics return such good results because the supervised learning model uses the same fields that the clustering used to determine the labels, to assign a label to a new activity.

Although the decision tree algorithm has a very good performance in assigning a new activity to its cluster label, we cannot forget that these cluster labels come from a faulty clustering model, described in the previous section 4.2.4.1.

4.2.4.3 Suggestion System Performance

In order to robustly assess the efficacy of our proposed approach, it is crucial to examine the degree to which our recommendation system provides pertinent and beneficial suggestions to the users when they are involved in creating a new activity. To facilitate this, we will utilize the evaluation system outlined previously in section 4.2.3.

As explained, our evaluation is centered around a comparative analysis of the quintuple-value lists that our system recommends to the user versus the verifiable true values pertaining to the fields in question. This examination, facilitated by our hybrid methodology, results in a precision of 0.353, a recall of 0.473, and an F1-score of 0.404. These figures encapsulate the efficacy of our system: it is capable of accurately identifying and suggesting 35% of the true values from the entire pool of suggestions it generates, and it manages to capture 47% of all existing true values. The F1-score, which offers a harmonic mean of precision and recall, indicates that our system's effectiveness in providing suggestions stands at a level of 40%.

These numbers are not adequate for delivery and implementation in CERN's planning tool for organization-wide use. This subpar performance is most likely attributable to the ineffective clustering within our system. It emphasizes the need for a revision of our approach to ensure a more efficient and accurate suggestion mechanism.

In light of these results, we will now explore what conclusions can be derived from this particular implementation and its ensuing outcomes. Our objective is to navigate towards a more fruitful approach for our recommendation system, taking into consideration these invaluable lessons learned from the current model's performance, which we will expand on in the next section.

4.2.5 Challenges and Lessons Learned

The development and implementation of our hybrid model for providing suggestions has revealed significant challenges and lessons, especially when dealing with datasets of low volume, as is our case. These challenges and lessons have profound implications for our approach to the current problem.

Clustering algorithms strive to identify structures or patterns within datasets by grouping similar data points together. However, when dealing with low-volume data, the ability of these algorithms to accurately discern patterns and distinguish clusters can be significantly impaired. Our

clustering model, for instance, exhibited a subpar performance, as evidenced by its Davies-Bouldin Index score of 2.720 and silhouette score of 0.142. This unsatisfactory clustering quality is likely attributable to the lack of sufficient data needed for the algorithm to correctly partition the data points into distinct groups.

This situation highlights the limitations of certain AI models when dealing with low data volume and it is evident that a different approach is necessary, one that is specifically designed or better suited to work effectively with limited data. This points us towards exploring machine learning techniques that do not heavily rely on large amounts of data, or algorithms that are better equipped to handle sparse data.

The lessons learned from this initial phase are invaluable. It highlights the significance of having a thorough understanding of the data we're dealing with and the suitability of the chosen machine learning models in relation to the volume and quality of data.

Going forward, we will use these insights to refine our approach. We will explore methodologies that are known to perform better with sparse data and will continuously evaluate and refine our models to improve their performance.

4.3 Final Algorithm Development and Optimization

After arriving at CERN, getting familiar with the available data, and realizing that it wasn't enough to develop a satisfactory clustering solution, we had to figure out a different way of grouping the activities. This was essential for us to tackle both of the problems described in section 3.2. For the first use case (3.2.1), we would then use these groups of similar activities to retrieve the most common values for each of the missing fields in the activity that is being created in PLAN. For the second one (3.2.2), similarly, those groups of activities and their associated contributions could be used to fetch the most probable contributions.

Our final algorithm designed to tackle this problem employs a combination of k-Nearest Neighbors (2.2.3.1) and Natural Language Processing techniques (2.4). The k-NN algorithm, an instance-based learning algorithm, will allow us to find the most similar activities in the dataset for a new one based on different criteria, making the most out of the limited data.

Instance-based learning (section 2.5), also known as lazy learning, is a type of machine learning algorithm that learns directly from the training instances themselves in real-time, without explicitly building a generalizing model. Instead, it stores the training instances and uses them as a reference to find similar instances. In the case of k-NN, the algorithm measures the similarity between instances based on their features and assigns the label or value of the nearest neighbors to the new instance.

In our scenario, where the quantity of data is limited, k-NN is particularly suitable. By leveraging the available instances as references, k-NN can effectively identify activities that are similar to a new one, even with a small dataset. It doesn't rely on assumptions about the underlying distribution of the data, making it a flexible approach that can handle diverse patterns and characteristics.

By combining the strengths of k-NN with NLP techniques, which extract semantic similarity from activity descriptions and titles, we enhance the similarity measurement and improve the accuracy of our suggestions. The use of instance-based learning allows us to overcome the data constraints and provide a practical, data-efficient solution within the PLAN system.

This section will provide more detail into the specifics of this approach, starting with how we designed our algorithm to find the similar activities for a new one (section 4.3.1), and culminating in the mechanisms of suggestion generation for both of our problems in section 4.3.2. In section 4.3.3, we will describe the evaluation methods used to assess the performance of our system. Then, we will explain the process of improving and tuning the hyperparameters used in the algorithm in section 4.3.4. We will finish the section by analyzing the performance and accuracy of our model (4.3.5).

4.3.1 Similar activity grouping

To tackle our activity grouping need, we will use a combination of k-NN and NLP. The combination of both of these measures will not only allow us to dynamically create groups of activities that are similar to the new one but will also allow us to actually rank activities based on how close they are to the new one.

As mentioned, the groups of activities will serve as a base for both of our use cases. However, the first one has some extra challenges that require this method to be more nuanced. In the first use case, since we want the suggestions to be presented to the user in real-time, the model needs to be able to handle different amounts of data already entered by the user. Ideally, we even want it to be able to make suggestions when nothing has been entered, solely based on the user that is creating the activity.

Going back to the interface the user is presented with when creating a new activity in PLAN (3.5), once again, we defined the categorical columns 'GROUP_RESPONSIBLE_NAME', 'RESPONSIBLE_WITH_DETAILS', 'ACTIVITY_TYPE_EN', 'WBS_NODE_CODE', and 'CREATOR_NAME' as being the relevant fields that we will use for our k-NN implementation. To tackle our varying data issue, before applying k-NN, we go through these fields for our new activity, and we remove every field that is missing from the distance calculation. By doing so, the nearest neighbors will be calculated based only on the information we have. It is also important to note that the 'CREATOR_NAME' field will always be present since it corresponds to the name of the user. This is also the information we will use when the user still hasn't entered any information for the new activity.

Once we know what fields are to be used for the k-NN algorithm, our system elaborates a distance matrix with the calculation of the Gower distances (explained in section 4.2.1) of the new activity to all the existing activities in the dataset. This will give us a measure of how dissimilar the new activity and every other past activity are. The returned values range from 0 to 1, with 0 meaning that the activities are identical.

However, this dissimilarity calculation is not enough to accurately retrieve all of the most similar activities to the new one. The 'TITLE' field is probably the piece of information that best

describes the activities, and we are not using it yet. Not only might it be the most descriptive information, but it is also most of the time the first information the user will enter in PLAN, and we want the suggestions made by our system to be accurate and useful as soon as possible.

The 'TITLE' field is a free text variable, so it can't be used in the k-NN calculation like the categorical variables. This is where we integrated Natural Language Processing techniques into our system. When the 'TITLE' is available, we compute the Term Frequency-Inverse Document Frequency (TF-IDF) (2.4.1) embeddings for the 'TITLE' field of all existing activities. Here, TF-IDF is a numerical representation of data, that captures the importance of each word in the document with the title for the new activity relative to the collection of documents of the titles from past activities. We exclude common English stop words from this vectorization step.

In our case, the decision to numerically represent text with TF-IDF over a more complex word embedding technique was made because TF-IDF is a simpler approach, which is faster and computationally less expensive. Even though a word embedding technique would capture the semantic meaning of words, and as the 'TITLE' field contains short phrases, a method that uses less computational resources, such as TD-IDF, is suitable.

Once both our entities have been embedded, we use the cosine similarity (2.4.2) to measure the similarity between the new activity's 'TITLE' embedding and each existing activity's 'TITLE' embedding. We now have a second similarity matrix, related to the 'TITLE' field. The cosine similarity scores range from -1 to 1, where 1 indicates maximum similarity between the vectors and -1 indicates maximum dissimilarity.

To obtain the final hybrid similarity measure for activities, we combine the Gower distance matrix and the cosine similarity matrix. In order to ensure that the two scores have the same range, we reshape the cosine similarity matrix and convert the Gower matrix to a similarity measure. The reshaping process involves scaling the cosine similarity values to a range of 0 to 1 by adding 1 to each value and dividing the result by 2. For the Gower matrix, we transform it into a similarity measure by subtracting each value from 1. These transformations allow us to have both matrices in the range of 0 to 1, where higher values indicate greater similarity.

After aligning both metrics to have a consistent format, the next step is to merge them to create a unified similarity score. This requires deciding the weight assigned to each metric in the combination process. As previously mentioned, the 'TITLE' field holds significant importance, leading us to consider giving a higher weight to the cosine similarity score. However, in order to evaluate the system's performance objectively, an analytical assessment will be conducted using various weight combinations in section 4.3.4.

It is important to note that when only the 'TITLE' field is available, only the cosine similarity results are utilized, and conversely when the 'TITLE' has not yet been entered, the Gower results from the k-NN method are used.

Once the final hybrid similarity score is obtained, the k-nearest neighbors are determined by identifying the indices with the highest values in the hybrid matrix. Subsequently, the k-nearest neighbors are retrieved from the original dataset and organized based on their similarity scores. In cases where ties exist in the similarity score, an additional sorting criterion is applied, using the

CREATION_DATE of the activities. This is done to prioritize more recent activities, as they might be more relevant and useful than older ones.

After completing the previous steps, we have successfully obtained a set of similar activities. The size of these activity groups can be adjusted in our final algorithm based on specific requirements. However, similar to the analysis conducted for the weights of the matrices, we will also perform an evaluation to determine the optimal number of activities that yields the best results.

Through this evaluative process (section 4.3.4), we aim to refine and fine-tune our recommendation system to ensure it delivers the most satisfactory outcomes for users. By considering both the weight combinations of the matrices and the appropriate number of activities, we can enhance the overall performance and utility of the suggestion system, later explored in section 4.3.5.

Having discussed the foundational aspects of our suggestion system, we will now shift our focus toward a practical perspective. In the next section, 4.3.2, we will explore how our system can actually suggest values for the two problems under consideration: suggestions for activity information (4.3.2.1) and contribution suggestion (4.3.2.2).

4.3.2 Suggestion Generation

As mentioned, in the present section we will describe how our system takes the groups of similar activities generated from our k-NN and NLP model described in the previous section and uses it to generate suggestions on the PLAN platform.

To start us off, in section 4.3.2.1, we will describe what our model is able to do to solve our first use case, where we want PLAN to be able to somehow help the user at the moment where he is creating a new activity. In the following section, 4.3.2.2, we dive into how our model helps the PLAN user in requesting contributions from other CERN teams, for an already created activity. We will provide some examples for each use case.

4.3.2.1 Activity Suggestion

When a user is creating an activity in PLAN, our model intends to play a crucial role in facilitating the process. To satisfy this requirement, we have implemented two different methods within our suggestion system. The first method involves simply presenting the user with a ranked list of similar activities.

Based on the desired number of suggested past activities, the model returns a ranked group of activities, as described in section 4.3.1. This allows for a more personalized approach by CERN, as the planning platform can later use the group of activities in various ways, be it by cloning some fields, an entire activity, or giving options to the users, among others.

Let's consider an example of an activity in the creation state within the PLAN system, where the Title, Type of the activity, and Responsible Group have already been entered by the user, in Figure 4.5.

Figure 4.5: Activity Creation Example

As explained in section 4.3.1, the model would consider the three entered fields and the name of the user that is creating the activity (in this example, it was myself, part of the EN-ACE-OSS group) and create a list of the most similar activities. Assuming we have selected to retrieve the ten nearest neighbors, in this case, the system would return the following activities.

ACTIVITY	TITLE	GROUP	RESPONSIBLE_WITH_DETAILS	ACTIVITY	WBS_NOI	FACILITY	PRIORITY	CREATOR_NAME	LOCATION	GOAL	IMPACT_NOT_DONE
8a1c176e	Injectors De-cabling phase 2 project - PS ring (2 more octants)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	PS	1. Safety	FERNANDO BALTA	PS Ring	Reduce the fire	In case of Fire, all this f
501bb340	Injectors De-cabling phase 2 project - PS ring (remaining octants)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	PS	1. Safety	FERNANDO BALTA	PS Ring	Reduce the fire	In case of Fire, all this f
3348ee2d	Injectors De-cabling phase 2 project - PS ring (2nd and 3rd octants)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	PS	1. Safety	FERNANDO BALTA	PS Ring	Reduce the fire	In case of Fire, all this f
01f0cedd	Injectors De-cabling phase 2 project - PS ring (1st octant)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	PS	1. Safety	FERNANDO BALTA	LINAC3	Reduce the fire	In case of Fire, all this f
19094632	Injectors De-cabling phase 2 project - LS51 de-cabling	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	SPS	1. Safety	FERNANDO BALTA	LS51 right	Reduce the fire	In case of Fire, all this f
9a69df4f	Injectors De-cabling phase 2 project - SP52 (BA2, Shaft and TA2)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	SPS	1. Safety	FERNANDO BALTA	BA2Shaft	Reduce the fire	In case of Fire, all this f
608b1bdd	Injectors De-cabling phase 2 project - SP54 (BA4, Shaft and TA4)	EN-ACE	FERNANDO BALTASAR DOS SAN	Safety	OTHER	SPS	1. Safety	FERNANDO BALTA	BA4Shaft	Reduce the fire	In case of Fire, all this f
e13d7fe5	Ventilation and fire resistant door HL LHC 1 -US lift SAS	EN-ACE	SERGE GRILLOT (EN-ACE-COS)	Safety	HL-17	HL-LHC	3. HL-LHC	SERGE GRILLOT (EN	US17	HL-LHC	As these door are need
32ea18ce	Fire resistant door HL LHC 5 - UR	EN-ACE	SERGE GRILLOT (EN-ACE-COS)	Safety	HL-17	HL-LHC	3. HL-LHC	SERGE GRILLOT (EN	UR55	HL-LHC	As these door are need
f4dfbbd6	Fire & pressure-resistant door HL LHC 5 -UL	EN-ACE	SERGE GRILLOT (EN-ACE-COS)	Safety	HL-17	HL-LHC	3. HL-LHC	SERGE GRILLOT (EN	UL53UL5	HL-LHC	As these door are need

Figure 4.6: Returned activities

The PLAN platform can then offer the user the option to select and clone an activity entirely or clone specific missing values such as the person responsible, facilities, and more. This flexibility empowers users to leverage the suggested activities according to their specific requirements.

On the other hand, another possibility offered by our model is to directly suggest values for the missing fields in the activity. Here, instead of directly returning the list of similar activities, the model uses it as a group. It iterates through the relevant variables in the activity being created and calculates the most common instances within the group, considering the categorical nature of the relevant fields. The model then generates ranked lists of those common values for each missing piece of information. The number of returned suggestions is configurable to allow for PLAN developers to later configure it as they wish.

If we take the activity illustrated in Figure 4.5, the system would suggest the following values:

```
Ordered suggestions for the missing information on the new activity:

RESPONSIBLE_WITH_DETAILS:
  Suggestion 1: FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)
  Suggestion 2: SERGE GRILLOT (EN-ACE-COS)

WBS_NODE_CODE:
  Suggestion 1: OTHER
  Suggestion 2: HL-17
```

Figure 4.7: Suggested Activity Fields

Once again, from here, the user could select the suggested values for the new activity without having the need for manual entry.

In the subsequent sections, we will dive into the specifics of how the suggestion system enhances the user experience when requesting contributions from other CERN teams. The system provides intuitive and efficient suggestions to streamline the process and improve overall efficiency.

4.3.2.2 Contribution Suggestion

As detailed in section 3.2.2, the second scenario involves contributions to various activities. Whenever an activity is set up on the platform, as shown in the resources tab (3.6), any approved user can send contribution requests to different sections or departments at CERN. Much like in the previous section, our model introduces two features that help make this process quicker and easier.

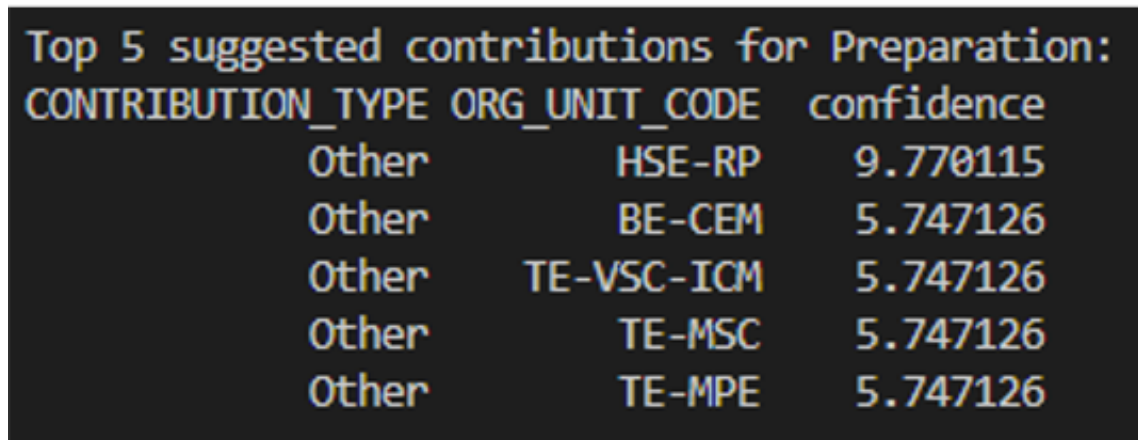
Our model, based on the general information about the activity entered in the general tab (as displayed in Figure 3.5), is capable of suggesting the most suitable contributions for the activity. These suggestions are offered for each of the three phases: Preparation, Installation, and Contribution.

Here, as we don't have to deal with missing information for the important fields of the similarity score calculation (4.3.1), the model can provide a more accurate list of similar activities. From this list, for each phase, the model pulls together all the contributions related to the similar activities, from the contribution dataset (4.1.3). It groups them by contribution type and by their group or section and counts the number of occurrences of each pair. The algorithm is now able to return a list of configurable size with the most common contributions for the activities similar to the activity the user is trying to create contributions requests on.

Going back to the example provided in Figure 4.5, in the case where the user finished creating the activity and filled in the missing relevant variables as follows:

- **Responsible:** FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)
- **Project/Study:** OTHER

Our model would take the information from the created activity and use it to generate three different ordered lists of contributions, one for each activity phase, as we can see in Figure 4.8.



```

Top 5 suggested contributions for Preparation:
CONTRIBUTION_TYPE ORG_UNIT_CODE confidence
Other              HSE-RP          9.770115
Other              BE-CEM          5.747126
Other              TE-VSC-ICM     5.747126
Other              TE-MSC          5.747126
Other              TE-MPE          5.747126

```

Figure 4.8: Suggested Contributions for Preparation Phase

Suppose we're interested in identifying the top five potential contributions. In this case, the model displays the most frequently observed combinations of 'CONTRIBUTION_TYPE' and 'ORG_UNIT_CODE' from the activities most closely related to the one in question. Beyond this, the model also provides information about the confidence level for each contribution, which indicates how likely each contribution is to be beneficial to the user.

This confidence level is essentially the percentage of times that a particular contribution was present among the contributions associated with similar activities for the given phase (in this case, Preparation). Therefore, a higher confidence level means that the contribution has been found useful more often in similar contexts, suggesting that it could also be useful in the current situation.

However, consider a situation where the user finds one of the suggested contributions useful and decides to select it. Alternatively, the user might have already manually entered some contributions they believe are necessary for the particular activity. In such cases, it would not be logical for the model to continue making suggestions based solely on the description of the activity while overlooking the contributions that have already been input. This emphasizes the importance of the model's adaptability, being able to factor in existing inputs when generating further suggestions.

This is where our second contribution-related feature comes in, and where association rules (2.3.2) become extremely useful. Association rules are techniques often employed in data mining to uncover interesting relationships or correlations between variables in large datasets. Given that we have a situation where the user has already chosen certain contributions, these rules can help in identifying other contributions that are frequently associated or observed with those already chosen in similar activities across the contributions dataset.

By doing so, the model, armed with the insights provided by association rules, can suggest contributions that are not only in line with the activity description but also with the already selected contributions, thereby making the suggestions more contextual and relevant to the user.

In our case, when provided with information on an activity and a list of selected contributions by the user, the model will combine two methods. The one we have described previously for contribution suggestion, based on the retrieved similar activities and their most common contributions, and an association rules method.

In the context of this analysis, the model begins its operation by constructing a broad set of association rules. These rules are derived from the comprehensive contribution dataset, and are constrained by predetermined minimum thresholds for both support and confidence. The initial step of this procedure involves the elimination of any data points in the contributions dataset that possess missing values in any of the following fields: phase, section or group, and contribution type. Following this, we proceed to categorize the dataset by constructing unique, three-dimensional tuples that encapsulate the three aforementioned variables. In essence, this is our approach for distinguishing contributions. This results in a dataset that encapsulates every type of contribution.

In order to make this dataset compatible with association rule algorithms, we employ a technique known as one-hot encoding, as referenced in section 4.2.1. Upon successfully transforming the dataset, the model proceeds to apply the apriori algorithm, outlined in section 2.3.2, culminating in the establishment of the association rules.

Upon the establishment of this set of rules, the model initiates an iterative process, sequentially examining each association rule. Within each iteration, the model identifies rules whose antecedents - the 'if' part of an 'if-then' rule - align with the contributions already entered by the user.

When the model discovers such a match, it promptly stores the rule's consequent - the 'then' part of the 'if-then' rule - alongside its associated confidence level. This confidence level is a statistical metric indicating the rule's reliability or how frequently items in the antecedent and consequent occur together.

Then, a configurable size list is generated as before, only this time we prioritize the suggestions coming from matching association rules. Those selections get put at the top of each phase's list, followed by the suggestions retrieved from the similar activities. This is done arbitrarily because, to suggest contributions, we want to give more importance to suggestions obtained from already entered contributions than to the ones coming from activity information.

Let's say that when presented with the suggestions from Figure 4.8, the user finds the contribution from section HSE-RP useful and selects it. The set of association rules is generated from the contributions dataset, and the model looks for matching contributions. It just so happens that one of the rules has that same antecedent, as seen in Figure 4.9.

antecedents	((Preparation, Other, HSE-RP))
consequents	((Installation, Other, EN-HE-HH))
antecedent support	0.055245
consequent support	0.485413
support	0.031657
confidence	0.573034
lift	1.180508
leverage	0.004841
conviction	1.205217
zhangs_metric	0.161848

Figure 4.9: Association Rule Example

In this case, the model then takes the consequent of the rule: a contribution from EN-HE-HH with type Other and adds it to the top of the suggested contributions for its phase: Installation.

Given the dual capabilities of the proposed model, it holds the potential to offer a customized solution that enhances the process of requesting contributions in PLAN. The model does so by providing recommendation suggestions that are predicated not only on the activity description, but also on dynamically responding to the user's real-time input of contributions. In other words, as the user progressively adds contributions, the model actively adapts its suggestions, thereby facilitating the user's decision-making process regarding the requisition of contributions and engendering an overall increase in efficiency.

Nonetheless, to confirm the accuracy of these suggestions beyond theoretical assumptions, it's vital to test and evaluate the model. This will be carried out in the following sections.

4.3.3 Evaluation Method

In the present section, we will describe the methodology used for model evaluation, which includes the calculation of precision, recall, and F1 score (detailed in section 2.2.4).

To evaluate the performance of our suggestion system, two different evaluation methods had to be implemented. The first one is to evaluate the accuracy of our general activity suggestions, and the second one is to evaluate how good our model is in suggesting contributions.

In both cases, the method used is similar to the one described for the initial approach in section 4.2.4.3. However, here we will be using the group of nearest neighbors instead of the cluster. Let's start by describing the evaluation system used for the activity suggestion feature.

4.3.3.1 Activity Suggestion Evaluation

Briefly recapitulating, we start the process by creating a test dataset that is a random subset of the original data frame. It is important to note that, for both evaluation functions, even though we are using instance-based learning, which typically doesn't include a training phase, there is still a concept of training and testing. In our case, we are using all instances except the test instance

to calculate the distances or dissimilarities. Thus, we are using a technique called leave-one-out cross-validation. This is a type of resampling technique where one instance is used as the test set, and the rest are used as the training set. This process is repeated for each instance in the dataset, as we will now describe in further detail.

The resulting subset contains the relevant columns that are used for the suggestion, 'GROUP_RESPONSIBLE_NAME', 'RESPONSIBLE_WITH_DETAILS', 'ACTIVITY_TYPE_EN', 'WBS_NODE_CODE, and the 'CREATOR_NAME'. Once again, a random number of values are deliberately removed from each entry in the test data to simulate incomplete data entries, which is the real-world scenario we will be dealing with in PLAN when a user is creating a new activity.

The model then takes each incomplete entry from the subset and generates suggestions for each missing field based on the previously explained k-Nearest Neighbors and Natural Language Processing techniques. The generated suggestions are ranked, with the top suggestion considered the most probable.

To evaluate the performance of the model, the generated suggestions are compared with the original values that were removed from the test data. Precision, recall, and the F1 score are computed as metrics to assess the model's performance.

Instead of simply looking at whether the true value was among the suggested values, the precision calculation also takes into account the position of the true value within the suggested values. The rationale behind this is that the model's performance should be considered better if the true value is at the top of the list of suggested values compared to being at the bottom of the list.

To achieve this, each correct suggestion is assigned a weight based on its position within the list. The weight of a correct suggestion decreases linearly from the first position (highest weight) to the last position (lowest weight). The weighting function calculates a weight for a given position based on the maximum possible position within the list. The weight of a certain position is calculated as follows:

$$weight_{position} = \frac{max_{position} - position}{max_{position}} \quad (4.1)$$

Where *position* corresponds to the position in the list we are calculating the weight for, $weight_{position}$ is the weight of said position and $max_{position}$ is the number of items in the returned list.

When calculating precision, these weights are used to emphasize correct suggestions that were ranked higher in the list. This is done by calculating a weighted sum where each correctly suggested value contributes to the precision based on its calculated weight. This accumulated weighted sum is then divided by the total number of predictions, giving a weighted precision measure.

The recall and F1 score are then calculated using this weighted precision measure. The recall measures the fraction of the true values that were correctly suggested by the model. The F1 score

is a harmonic mean of the weighted precision and recall, serving as a single metric that reflects both the precision and recall of the model. It balances the trade-off between precision and recall, providing a single measure of the model's performance.

The entire process is repeated multiple times to enhance the robustness of the evaluation. The model's performance is then represented by the mean precision, recall, and F1 score across all iterations. This methodology offers a comprehensive assessment of the model's ability to accurately suggest and rank values for the missing fields.

4.3.3.2 Contribution Suggestion Evaluation

To evaluate the performance of the second use case, the contribution suggestion feature, a method slightly different from that of the first feature is employed, mainly due to the unique structure of the data that is processed. The model is engineered to suggest contributions for each of the three different phases of an activity: Preparation, Installation, and Commissioning.

As before, for each activity in a random subset, this time without the need to input missing values, the model generates suggestions of contributions for each phase using the k-Nearest Neighbors algorithm derived from past activity and contribution data.

Each phase of the activity is then evaluated individually. The model's suggestions for a specific phase are compared with the actual contributions for that same phase in the given activity in the test data.

In the context of the PLAN system, not all activities necessarily have contributions recorded for each phase. In such cases, the model's suggestions do not harm the user's experience but simply provide additional information that may not be utilized. Given this, it is deemed unnecessary to penalize the model for suggesting contributors when none exist in the actual data.

The evaluation metrics - precision, recall, and the F1 score - are computed for each phase. As with the first feature, these are weighted to favor contributors who are ranked higher on the suggestion list. This is done to emphasize the importance of not just the accuracy of the suggestions but the order in which they are suggested. The same formula as before, 4.1, is used to determine the weights.

After the individual phase evaluations, the precision, recall, and F1 scores for each phase are averaged to provide an overall score for the activity. This provides a comprehensive measure of the model's performance for a given activity.

This process is repeated multiple times, with different subsets of activities for each iteration. The model's final performance is then represented by the mean precision, recall, and F1 score across all these iterations. This method provides a detailed evaluation of the contribution suggestion feature, considering both the correctness and the order of the suggestions across all activity phases.

These two evaluation methods were essential not only in assessing how the final model performed, which we will look into later, but also, early on, in tuning the model to use optimal parameters. During the entire development process, we used the evaluation system to determine

hyperparameters such as the number of neighbors or the weight of the cosine similarity score compared to the Gower distance score, as we will see in section 4.3.4.

4.3.4 Model Tuning

In the following two sections, we will expand on the performance of our suggestion system. In the present section, we will discuss how the optimal parameters were selected for the model. These include the number of nearest neighbors (k) and the balance between the weights for the k -NN and for the NLP parts. Finally, in section 4.3.5, we will present the results from our evaluation functions and interpret them in the context of PLAN.

Firstly, to establish the optimal number of neighbors (k) for our suggestion system, we executed a systematic analysis involving the calculation of our performance metrics - precision, recall, and F1-score - for each value of k within a specified range. Subsequently, we visualized this data through a comprehensive plot showcasing how each of these metrics varied with the number of neighbors. The objective of this approach was to examine the relationship between the number of neighbors in the k -Nearest Neighbors algorithm and the performance of our suggestion system, thereby allowing us to pinpoint the most effective k value for our application.

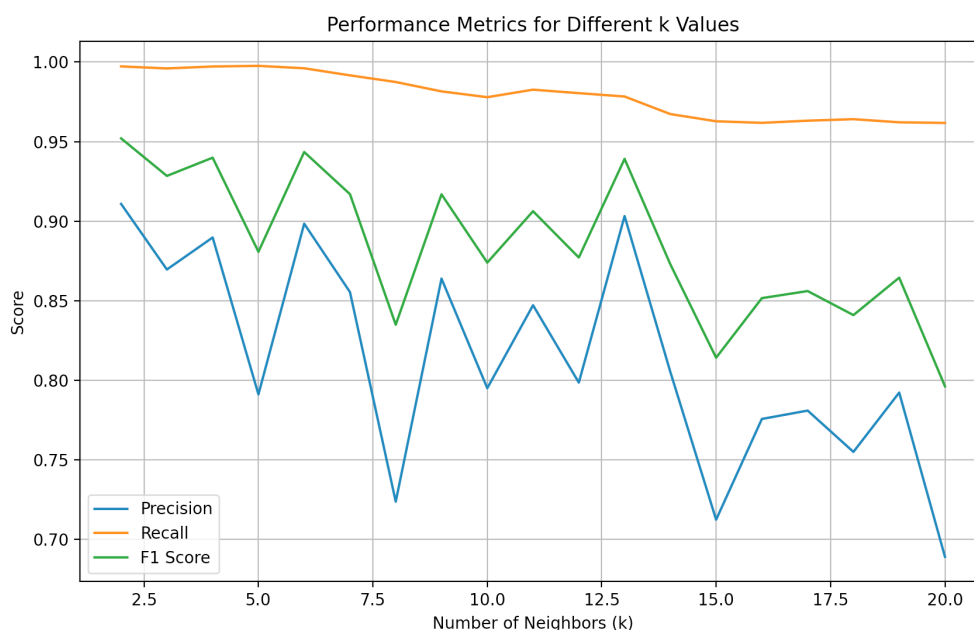


Figure 4.10: Similarity Score by Number of Neighbors

The optimal number of neighbors (k) in our k -Nearest Neighbors algorithm, as shown in Figure 4.10, was determined based on observing the fluctuation of similarity scores across a range of k values. For k ranging from 2 to 13, the similarity scores consistently maintained within a relatively narrow range of 0.75 to 0.9. Beyond $k=14$, however, there was a decline in the similarity score, which continually decreased as k increased. Based on this analysis, we concluded that a k value between 10 and 12 would provide an optimal balance between a high similarity score and

a significant amount of neighbors to later retrieve our suggestions from, enabling our suggestion system to generate useful and relevant suggestions.

In addition to selecting an optimal k , it was necessary to determine the relative weighting of the Gower distance and the cosine similarity in the calculation of the final similarity score. To do so, we used a similar method, where we calculated the Precision, Recall, and F1-score metrics with a varying k -NN weight. The k -NN weight ranged from 0 to 1, while inversely, the similarity score was given by $1 - k$ -NN weight. We can see the results in Figure 4.11.

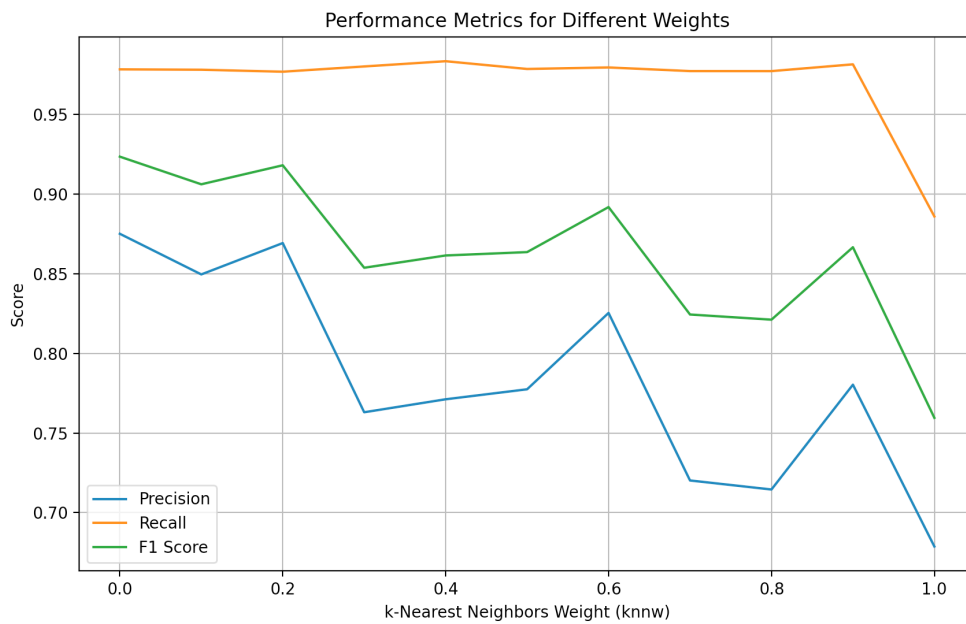


Figure 4.11: Similarity Score by k -NN weight

From our analysis, it becomes apparent that the similarity score decreases as the weight allocated to the k -NN or Gower distance metric increases. A notable decline in the final score is observed when the Gower distance's weight approaches 0.2 or 0.3. This observation suggests that a weight of approximately 0.7 or 0.8 for the cosine similarity score, which is the inverse of the Gower distance weight, would be the most appropriate for our application.

This conclusion corroborates our initial supposition that the 'TITLE' of an activity holds substantial importance in defining it and, subsequently, in identifying activities of a similar nature. The cosine similarity score, which predominantly focuses on this 'TITLE', thus bears significant influence on the final similarity score. As such, the higher weight assigned to the cosine similarity score reflects its crucial role in the accuracy and efficiency of our suggestion system.

4.3.5 Final Model Performance

Once the default parameter values have been set, it is time to actually analyze and interpret the final performance of our suggestion system. The final results obtained with our evaluation system (described in the previous section), for a group of 10 nearest neighbors and with the NLP

cosine similarity score constituting 80% of the final similarity score used are those present in the following table.

Table 4.1: Suggestion System Performance

Feature	Precision	Recall	F1-score
Activity field suggestion	85,6%	99,4%	91,6%
Contribution suggestion	75,5%	90,2%	77,6%

As elaborated in section 2.2.4, the recall metric provides an indication of the proportion of relevant suggestions that our system generates relative to all potential relevant values that could have been retrieved. Given that our system is designed to produce a list of suggestions, it is expected that the vast majority of the ground truth values will be found within this list, which is reflected in the high recall values of 99.4% and 90.2%.

However, within the context of our system, the precision results offer a more meaningful measure of performance. This is because precision takes into account the inaccuracies within the suggestions made by the system. In our application, precision signifies the weighted proportion of correct suggestions made out of all suggestions offered by the system. With a precision rate of 85% for suggesting missing values for activity information and 75% for predicting useful contributions for an activity, we are able to meet the requirements set by CERN and PLAN. This implies that, in a majority of instances, our system will be capable of generating predictions and suggestions that users find beneficial. In situations where our suggestions do not align with the user's requirements, there is no negative impact, as users have the option to disregard these suggestions and manually enter their preferred values.

Furthermore, it's worth noting that it's expected for all metrics to return higher values for the first feature. This is because we're only aiming to predict a single true value in this context. In contrast, when it comes to the contribution suggestion feature, which may involve multiple contributions, precision and recall metrics will naturally be lower. This is due to the increased complexity and challenges involved in accurately suggesting all desired contributions for a given activity.

In conclusion, the rigorously conducted evaluations of our suggestion system's performance indicate the substantial success of the algorithm. The optimal balance of parameters identified through these evaluations - specifically, the number of nearest neighbors (k) and the relative weighting of Gower distance and cosine similarity - allow our model to deliver accuracy and relevance in its suggestions.

Notably, our model's precision score, indicative of its ability to produce beneficial predictions, demonstrates considerable success in meeting PLAN's requisites. This demonstrates our system's ability to provide beneficial predictions and suggestions the majority of the time, potentially improving the user's experience while navigating the PLAN system.

Moreover, we understand that our model's performance can be enhanced further with continued refinement based on additional user feedback and activity data that will become available

when it is deployed in PLAN. This allows for our system to be adaptive and continually improve, reinforcing its relevance and effectiveness.

Lastly, we would like to highlight that the ultimate aim of this suggestion system is not perfection in predictions but to facilitate the activity creation process for users. Thus, even suggestions that don't perfectly match the user's requirements still offer valuable assistance by easing the time spent on activity creation.

In summary, the development and evaluation of this suggestion system, tailored specifically for the PLAN system, have proven successful. We believe it holds the potential to enhance the user experience and efficiency of activity planning within PLAN once the integration and deployment are done.

4.4 Summary

This chapter ultimately described the journey taken and all the challenges and obstacles faced towards the development of the final suggestion system.

Starting off with an exploration of the data journey, encompassing all stages from extraction from CERN databases to the subsequent revelation that the ultimate datasets were confined in size, thereby limiting their efficacy for applications in machine learning.

An exposition of the original strategy, consisting of a hybrid technique integrating clustering and supervised learning, was undertaken in an attempt to fulfill the requirements. However, following the performance evaluation of this strategy, it was promptly understood that the chosen method could not produce satisfactory outcomes due to the scarcity of data.

Given this setback, it became necessary to revise our approach and conceive a totally different strategy, one that would be better suited for scenarios with limited data. This gave rise to the implementation of a k-Nearest Neighbors (k-NN) algorithm in conjunction with several Natural Language Processing (NLP) techniques to establish a weighted similarity measure. Iterations, modifications, and fine-tuning allowed this model to generate precise and accurate recommendations, therefore offering a viable resolution to the problem.

Chapter 5

Practical Implementation and Deployment

This chapter focuses on the practical implementation and deployment of the suggestion system within the PLAN platform at CERN.

Section 5.1 describes the implementation of the API, which acts as a communication channel between the PLAN platform and the Python-based algorithm developed for the suggestion system.

Section 5.2 discusses the integration of the API with the PLAN platform and the design of the user interface. The current state of the integration is described along with the motivations and an example of what the final platform will look like with the suggestion system.

Section 5.3 explores the potential practical impact of the suggestion system. Although still in the proof of concept phase, we detail what benefits the system might bring to the platform.

Section 5.4 addresses maintenance and future improvements for the suggestion system. It emphasizes the importance of keeping datasets up-to-date, monitoring API performance, gathering user feedback, and optimizing system response times.

Finally, section 5.5 summarizes the chapter, leading us into the conclusions chapter.

5.1 API implementation and performance

As described in section 3.1.3, PLAN is a Java-based platform. The algorithm described throughout section 4.3 to tackle our task prediction problem was developed in Python. Consequently, to be able to integrate our algorithm with PLAN, we need to ensure the two entities are able to interact with one another.

To do so, we will use an Application Programming Interface (API). An API is a set of functions and procedures that define how two applications will communicate between each other. They can be used, among other things, to enable software component interaction, allowing them to exchange data and combine capabilities. They are crucial tools for software development because they enable code modularity and reuse while abstracting complexity.

In our case, an API is exactly what we need for algorithm integration. It will allow us to exchange information between PLAN and our algorithm, where PLAN will provide an activity and its current state, and our algorithm will reply with suggestions.

The API, developed using Flask (a Python framework for designing APIs), encapsulates our model and translates the four functionalities described in section 4.3.2 into endpoints. An endpoint is essentially a point where the API establishes a connection between the desired software systems.

The four endpoints are:

- **/get_similar_activities:** PLAN provides an activity at any point during or after its creation, and the API returns a list of the k most similar activities.
- **/get_activity_suggestion:** PLAN provides an activity at any point during or after its creation, and the API returns the most likely suggestions for the missing fields in the given activity.
- **/get_initial_contributions_suggestions:** PLAN provides an activity after its creation, and the API returns the suggested contributions based on the newly created activity.
- **/get_combined_contributions_suggestions:** PLAN provides an activity after its creation alongside the contributions that have already been entered, and the API returns the suggested contributions based on the newly created activity and its contributions.

As we have already gone over how these functionalities function on the algorithms side, we will provide and explain an example of the `/get_similar_activities` endpoint, to contextualize and understand how an endpoint establishes a connection and allows for information exchange between PLAN and our model.



Figure 5.1: API Request Example

In Figure 5.1, we can see an example of an API request. A request is a message sent to a server asking an API to provide a service or information. In our case, requests are sent by PLAN to the API. PLAN uses the URL seen on top of the figure to send the message and can configure the number of neighbors he wishes to receive in return by changing the k parameter.

Each request may need to have information in the body of the request. Here, each request must provide the current state of the activity in its body, in a JSON format, even if there is missing

information. We can see the example of the body of a request at the bottom of Figure 5.1. Note that for the `/get_combined_contributions_suggestions` endpoint, the body of the request will also contain the existing contributions.

Then, once PLAN has sent the configured request message, the API responds with the desired information. We can see the beginning of the response the API would give to the API request example we are analyzing, in the following figure.

```

{
  "ACTIVITY_TYPE_EN": "Safety",
  "ACTIVITY_UUID": "8a1c176e-3c7d-464c-991d-8f40c8f9a26a",
  "CREATION_DATE": "05-APR-22",
  "CREATOR_NAME": "FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)",
  "FACILITY_NAMES": "PS",
  "GOAL": "Reduce the fire load related to the old unused cables and in some cables of cables not respecting the pulling",
  "GROUP_RESPONSIBLE_NAME": "EN-ACE",
  "IMPACT_NOT_DONE": "In case of Fire, all this fire load could make the Fire Brigade intervention impossible an cables\r\nElectro magnetic interference (EMC) due to the mix of cables",
  "LOCATION_INFORMATION": "PS Ring",
  "PRIORITY_EN": "1. Safety",
  "RESPONSIBLE_WITH_DETAILS": "FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)",
  "TITLE": "Injectors De-cabling phase 2 project - PS ring (2 more octants)",
  "WBS_NODE_CODE": "OTHER",
  "similarity_score": 1.0
},
{
  "ACTIVITY_TYPE_EN": "Safety",
  "ACTIVITY_UUID": "501bb340-75b1-4ae9-bfd8-5ec04be49685",
  "CREATION_DATE": "05-APR-22",
  "CREATOR_NAME": "FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)",
  "FACILITY_NAMES": "PS",
  "GOAL": "Reduce the fire load related to the old unused cables and in some cables of cables not respecting the pulling",
  "GROUP_RESPONSIBLE_NAME": "EN-ACE",
  "IMPACT_NOT_DONE": "In case of Fire, all this fire load could make the Fire Brigade intervention impossible an cables\r\nElectro magnetic interference (EMC) due to the mix of cables",
  "LOCATION_INFORMATION": "PS Ring",
  "PRIORITY_EN": "1. Safety",
  "RESPONSIBLE_WITH_DETAILS": "FERNANDO BALTASAR DOS SANTOS PEDR (EN-ACE-OSS)",
  "TITLE": "Injectors De-cabling phase 2 project - PS ring (remaining octants)",
  "WBS_NODE_CODE": "OTHER",
  "similarity_score": 0.941987236173413
},

```

Figure 5.2: API Response Example

As we can see, for this particular endpoint, the API would send a response back to PLAN, that not only acknowledges the reception of the request, but also sends back information on the most similar activities to the one that was provided, in the format of a ranked list.

A new piece of information is added to each activity, which is the similarity score calculated by our model. For example, in Figure 5.2, we can see that the first returned activity has a similarity score of 1.0, which means that it is a 100% match with the provided activity. This will help PLAN make more informed decisions on what to do with the returned activities.

When developing the API, the main focus was to make it have an optimal performance. This is, for the response time to the request to be the lowest possible, for PLAN to be able to make the API requests in real-time, as the user navigates through the platform. We ended up with the

performance from table 5.1. The results were obtained from Postman, a platform for building and using APIs.

Table 5.1: API Performance

Endpoint	Mean response time (ms)
/get_similar_activities	159
/get_activity_suggestions	126
/get_initial_contributions_suggestions	232
/get_combined_contributions_suggestions	1176

As we can see in table 5.1, with the exception of the last endpoint, we have been able to achieve a low response time, that will allow the users to have an almost instant response to the information they enter. The /get_combined_contributions_suggestions endpoint has a slower response time because the algorithm is currently computing all the association rules for each request. This will be further discussed in section 5.4.

The API's usability was also a critical aspect of its integration with the PLAN platform. A comprehensive, well-structured documentation was provided to guide PLAN developers through the process, detailing API endpoints, request and response formats, and error handling procedures. In addition, unit testing was conducted to ensure the robustness and reliability of the API. These measures aimed to offer a seamless developer experience, facilitating efficient integration and contributing to the overall success of the deployment

In the next section, we will thus describe how the API was integrated and deployed into the PLAN platform.

5.2 Integration and Interface Design

Once the Application Programming Interface was fully developed and implemented, the next step in the working process was to integrate it with the task planning platform, PLAN.

The first step in the integration is to go through a proof of concept phase (POC). Proof of concept is evidence that usually comes from an experiment, which demonstrates that a concept is feasible. In this case, the POC wants to prove that the suggestion system developed is good enough and able to seamlessly be featured in the platform.

In this phase, we used Docker to allow both our algorithm and PLAN to be locally containerized. By doing so, they are able to communicate with each other on a local machine, allowing PLAN developers to design and define how the suggestion system will fit into the interface.

From our side, we were charged with creating a docker image and container for the API and making all code and documentation related understandable and easy to use and modify. From there, deployment was made, and PLAN developers would use and experiment with the suggestion system as they wished.

Currently, due to other PLAN features' priority and PLAN sprints' time constraints, the integration and deployment are still in the proof of concept phase. The first feature of the suggestion

system has already been experimented with, and the interface design and interaction with the user are being refined.

When a user is creating an activity, he will be presented with the following modified interface from what we have seen in Figure 4.5.

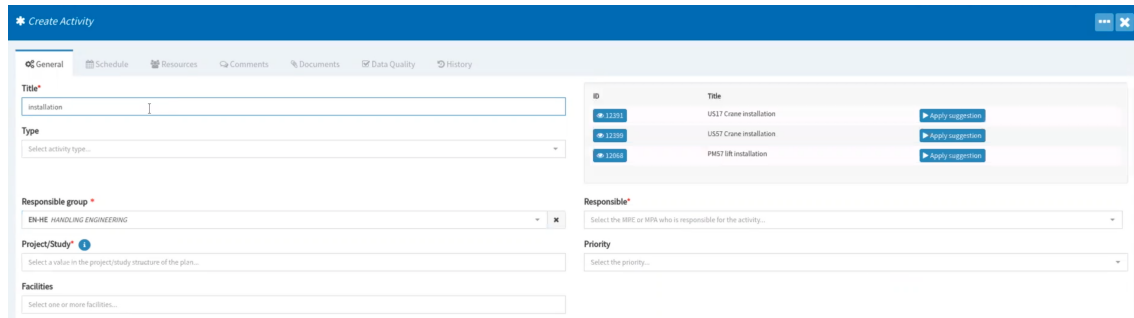


Figure 5.3: PLAN Interface with Suggestions

PLAN now displays the 3 most similar activities to the one being created. They are updated in real-time as the user enters information. For example, in Figure 5.3, the suggestion system computes the nearest neighbors with information on the responsible group and with the "installation" word entered in the title. From the returned activities, the user now has the option to apply one of the activities or to check the full information of the activities before doing so. If he clicks on the view button from the last presented feature, the following menu will appear.



Figure 5.4: PLAN Activity Suggested Interface

As we can see, the title of the activity has the word "installation" in it, and the responsible group matches the one entered by the user. From here, if the user deems the activity to be useful and similar to what he wants to create, he can apply the suggestion, and PLAN will automatically clone and fill out some of the activity fields.

The design of the interface was carried out with a strong focus on user intuitiveness and ease of use. The goal was to make the suggestion system a seamless part of the PLAN platform that adds value without adding complexity. To achieve this, PLAN developers worked on minimizing the number of steps and decisions the user has to make while maximizing the utility and relevance of the suggested activities. The result is an interface that presents clear options and actions to the user, with suggestions dynamically updating as the user inputs data into the system.

This thoughtful integration and user-centric design of the suggestion system sets the stage for assessing its potential practical impact. The ultimate measure of the system's effectiveness lies in how it performs in real-world conditions and the response it garners from the users of the

PLAN platform. The following section dives into these aspects, providing insight into the tangible benefits and improvements that could be brought by the system.

5.3 Practical Impact

The suggestion system's primary intent is to bring a significant practical impact to CERN's PLAN platform, transforming the way activities are created and managed. Although currently in the proof of concept phase, we can project the potential advantages and efficiencies this system is expected to introduce.

One of the key benefits of the suggestion system is anticipated to be a considerable reduction in the time spent on creating activities. With the assistance of the system, users can find similar tasks and clone them, thus eliminating the need to create new ones from scratch. This not only saves time but also ensures consistency in the way activities are structured and documented. The same can be said for contribution requests.

Additionally, the system is envisioned to provide valuable decision-making support, particularly when users are uncertain about the specific contributions required for an activity. By presenting a list of similar past activities, the system can provide users with an idea of what contributions might be relevant or necessary, thereby facilitating informed decision-making.

Lastly, the interactive nature of the suggestion system can potentially stimulate user engagement and interest. It is common that user-friendly, responsive interfaces, particularly ones that proactively offer assistance, tend to attract and retain users. As users interact with the system and discover its utility, their experience with the PLAN platform could be enriched. This engagement can also encourage more frequent and effective usage of the platform, consequently leading to more robust and comprehensive task planning.

While these impacts are currently projected, the potential of the system can only be fully realized once it is deployed beyond the proof of concept phase. User feedback, combined with further enhancements and improvements (section 5.4), will help shape the suggestion system into a robust, user-friendly tool integral to the PLAN platform.

5.4 Maintenance and Future Improvements

The present state of our suggestion system represents a significant milestone, yet it is by no means the final product. As the system is deployed, continuous maintenance, monitoring, and improvements will be essential to keep it relevant and effective in serving the needs of the PLAN platform users. This section will discuss some of the key areas of focus for maintaining the system's efficiency and effectiveness, as well as potential future enhancements.

First of all, keeping the activities and contributions datasets up-to-date is crucial for the ongoing compatibility and relevance of the recommendation system with PLAN. One possible way to ensure this is to establish a periodic endpoint that updates the tables. Python scripts that serve this purpose have already been developed, only missing a continuous connection to PLAN databases.

This could be programmed to occur during periods of low user activity, such as overnight, thereby minimizing the impact on system performance.

Monitoring the API performance is another important area for ongoing maintenance. Regular performance checks can help identify and rectify any emerging issues promptly, thereby ensuring that the system continues to operate smoothly and efficiently. This includes monitoring the system's response times to API requests, particularly during peak usage times.

User feedback is an invaluable resource for improving the system. Addressing feedback, both positive and negative, can provide vital insights into potential areas for enhancement. These could range from tweaks to the user interface, making it even more intuitive and user-friendly, to more substantial changes to the underlying algorithms and methods that drive the recommendation system. Not only subjective feedback from users could be used through forms or simple interviews, but metrics such as the number of times a user uses the activities suggested, or the percentage of accepted contributions compared to the percentage before the suggestion system was implemented, could be useful for the analysis and refinement of the system. A constant focus on user experience is key to maintaining user satisfaction and engagement.

Finally, regarding the association rules calculation currently performed for each request, it would be wise to make this operation periodic. By storing the computed rules for reuse instead of computing them for each request, the system could significantly reduce the response time for the `/get_combined_contributions_suggestions` endpoint. This would, similarly, improve overall system performance and enhance the user experience.

In summary, while the suggestion system is already providing value, there are numerous opportunities for continuous improvement and optimization once the integration is complete and user feedback starts to exist. By focusing on these areas of maintenance and future development, the system can continue to evolve and provide even greater benefits to the PLAN platform and its users.

5.5 Summary

This chapter detailed the implementation and integration of the task prediction algorithm into the PLAN platform via an API. This API was designed with Flask, to bridge the gap between the Java-based PLAN platform and the Python-based prediction algorithm, offering four endpoints to serve specific functionalities.

The performance of the API was optimized to ensure real-time interactions. Although one endpoint has a slightly slower response time, overall performance met the requirements. The integration process is still in the proof of concept phase due to other priorities within PLAN.

The anticipated practical impact of the suggestion system includes reduced time spent on activity creation, enhanced decision-making support, and increased user engagement. To maintain its effectiveness, continuous system maintenance, and future improvements are necessary, which include data updates, performance monitoring, user feedback, and algorithmic enhancements.

Overall, the chapter illustrates how the suggestion system has begun to offer value to the PLAN platform and its users, and it highlights numerous opportunities for further improvements and optimization.

Chapter 6

Conclusions

This chapter serves as the conclusion of this dissertation, providing a concise summary and review of the project's key findings and methodologies in section 6.1. It highlights the main accomplishments and outcomes achieved throughout the research, offering a comprehensive overview of the work conducted.

Section 6.2 discusses the limitations and challenges faced during the project, which may have imposed certain constraints on its advancement. It also presents the potential for future research and development.

In section 6.3 the implications and contributions of the research are discussed, emphasizing its relevance and applicability to both CERN and the broader scientific community.

Finally, section 6.4 concludes the dissertation by reflecting on the journey undertaken to reach the final solution. It provides a thoughtful analysis of the lessons learned, challenges encountered, and achievements made throughout the research process.

6.1 Summary of Findings

The goal of the research journey undertaken for this thesis was to create a suggestion system that would improve the usability and usefulness of PLAN, the task planning platform used by CERN. The key goal of this project was to design an algorithm that can successfully offer activity and contribution suggestions despite the limitations of the available data (chapters 1 and 3, section 4.1).

The initial approach was a hybrid model that leveraged the strengths of clustering techniques and supervised learning. The motivation behind this methodology was straightforward. We anticipated that the clustering methods would be able to detect the underlying patterns and structures in the historical data on activities, grouping them into distinct categories. Subsequently, the supervised learning algorithm would predict the group or cluster to which a new activity would belong, thereby enabling us to generate suggestions based on the common characteristics of activities within the identified cluster (section 4.2).

However, this initial approach encountered significant challenges. The performance of the clustering model was subpar, with a high Davies-Bouldin Index score and a low silhouette score indicating poor quality of clustering. This less-than-satisfactory outcome was primarily attributable to the sparse nature of the available data, demonstrating that the chosen approach was less effective when dealing with data sets of low volume (sections 4.2.4 and 4.2.5).

Due to this situation, we had to review our methodology and think of some different approaches that would be more appropriate given the characteristics of our data. Our new strategy brought us to the combination of the instance-based learning algorithm k-Nearest Neighbors (k-NN) and Natural Language Processing (NLP) methods. When dealing with sparse data, this method proved to be noticeably more adaptable and effective (section 4.3).

The instance-based learning, inherent to the k-NN algorithm, leveraged the available instances as references, effectively identifying activities that were similar to a new one, even within the constraints of a small dataset. Additionally, the integration of NLP techniques enhanced the similarity measurement by extracting semantic similarity from activity titles, thus improving the accuracy of our suggestions (section 4.3).

The end result was a practical and data-efficient API that could generate helpful suggestions within the PLAN system, effectively meeting CERN's requirements. The developed algorithm demonstrated a good precision measure for the suggestions, signifying the success of the chosen approach (section 4.3.5). However, it is important to acknowledge that every solution has its limitations, and ours was no exception. The nature of these limitations, as well as the potential ways to overcome them, will be discussed in section 6.2.

As the suggestion system is still in its proof-of-concept phase, comprehensive user feedback and experiences have yet to be collected (section 5.2). Nonetheless, the preliminary feedback received from CERN's supervisor, Fernando Pedrosa, and from Rodrigo Lanza, a PLAN developer, has been very positive, which is promising for future user experience.

In summary, this thesis has documented a successful journey in the development of a suggestion system for PLAN, where significant advances were made from the initial algorithmic approach to the final solution. The key challenge of sparse data was addressed through a strategic change in methodology, and the resulting solution met the required criteria. As we continue to improve and refine this system, we look forward to unlocking further potential and maximizing its impact on PLAN's functionality and user experience.

6.2 Limitations and Future Work

The work presented in this thesis has successfully navigated through various challenges to develop a suggestion system for PLAN. However, it's essential to acknowledge certain limitations that were encountered during this journey, as well as identify the potential for future work to enhance the effectiveness of the developed system.

One of the key limitations since the beginning was the scarcity of data, which restricted the types of algorithms that could be effectively employed and influenced the performance of the final

solution. In machine learning, the volume and quality of data play a crucial role in the performance and accuracy of the models (2.5). The initial approach using clustering and supervised learning failed to deliver satisfactory results due to the small dataset size. Although the final solution using k-NN and NLP managed to perform well with the limited data, it's important to acknowledge that the performance of the system could potentially be improved with larger data volumes.

The quality of the data, particularly the activity titles, also posed a limitation. The efficiency of the NLP part of the approach heavily depends on the descriptiveness and consistency of these titles. If the titles are not detailed or descriptive enough, this could lead to inaccuracies in the suggestions provided by the system. The data quality issue might suggest a need for guidelines or standards in activity title creation to improve the performance of the suggestion system.

Another limitation that modified the project timeline and progress was the dependence on the PLAN development team for the integration. The integration process being dependent on a third party, which I couldn't control, led to a delay in moving beyond the proof-of-concept phase, also leading to not having any user experience or feedback (5.2).

Moving forward, there are several paths for potential future work to improve the performance and utility of the suggestion system. One major area of focus would be refining the NLP techniques employed (4.3.1). Although TF-IDF was used to extract semantic similarity from activity titles, there's room for improvement. More advanced NLP techniques such as synonym detection, disambiguation, or even the utilization of sophisticated models like BERT or Word2Vec could enhance the semantic understanding, thereby improving the quality of similarity measures and the precision of the suggestions.

The suggestion system could also be improved by addressing the potential latency issue associated with the k-NN algorithm. As the dataset grows in size, the computation time for the k-NN algorithm could increase, leading to slower response times. Future work could investigate strategies to optimize the k-NN computation, such as using indexing methods and approximate nearest neighbor techniques, or even to consider other approaches.

Finally, keeping the datasets updated with the latest activities and contributions in PLAN is crucial for the system to remain relevant and accurate. Establishing a periodic endpoint to update the system's tables could ensure compatibility with PLAN. Furthermore, implementing a mechanism for continuous monitoring of API performance, addressing user feedback once we have it, and refining user experience or algorithms based on the feedback could lead to considerable improvements in the system's performance and usability (5.4).

In conclusion, while the development of the suggestion system for PLAN has been a successful endeavor, there are limitations to acknowledge and plenty of opportunities for future work. By understanding these limitations and actively working towards addressing them, we can strive to improve the suggestion system's performance, utility, and user experience in the future.

6.3 Implications and Contributions

The work undertaken in this thesis provides several implications and contributions to both the field of artificial intelligence applications in task management and, more specifically, to CERN's PLAN platform.

Through the development of the suggestion system, this project has contributed to addressing our first research question: "How can we use machine learning algorithms to improve platform interaction and to provide assistance to the user?". The successfully implemented system that combines k-Nearest Neighbors and Natural Language Processing techniques has proven that machine learning can indeed enhance user interaction on a task planning platform such as PLAN. By offering data-driven suggestions during the process of activity creation, the system enables a more seamless and efficient interaction for the users, thereby improving their user experience (1.4).

This project has also made a significant contribution to answering our second research question: "Can machine learning be used to improve task management and resource planning in PLAN, considering the low quantity of historical data available?" Despite the challenging constraints the limited historical data presented, this project has demonstrated the feasibility of applying AI to improve task management and resource planning in PLAN with the promising precision results of the final model, that showed an 85% success rate in predicting activity information and 75% for contribution requests suggestion (1.4).

In regards to our third and final research question: "Which machine learning techniques are most appropriate and efficient in low-data situations", having ran through different approaches and techniques during the development process, the strategic approach of combining k-NN and NLP methods has proven that even with sparse data, it is possible to achieve satisfactory accuracy and precision when making suggestions (1.4).

While this research focused primarily on the development of a suggestion system for CERN's PLAN platform, the underlying techniques utilized, specifically the combination of k-Nearest Neighbors and Natural Language Processing, can have broader applications. For instance, these methods can be used to improve autocomplete features in various business applications. As for our first use case, form field completion is a common task in such applications, and the ability to accurately predict and suggest likely entries can greatly enhance user experience. This approach could work for any form-based business application.

The second area of broader application is related to our second use case and is in the context of recommending items to add to lists. A good example of this is the creation of a purchase order, which typically involves selecting from a list of articles. By applying the same k-NN and NLP techniques, we can analyze the historical data of previous orders, identifying patterns and associations between items and contextual information. With this information, we could suggest relevant items to users based on their current selections, reducing the time required to complete the order and potentially upselling additional products that may be relevant to the user.

Beyond the successful development of the suggestion system, another vital contribution of this work is the foundation it has laid for future AI-based improvements in the PLAN system or

other platforms at CERN. The methods developed, and lessons learned from this project form an invaluable basis for subsequent projects that aim to use machine learning in low-data contexts, which was one of CERN's main goals. The knowledge acquired from this journey serves as a robust starting point for future development in this area, accelerating the pace of AI usage in such environments.

The project's success highlights how AI may improve platform usability and task management even with a small amount of data, opening up new opportunities for AI applications across a variety of industries.

This project has not only been about improving a specific platform but also about demonstrating that with the right approach, even seemingly insurmountable challenges, such as limited data, can be overcome. It is evidence that artificial intelligence (AI) and machine learning are strong technologies that can be used in a variety of contexts to enhance user experience and operational effectiveness.

6.4 Final Thoughts and Conclusions

As this thesis work comes to an end, this section will conclude and reflect on the advances made and the lessons learned.

At the beginning of this project, the aim was to explore the usage of machine learning in task management, taking into account the unique challenge of a limited quantity of historical data within the PLAN system at CERN. Throughout this journey, from the initial approach to the final execution, the objective was consistently met with innovative problem-solving, the drive for continuous learning, and a deep dive into machine learning algorithms that could aptly adapt to the constraints of low data volume.

In successfully developing a functional suggestion system that not only enhances the PLAN platform but also sets the groundwork for future AI implementations in low-data environments, this thesis has made significant contributions to the field and to the operations at CERN. However, the research went beyond its technical aspects, providing an enriching personal experience that enhanced problem-solving skills and broadened knowledge.

Working on this project as an intern at CERN was also a significant part of this learning journey. The experience of working in a diverse and dynamic environment like CERN provided invaluable insights into collaboration, cross-disciplinary communication, and real-world problem-solving.

Beyond the technical skills and knowledge, this journey has also reinforced life skills such as perseverance, adaptability, and collaboration. Encountering and overcoming problems, adapting to changing circumstances, and working closely with a diverse team of professionals were all integral parts of this experience.

As this project comes to an end, the learnings, experiences, and accomplishments serve as stepping stones for future work in the field of machine learning. It is with immense gratitude and

a sense of fulfillment that I conclude this thesis, eagerly looking forward to the future challenges and opportunities that await in the world of data science.

In conclusion, this project demonstrated that, indeed, machine learning could make significant contributions to task management and resource planning, even in low-data environments, given the right approach and techniques. Through continuous learning, we successfully developed a suggestion system for the PLAN platform at CERN, thereby answering our initial research questions positively and paving the way for future work in this area.

References

- [1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules, 1994.
- [2] David W Aha, Dennis Kibler, Marc K Albert, and J R Quinlan. Instance-based learning algorithms. 6:37–66, 1991.
- [3] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation, 8 2020.
- [4] Asana. Why asana? <https://asana.com/guide/resources/info-sheets/why-asana>, 2023.
- [5] Baeldung. Silhouette plots. <https://www.baeldung.com/cs/silhouette-value-s-clustering>, 2023.
- [6] CERN. Cern - about. <https://home.cern/about/who-we-are/our-people>, 2023.
- [7] CERN. Cern lhc. <https://home.cern/science/accelerators/large-hadron-collider>, 2023.
- [8] Muhammad Umar Cheema and Qudsia Zearlish. The choice of project management software by project managers; with the moderating impact of top management support, 2022.
- [9] K. R. Chowdhary. Natural language processing. *Fundamentals of Artificial Intelligence*, pages 603–649, 2020.
- [10] David Davies and Don Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979.
- [11] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.
- [12] Wael Gomaa. A survey of text similarity approaches. *International Journal of Computer Applications*, 68:13–18, 4 2013.
- [13] Sudhanshu Gupta. K-nearest neighbor(knn) algorithm for machine learning. https://medium.com/@sudhanshugupta_66164/k-nearest-neighbor-knn-algorithm-for-machine-learning-1b506eb2c4a4, 8 2021.
- [14] Wen Ming Han. Discriminating risky software project using neural networks. *Computer Standards and Interfaces*, 40:15–22, 6 2015.

- [15] Yong Hu, Jiaying Huang, Juhua Chen, Mei Liu, and Kang Xie. Software project risk management modeling with neural network and support vector machine approaches, 2007.
- [16] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 12:283–304, 1998.
- [17] Jira. What is jira used for? <https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for>, 2023.
- [18] Tushar Joshi. Evaluating clustering algorithm — silhouette score. <https://tushar-joshi-89.medium.com/silhouette-score-a9f7d8d78f29>, 2021.
- [19] Fatih Karabiber. Cosine similarity. <https://www.learn-datasci.com/glossary/cosine-similarity/>.
- [20] Harold Kerzner. *Project management: A systems approach to planning, scheduling, and controlling*. John Wiley Sons, 2009.
- [21] S B Kotsiantis. Supervised machine learning: A review of classification techniques, 2007.
- [22] Sotiris Kotsiantis, Dimitris Kanellopoulos, and P E Pintelas. Data preprocessing for supervised learning, 2014.
- [23] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. Cosine similarity to determine similarity measure: Study case in online essay assessment.
- [24] Christopher D. Manning, Prabhakar. Raghavan, and Hinrich. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.
- [25] Aditya Mishra. Metrics to evaluate your machine learning algorithm. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>, 2018.
- [26] Kevin P Murphy. Machine learning a probabilistic perspective, 2012.
- [27] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341, 2007.
- [28] Microsoft Project. Project management software | microsoft project. <https://www.microsoft.com/en-us/microsoft-365/project/project-management-software>, 2023.
- [29] Redmine. Redmine - wiki. <https://www.redmine.org>, 2023.
- [30] Natasha Sharma. Importance of distance metrics in machine learning modelling. <https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d>, 2019.
- [31] Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of Big Data*, 8, 12 2021.
- [32] Ana Sá and Sousa Carneiro Da Silva. Suggesting human resources for project tasks, 2021.
- [33] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. IEEE, 2019.

- [34] Trello. Trello makes it easier for teams to manage projects and tasks. <https://trello.com/tour>, 2023.
- [35] Hastie Trevor, Tibshirani Robert, and Friedman Jerome. *Overview of Supervised Learning*, pages 9–41. Springer New York, 2009.
- [36] Eugenio Zuccarelli. Performance metrics in machine learning — part 3: Clustering. <https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6>, 2021.