UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Research and Development

BACHELOR THESIS

Ngô Thiên Nga – BI12-310

Information and Communication Technology (ICT)

Title:

"Machine Learning-based Toolkit Recommendation Platform"

External supervisor: Trần Đức Nam – FPT Software

Internal supervisor: Trần Giang Sơn – USTH ICT Laboratory

Hanoi, 2024

DECLARATION

I hereby, Ngo Thien Nga, declare that my thesis represents my own work after doing an internship at FPT Software.

I have read the USTH University's internship guideline. In the case of plagiarism appearing in my thesis, I accept responsibility for the conduct of the procedures in accordance with the University's Committee.

Hanoi, September 2024

Signature

Ngo Thien Nga

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I am deeply grateful for the opportunity to intern at FPT Software, an experience that has been truly invaluable. I would like to extend my heartfelt thanks to my colleagues who supported me throughout my internship. In particular, I would like to express my sincere appreciation to PM Trần Đức Nam and Mr. Phạm Tấn Đăng for their guidance and mentorship during the completion of my graduation thesis. It has been a remarkable journey, and I will cherish these memories for a long time.

Next, I would like to thank USTH and the ICT department. The school and the department have provided an exceptional environment where I have been able to learn, grow, and reach this important milestone. My heartfelt thanks go to the professors in the ICT department for equipping me with the knowledge and skills that will serve as a solid foundation as I step forward into the future. I am especially grateful to Mr. Trần Giang Sơn, who generously accepted the role of my thesis advisor and supported me throughout the internship and thesis process.

I would also like to express my deepest gratitude to my parents. They have worked tirelessly to ensure that I could pursue my education at an institution like USTH, and their unwavering support throughout my over 20 years of schooling has meant the world to me. Though I may be shy in saying it aloud, I want them to know that I love them very much.

Finally, I would like to thank my friends and peers at USTH. From classmates in other departments to senior and junior students, each of you has played a significant role in shaping the person I am today. The support and encouragement you've given me during challenging moments have left a lasting impact.

To the reader of this acknowledgment, I offer my sincere thanks for taking the time to read through these words. This acknowledgment is my way of honouring the teachers, family,

and friends who have supported me on this journey. And to you, the reader, thank you for your time.

Wishing you all the best!

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequency
KNN	K-Nearest Neighbors
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
F1	F1 Score (a measure of model accuracy)
MICE	Multivariate Imputation by Chained Equations
NN	Neural Network
CSV	Comma Separated Values (file format)
ICT	Information and Communication Technology
USTH	University of Science and Technology of Hanoi
PM	Project Manager
FPT	FPT Software (company)
AI	Artificial Intelligence
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
LightGBM	Light Gradient Boosting Machine
ETL	Extract, Transform, Load

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In today's rapidly evolving technological landscape, the efficient management and recommendation of development toolkits have become critical for technology companies. The current system at our company utilizes an outdated website that allows users to search for toolkits by name or filter based on specific criteria such as domain or framework. However, this system lacks the capability to rank or recommend toolkits according to their suitability for new projects, resulting in inefficiencies and time-consuming selection processes.

This research proposes the development of an intelligent recommendation platform that leverages machine learning and natural language processing (NLP) to optimize toolkit selection for new projects. By utilizing historical data from previous projects, the system aims to predict and suggest the most relevant toolkits. Various machine learning techniques, including LightGBM, K-Nearest Neighbors (KNN), and TF-IDF, along with label encoding, are employed to enhance the prediction process.

The project evaluates the accuracy and efficiency of these models using performance metrics such as F1-Score and Root Mean Square Error (RMSE). The proposed platform is designed to provide a more sophisticated approach to toolkit management, improving user experience and project development efficiency.

The outcome of this research not only addresses the limitations of the current system but also demonstrates the potential of predictive models in enhancing decision-making processes within the company.

Keywords: Toolkit Recommendation, Machine Learning, Natural Language Processing (NLP), LightGBM, K-Nearest Neighbors (KNN), TF-IDF, Label Encoding, Project Management, Predictive Models, F1-Score, Root Mean Square Error (RMSE), Decision-Making Optimization.

I/ INTRODUCTION

1. Context

In the rapidly advancing technological landscape, the efficient management and utilization of toolkits for project development have become crucial for technology companies. Currently, the company's system relies on an outdated website where users can search for toolkits by name or filter them by domain or framework. While this system provides basic search and filtering capabilities, it lacks the functionality to rank or recommend toolkits based on their relevance to new project requirements, leading to user inconvenience in selecting the most suitable tools.

2. Motivation

With the evolution of data processing technologies and artificial intelligence, there is an urgent need to create a more intelligent platform to improve toolkit recommendations. The current system's reliance on simple keyword searches and filters often results in inaccurate suggestions that fail to meet the complex needs of modern projects.

The company's leadership seeks to build a system that leverages historical project data, enabling more accurate predictions of appropriate toolkits. By incorporating advanced technologies such as Machine Learning and Natural Language Processing (NLP), this new platform will analyze and process data more effectively. Predictive models like LightGBM, TF-IDF, and K-Nearest Neighbors (KNN) will be used to enhance the accuracy and relevance of toolkit recommendations, ensuring that project managers receive better-suited suggestions for their specific requirements.

3. Objective

The primary objective of this internship as a Data Analyst Intern is to develop an advanced platform that employs machine learning algorithms and NLP techniques to recommend

optimal toolkits based on data from previous projects. This involves researching and applying various algorithms, including LightGBM, KNN, and TF-IDF, along with label encoding techniques, to enhance prediction accuracy. The success of these models will be evaluated using metrics such as F1-Score and RMSE to ensure precise toolkit recommendations for new projects.

4. Proposed Technology

To address the challenges identified, this project will utilize advanced technologies such as LightGBM for data processing and machine learning, and NLP techniques to handle textual data. LightGBM, known for its fast computation and ability to handle large datasets, will be used for model training, while TF-IDF and label encoding will facilitate data preprocessing. Additionally, KNN will be employed to evaluate the similarity between new and existing projects, ensuring accurate toolkit recommendations. The effectiveness of these approaches will be assessed through various performance metrics to ensure high-quality predictions.

5. Thesis Structure

The thesis is separated into 4 main chapters:

- Chapter I: Introduction.
- Chapter II: Material and Methodology.
- Chapter III: Result and Discussion.
- Chapter IV: Conclusion.

II/ MATERIALS AND METHODS

1. System Architecture

The system architecture is designed with a modular approach, integrating different components to handle the entire lifecycle of data processing, model prediction, and toolkit recommendation. Each component of the architecture serves a distinct role, ensuring that the system operates efficiently and can scale as needed. Here's a breakdown of each part:

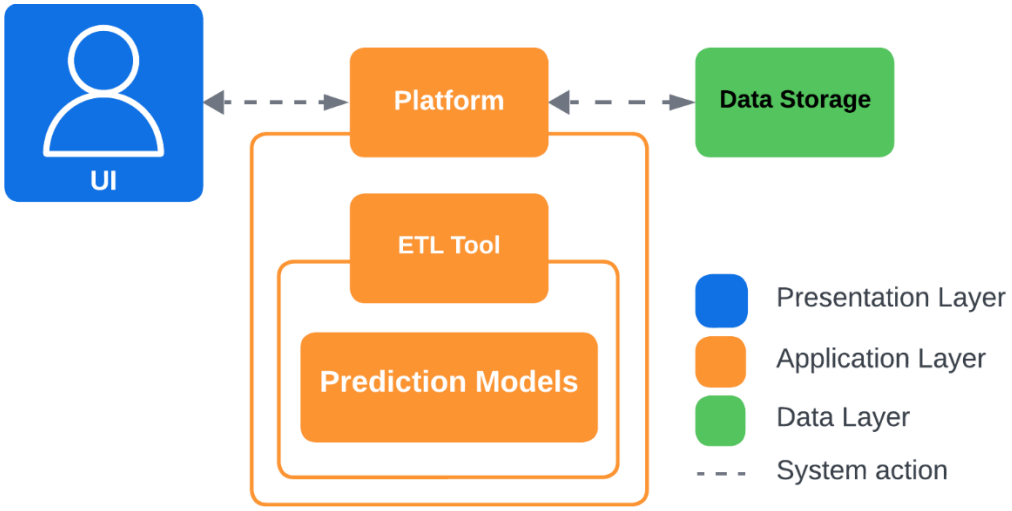

Figure 1: System Architecture

- Users interact with the Frontend to submit project details or search for suitable toolkits.
- The Backend processes these inputs, storing them in the Data Storage (PostgreSQL) and triggering the ETL process (Apache Airflow).
- Apache Airflow ensures the data is cleaned and transformed, and it passes the data to the Prediction Models (LightGBM, KNN, Cosine Similarity) for analysis.
- The system then returns predictions and recommendations back to the Frontend for user interaction.

This architecture balances user interactivity, automated data processing, and intelligent recommendation systems, ensuring both efficiency and scalability.

2. Materials

The Materials section outlines the fundamental resources and data sources that underpin the development and implementation of this system. Following the System Architecture section, which introduced the overall structural design, this section focuses on the types of data used, including their origins and specific characteristics that make them suitable for achieving accurate toolkit recommendations.

2.1. Data Sources

This subsection outlines the key data sources used to train and validate the toolkit recommendation system, including project histories and toolkit attributes. Each source is briefly described, covering its role, format, and preprocessing requirements.

- **Toolkit Data:**

The data about toolkits is currently stored in the company's old web system in CSV file format. This dataset includes various toolkits, some of which are outdated and no longer in use. A significant issue is the presence of many missing values and errors in the data, which reduces the quality of the dataset. Therefore, data preprocessing is the first step before using it in prediction models.

- **Old Project Data:**

The data on old projects is collected from two sources: previous projects and new forms updated by team members. These datasets need to be combined to provide a comprehensive and accurate view of the toolkit usage history. However, a major challenge is ensuring the accuracy and completeness of information from new forms, especially when assessing the effectiveness of toolkits.

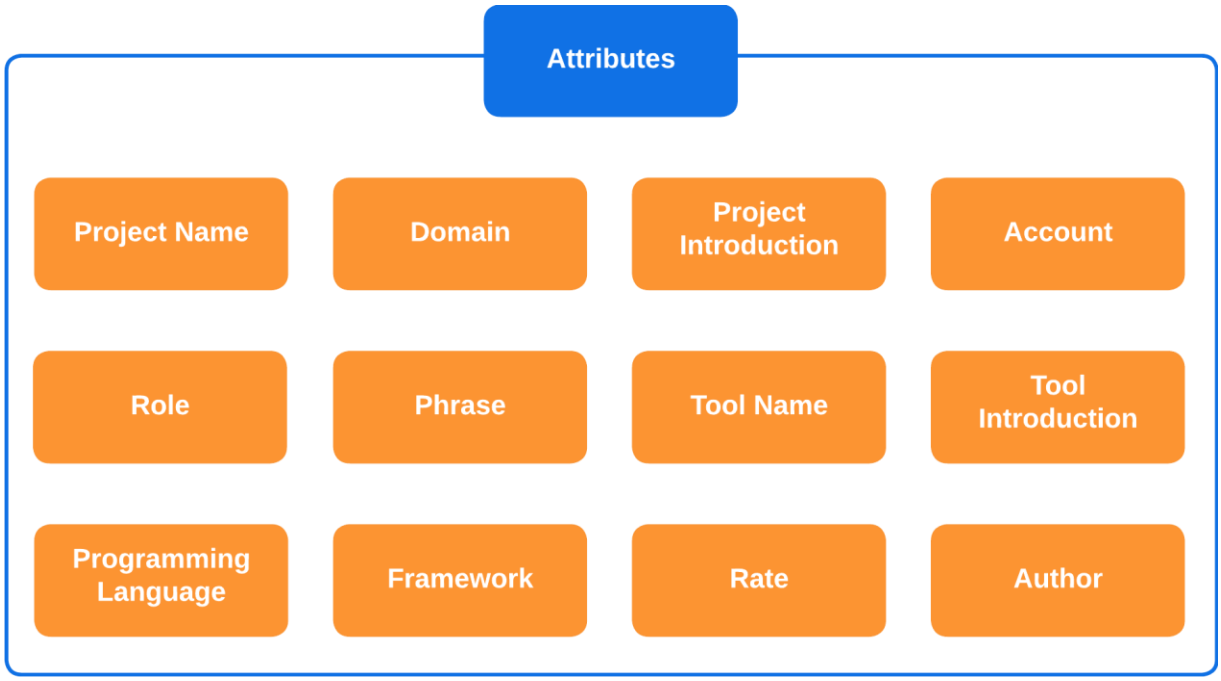

Figure 2 Database Design

After integrating the two primary data sources, the database was structured to support efficient querying and accurate recommendations. This design includes approximately 12 attributes, each selected to capture key project and toolkit details, such as project domain, toolkit name, programming language, and relevant features. The combined dataset comprises roughly 2,000 rows, offering a robust foundation for model training and evaluation. Each attribute underwent initial inspection to ensure compatibility and consistency across both sources.

With the database structure established, the next step involves Data Processing to prepare the data for modeling. This includes cleaning, encoding, and handling missing values to ensure that all attributes are standardized for input into the recommendation model.

2.2. Data Processing

The Data Processing phase prepares the combined dataset for modeling by applying essential transformations. This stage involves cleaning the data, encoding categorical values, and addressing missing entries to ensure consistency and reliability. Each step

optimizes the data, enabling the recommendation model to achieve high accuracy in predicting relevant toolkits.

The data pipeline has been designed to handle large volumes of project and toolkit data efficiently, ensuring that data is processed systematically to generate accurate recommendations. The flow of the pipeline can be summarized in the following stages:

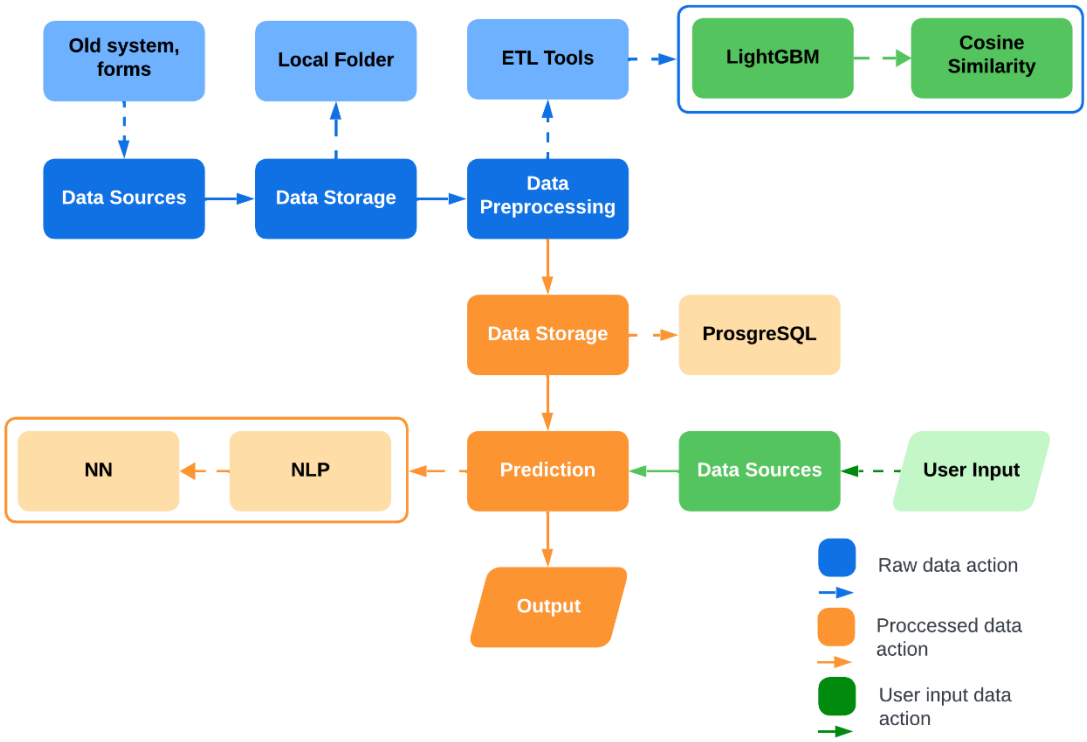

Figure 3: Data Pipeline

The updated data flow includes two main paths:

- Data Sources → Data Storage → Data Preprocessing → Prediction Models → Output.

  The raw data flow begins with sourcing data from the existing system, capturing historical project and toolkit information. This data is then saved locally as a staging step before preprocessing. The blue line represents this raw data action, moving from storage to preprocessing through an ETL tool. Within preprocessing, two main tasks are handled:

- **LightGBM:** This model addresses missing values by predicting and filling gaps in the data.
- **Cosine Similarity:** This step detects and flags duplicate entries to maintain data integrity.

After these preprocessing steps, the cleaned and structured data is considered processed data. Represented by the orange line, this processed data is then stored in PostgreSQL for subsequent analysis. From here, it moves to the Prediction stage, where several key actions take place:

- **NLP:** Natural Language Processing methods are applied to remove stopwords, refining the data for the next steps.
- **Neural Network (NN):** A neural network model then processes the data to predict suitable toolkits based on project requirements.

The output generated from this flow is a ranked list of recommended toolkits.

- Data Sources → Prediction Models → Output.

The second data flow begins with direct user input. This inputs bypasses the raw data staging and preprocessing steps, moving directly into the Prediction stage. Here, the same NLP and neural network methods are applied to match the user's project requirements with the most relevant toolkit recommendations.

This second direct path enables the system to handle data that is already formatted or provided in real-time (e.g., user input) without requiring preprocessing, streamlining the recommendation process.

This dual-path approach enhances the flexibility of the system, allowing it to handle both complex legacy data and simpler, real-time user inputs efficiently. By supporting both paths, the system ensures timely and accurate recommendations regardless of the data's origin or complexity.

With data processing complete, the dataset is now clean, structured, and ready for model input. This preparation stage ensures that the data is consistent and optimized for

accurate toolkit recommendations. In the next section, Methodology, we will outline the specific machine learning techniques and algorithms used to develop and evaluate the recommendation model, providing an in-depth view of the approach taken to achieve reliable predictions.

3. Methodology

This section outlines the methods used to build an accurate toolkit recommendation model, starting with Data Processing Methods to prepare the data for training and prediction. Each step ensures data quality, setting a strong foundation for model development.

3.1. Data Processing Method

Effective data processing is critical for ensuring the accuracy and performance of predictive models, particularly when dealing with diverse and incomplete project data. This section outlines the methodologies applied to clean, transform, and prepare data for model training and evaluation. The primary objectives of these methods are to handle missing values, encode categorical variables, and structure the data to optimize it for machine learning algorithms.

3.1.1. Technology and algorithms

With the data prepared, the next step involves selecting and implementing the appropriate Technology and Algorithms. The primary programming language for this project is Python, chosen for its flexibility and maintainability. To automate the data processing workflow, we utilize Apache Airflow, minimizing manual intervention and ensuring that the process follows the correct sequence. Data is stored in PostgreSQL, an open-source database that integrates seamlessly with both Apache Airflow and the data processing algorithms employed in the project, transforming processed data into accurate and relevant toolkit suggestions.

- **Libraries and Frameworks:**

During development, we use several scikit-learn libraries to support tasks such as data encoding with TF-IDF and Label Encoding and building prediction models like LightGBM and KNN. These libraries enhance the accuracy and efficiency of data processing and prediction.

- **LightGBM:**

We choose LightGBM as the main model for handling missing data and making predictions. LightGBM processes data row by row rather than column by column, making it suitable for datasets with many missing values.

- **Cosine Similarity:**

To address duplicate issues, we use Cosine Similarity, a common method for measuring similarity between text strings. Combined with TF-IDF, this method helps detect and remove duplicate data rows, ensuring data accuracy.

- **NN with Label Encoding:**

After cleaning and processing the data, we use the Neural Network (NN) model to find suitable toolkits for new projects. This model uses Label Encoding to encode data, helping detect similarities between toolkits and new projects, and making accurate predictions.

Having established the technologies and algorithms that underpin the toolkit recommendation system, we now turn to a critical aspect of data preparation: Handling Missing Data. This section discusses the strategies and techniques employed to identify, address, and mitigate the impact of missing values within the dataset, ensuring that the model remains robust and effective in generating accurate recommendations.

3.1.2. Handling Missing Data

Effective management of missing values is crucial for maintaining data quality. This section explores the techniques employed to identify and address these gaps, including imputation and data cleaning, to ensure reliable data that supports accurate predictions.

This diagram shows the sequence of actions when a data analyst interacts with the Data Pipeline System to automate the process of collecting, processing, and updating project and toolkit data.

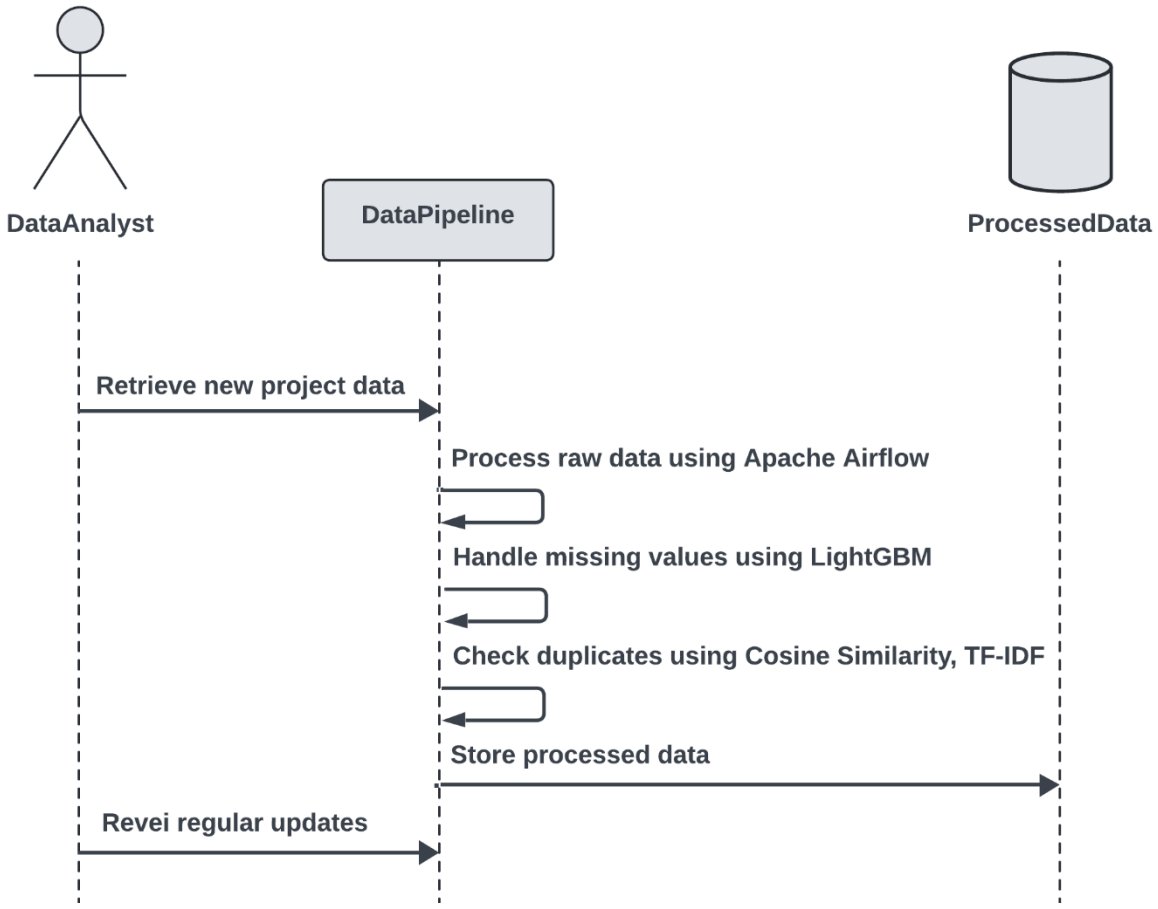

Figure 4: Data Pipeline Management for Continuous Updates diagram

- **Description:**

Allows the data analyst to automate data collection, cleaning, and updating in the recommendation system to ensure that the system continuously operates with up-to-date data.

- **Operation:**
    - The data analyst triggers the system to begin the data pipeline process, ensuring new data is regularly updated.

- The system retrieves new project data from various sources (e.g., forms filled by team members, exports from the old system).
- The system processes the raw data through Apache Airflow, which automates the data pipeline tasks.
- During processing:
  - Missing values are handled using LightGBM and KNN models based on patterns from previous project data.
  - Duplicate records are identified and resolved using TF-IDF and cosine similarity techniques.
- The system stores the cleaned and processed data in the PostgreSQL database.
- The data analyst receives regular notifications confirming that the data has been successfully updated and is now available for toolkit recommendations.

- **Evaluating Model Performance for Predicting Toolkits:**

This diagram shows the sequence of actions when a data scientist interacts with the Model Evaluation Platform to assess the performance of different models for predicting missing data and recommending toolkits.

- **Overview of Handling Missing Data Model**

This is a tree-based algorithm that processes large datasets quickly and efficiently. LightGBM uses leaf-wise splitting rather than level-wise splitting, reducing computation time while maintaining high accuracy. In this project, LightGBM helps handle missing data and predict suitable toolkits for new projects.

- **Deployment of Handling Missing Data Model**

Handling missing values is crucial to ensure complete and accurate data before feeding it into models. We use LightGBM to automatically fill in missing values in data rows. LightGBM uses available attributes within the same data row to predict and fill in missing values, rather than processing entire columns at once. This improves accuracy and ensures no values in important data rows are overlooked.

After addressing missing values, the next crucial step is Handling Duplicate Data. This section outlines the methods used to identify and remove duplicate entries, ensuring the integrity and accuracy of the dataset.

3.1.3. Handling Duplicate Data

In any dataset, duplicates can lead to biased results and skewed analysis. This section discusses the techniques implemented to identify and eliminate duplicate entries in the dataset. By employing methods such as cosine similarity and threshold-based matching, we ensure that the data used for model training and predictions is clean and representative of unique project-toolkit relationships.

- **Overview of Handling Duplicate Data Model**

Cosine Similarity is a metric used to measure the similarity between two vectors by calculating the cosine of the angle between them. It ranges from -1 to 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates complete dissimilarity. This method is commonly used in text analysis and recommendation systems to compare the similarity of two items based on their feature vectors, regardless of their magnitude.

- **Deployment of Handling Duplicate Data Model**

After filling in missing values, we check for duplicate data. Cosine Similarity is the main algorithm used to detect duplicate data rows. This algorithm measures similarity between two text strings by calculating the angle between the vectors representing these strings. If the similarity value is high, the system identifies the data row as duplicate and removes it to ensure no errors in prediction.

Having effectively managed duplicate entries, we now shift our focus to the Prediction Model. This section explores the algorithms and methodologies applied to generate toolkit recommendations, utilizing the cleaned dataset to deliver accurate and relevant predictions based on user project inputs.

3.1.4. Prediction Model

This section details the prediction model developed to recommend toolkits based on user inputs. By leveraging advanced machine learning algorithms and natural language processing techniques, the model analyzes project requirements to provide tailored toolkit suggestions, enhancing decision-making for project managers.

This diagram shows the sequence of actions when a data scientist interacts with the Model Evaluation Platform to assess the performance of different models for predicting missing data and recommending toolkits.

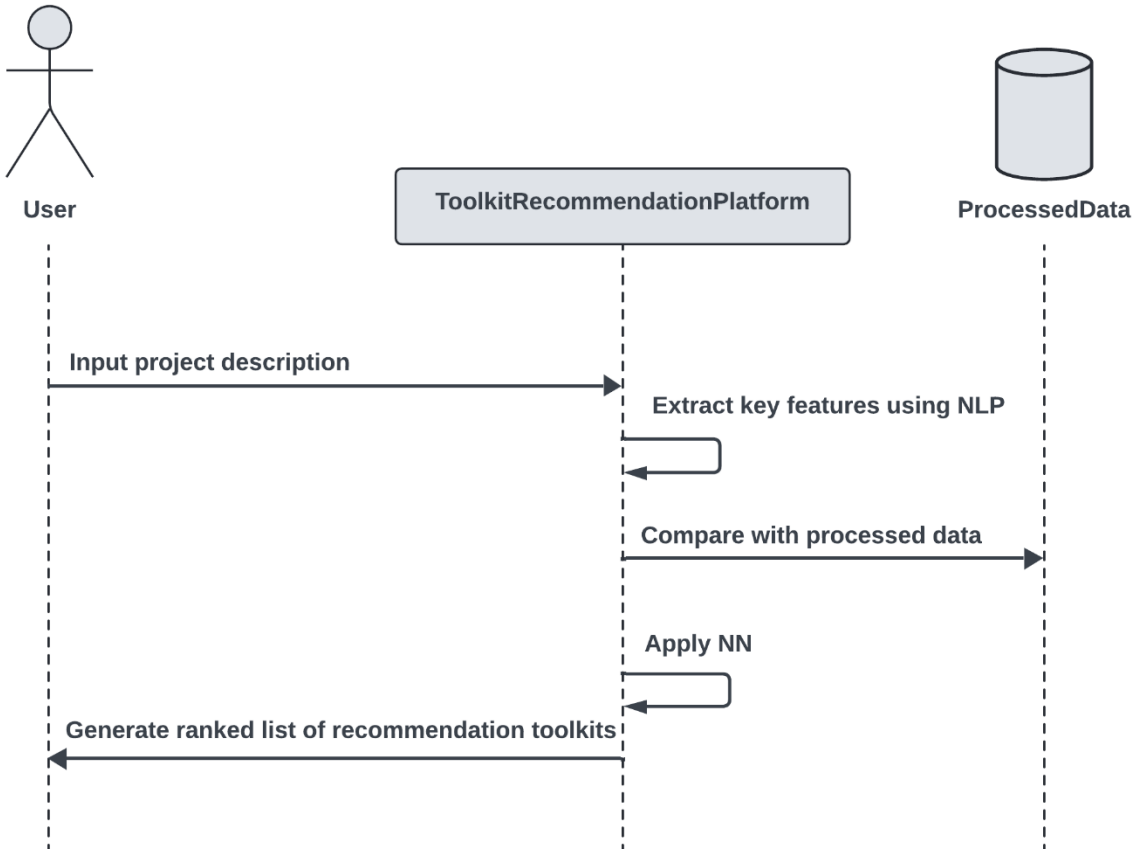

Figure 5: Sequence diagram of Toolkit Recommendation for New Project

- **Description:**

Allows the project manager to input new project information, receive toolkit recommendations, and select the best toolkit.

- **Operation:**

- The project manager enters the new project’s description, including details like the domain and technologies, into the system's interface.
- The system receives the project description and processes it using Natural Language Processing (NLP) to extract key features such as domain, technologies, and programming languages.
- The system then queries its database of previous projects and associated toolkits, looking for similar projects.
- Using NN (Neural Network) and cosine similarity, the system calculates the similarity between the new project and previous projects.
- The system generates a ranked list of recommended toolkits based on similarity scores and displays it to the project manager.
- The project manager reviews the recommended toolkits, sorted by relevance to the project, and selects the most suitable toolkit.
- The system stores the selected toolkit for future use or further actions.

3.1.4.1. Neural Network model

- **Overview of Neural Network model**

Neural Networks (NN) are computational models that mimic human brain function to recognize patterns and make predictions. Each new project input is processed through layers of interconnected neurons, allowing the model to learn complex relationships within the data. Once trained, the NN recommends relevant toolkits for new projects by matching their features with patterns identified in previous projects.

- **Deployment of Neural Network model**

The prediction in this project primarily revolves around using Neural Networks (NN). We utilize data from past projects to predict suitable toolkits for new projects. By leveraging the capabilities of NN, we enhance prediction accuracy and ensure that the recommended toolkits align with the specific requirements of each project.

Having established the prediction model using Neural Networks (NN), we now turn our attention to the Encoding Strategy Features. This section discusses the techniques employed to convert categorical and textual data into a numerical format that the NN can effectively process, ensuring that all relevant features are adequately represented in the model.

3.1.4.2. Encoding Strategy Features

Since data is not always in numerical form, we use two main encoding methods to convert data into a numeric format that models can process:

- **Label Encoding:**

This method converts categorical values into integer values. It is primarily used for LightGBM and KNN, as these models require numeric data for calculations.

- **TF-IDF (Term Frequency - Inverse Document Frequency):**

This method converts text strings into numeric vectors based on word frequency. TF-IDF is mainly used for Cosine Similarity to measure the similarity between text lines and to detect duplicate data.

After implementing the encoding strategies, we now focus on the NLP Features. This section explores the natural language processing techniques applied to extract meaningful insights from textual data, enhancing the model's ability to understand and analyze project descriptions effectively.

3.1.4.3. Natural Language Processing (NLP) Features

In this project, we use various Natural Language Processing (NLP) techniques to analyze text descriptions of projects and toolkits. One key technique is TF-IDF, which converts descriptions into numeric vectors for comparison in the computation space. NLP also helps extract important keywords from project and toolkit descriptions, improving the accuracy of prediction models like KNN and LightGBM.

With the NLP features integrated into the model, we now turn to Model Evaluation Methods. This section outlines the metrics and techniques used to assess the performance

of the prediction model, ensuring that the toolkit recommendations are accurate and reliable based on the processed data.

3.1.5. Model Evaluation Methods

After predictions are complete, toolkits are sorted and filtered based on their similarity to the new project. We use evaluation metrics like RMSE (Root Mean Squared Error) and F1 Score to test prediction accuracy. The system provides the project team with a list of the most suitable toolkits based on KNN and LightGBM predictions, helping them quickly choose the necessary tools for the project.

This diagram shows the sequence of actions when a data scientist interacts with the Model Evaluation Platform to assess the performance of different models for predicting missing data and recommending toolkits.

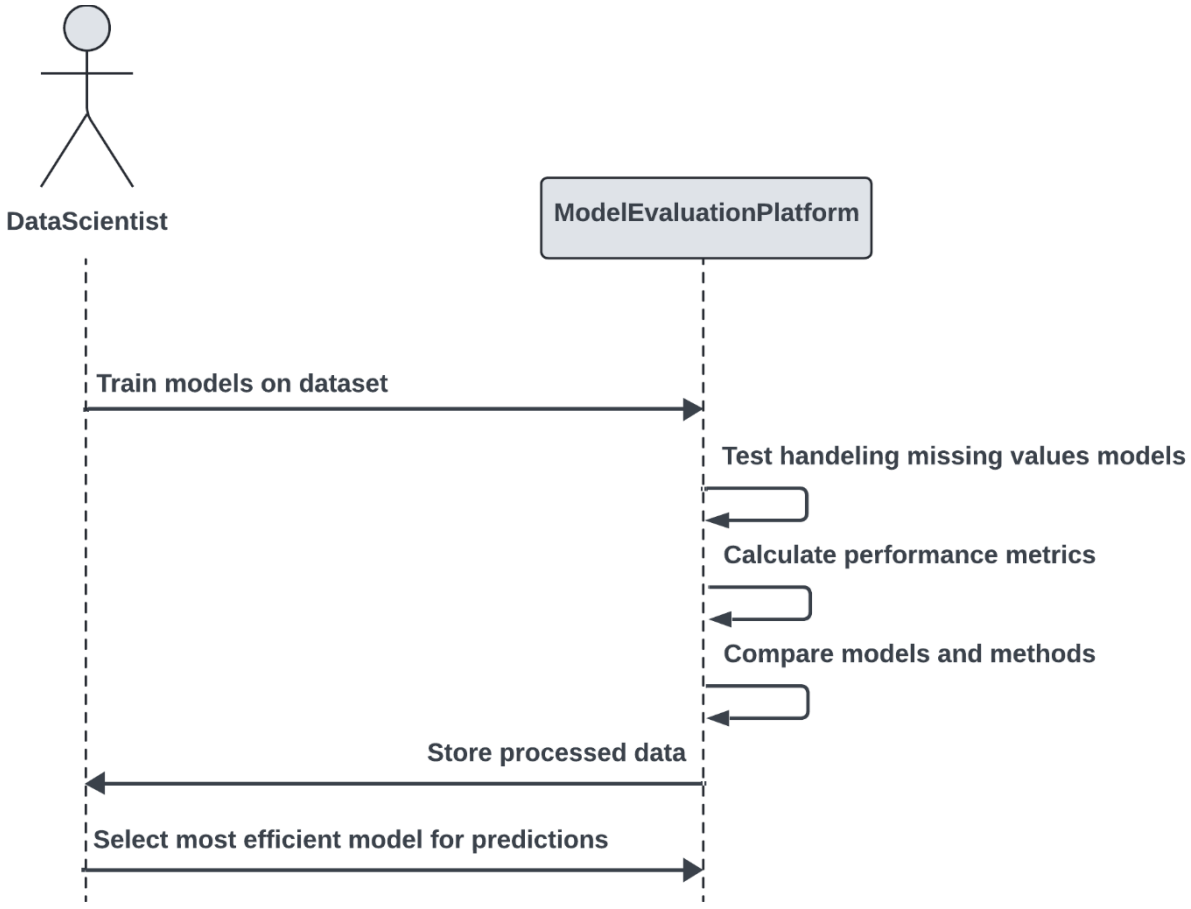

Figure 6: Sequence diagram of Evaluating Model Performance for Predicting Toolkits

- **Description:**

Allows the data scientist to evaluate the performance of various models (e.g., LightGBM, Neural Network) to determine their effectiveness in predicting missing data and recommending toolkits.

- **Operation:**
    - The data scientist initiates the model training process by providing the system with a dataset of previous projects and toolkits.
    - The system trains different models, including LightGBM and Neural Network, on the provided dataset to predict missing values in toolkit data.
    - After training, the system calculates key performance metrics:
        - RMSE (Root Mean Square Error)
        - MAE (Mean Absolute Error)
        - F1-score
    - The system compares the results from both the Sequential Prediction Method and Simultaneous Prediction Method for handling missing data.
    - Based on the calculated metrics, the system identifies the model that best fits the data and recommends which model should be used in production.
    - The data scientist selects the most efficient model, and the system is updated to utilize the best-performing model for future toolkit predictions

Having evaluated the model's performance through various metrics, we now shift our focus to Website Development and System Integration. This section details how the prediction model is integrated into a user-friendly web platform, enabling seamless interactions between users and the toolkit recommendation system.

3.2. Website Development and System Integration

This section outlines the development of the Toolkit Data Warehouse website, designed to offer users an intuitive platform for searching and discovering toolkits tailored to various

projects. Built with a combination of technologies, the system ensures scalability, efficiency, and ease of use. Its architecture comprises several key components, including:

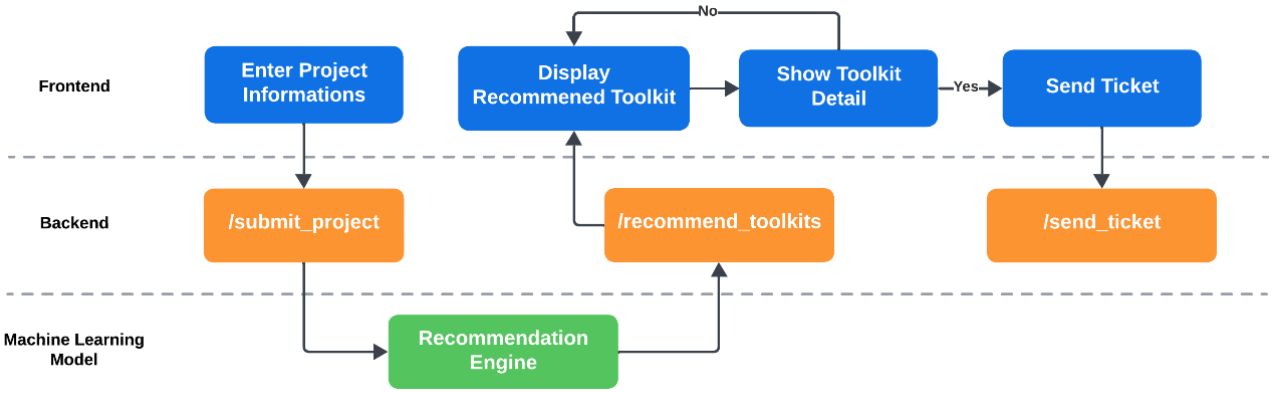

Figure 7: Platform Architecture

In conclusion, the integration of the prediction model within the Toolkit Data Warehouse website facilitates seamless user interactions and enhances the overall functionality of the platform. The following section delves into the specific methods employed in the website development process, detailing the technologies and approaches that contributed to its design and implementation.

3.2.1. Website Development Methods

A variety of methods were employed to ensure effective website development, focusing on both functionality and user experience. Among these methods, particular attention was given to the User Interface (UI), which plays a vital role in how users interact with the Toolkit Data Warehouse. The next section will explore the design principles and components that shaped the frontend of the website, ensuring it is both intuitive and visually appealing.

3.2.1.1. User Interface (Frontend)

- **Web Application:** Developed using HTML, CSS, and JavaScript, providing a seamless user experience for project managers.

- **Components:**
    - **Project Information Form:** Allows users to input details about the new project (e.g., domain, technologies).
    - **Results Display:** Shows the recommended toolkits based on user input.
    - **Toolkit Selection:** Users can choose their preferred toolkit from the recommendations.
    - **Action Section:** For users to indicate if they want to send a ticket or select another toolkit.

3.2.1.2. Application Server (Backend)

- **Flask Server:**

Handles incoming requests from the frontend and manages the backend logic.

- **Endpoints:**
    - **/submit_project**: Receives the project information from the frontend.
    - **/recommend_toolkits**: Processes the input data to find similar projects and recommend toolkits.
    - **/send_ticket**: Manages requests to send feedback or tickets to the author.

3.2.1.3. Machine Learning Model

- **Recommendation Engine:** Executes the models (e.g., KNN, cosine similarity) to find similarities based on the project description.
- **NLP Processing:** Analyses user input to extract relevant features for accurate recommendations.

3.2.1.4. Data Flow

- **User Input:** The user enters new project information into the frontend.
- **Data Transmission:** The frontend sends the project data to the Flask backend.
- **Processing:** The backend receives the input, extracts relevant features using NLP, and applies the recommendation engine to find similar projects.

- **Result Generation:** The backend generates a ranked list of recommended toolkits based on the similarity scores.
- **Response:** The backend sends the recommended toolkits back to the frontend for display.
- The user reviews the recommended toolkits displayed on the frontend.
- If the user selects a toolkit:
    - The frontend sends a request to the /**send_ticket** endpoint to notify the author.
- If the user decides not to select any toolkit, they can return to the results display to choose another recommendation.

In summary, the website development methods employed have laid a solid foundation for creating an effective and user-friendly platform. These methods ensure that users can navigate seamlessly while accessing toolkit recommendations. The subsequent section will illustrate a practical use case, demonstrating how users can interact with the Toolkit Data Warehouse to fulfill their project needs.

3.2.2. Use Case

This section presents a practical scenario demonstrating how users interact with the Toolkit Data Warehouse. It outlines the steps involved in accessing toolkit recommendations and highlights the system's functionality in addressing specific project requirements.

The system developed for the Toolkit Data Warehouse aims to streamline the process of selecting appropriate toolkits for new projects, allowing users to input project information, receive tailored toolkit recommendations, and submit requests for toolkits efficiently. This use case demonstrates how users can interact with the system, beginning by entering details about a new project, selecting a recommended toolkit, and sending a ticket to request access or further information about the selected toolkit.

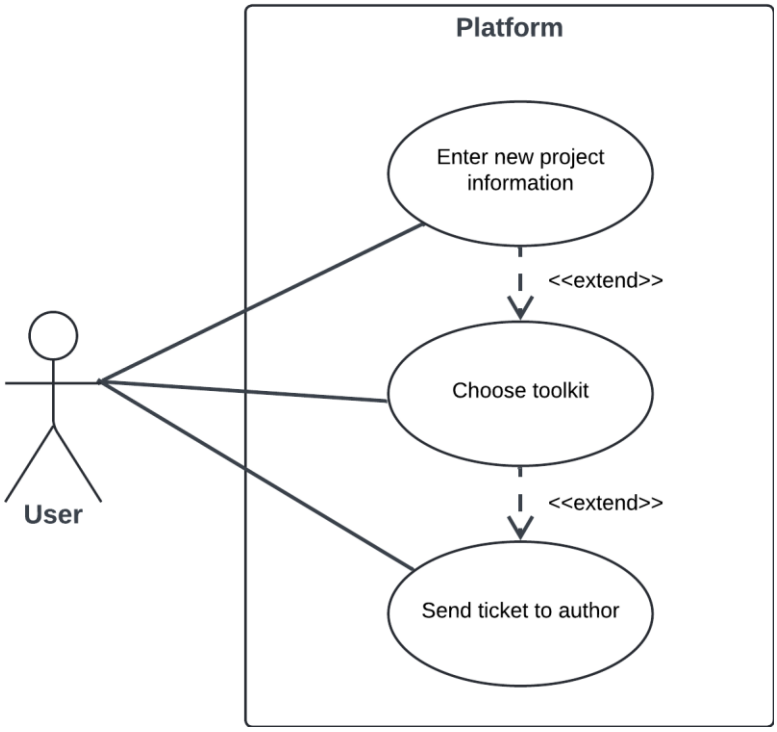

Figure 8: Use Case Diagram

The process consists of three main steps:

- **Step 1 (Enter New Project Information):** Users provide details about their project, including domain, technology, and any specific requirements. The system then processes this input to generate relevant toolkit recommendations.
- **Step 2 (Choose Toolkit):** Based on the recommendations provided, users can review and select the most appropriate toolkit for their project.
- **Step 3 (Send Ticket):** After choosing a toolkit, users can send a ticket requesting access to the toolkit or additional information. The ticket submission ensures that the appropriate team is notified to assist with the next steps.

This use case ensures a smooth flow from project initiation to toolkit selection and access, enhancing user experience and system efficiency.

Name	Enter Project Information, Choose Toolkit, Send Ticket
Actor	User
Preconditions	- The user has access to the platform. - The platform has stored project and toolkit data. - Toolkit recommendation models are operational.
Trigger	User initiates a new project on the platform by entering project details.
Basic Flow	1. User enters new project information. 2. The system processes the input and generates toolkit recommendations. 3. User reviews and selects a toolkit.
Extensions	- If the system cannot find suitable recommendations, it suggests refining the project description. - The system allows the user to send a support ticket.
Postconditions	- User has submitted a ticket to request further action. - The selected toolkit has been logged for potential usage.
Alternative Flow	User can return to the project information step if they want to change the details and request new recommendations.

Table 1: Requirement Specification for Use Case

This table outlines the steps of the use case, from entering project details to submitting a ticket, ensuring clarity on the interaction between users and the system.

In conclusion, the Materials and Methods outlined in this project provide a robust framework for developing the Toolkit Data Warehouse and its associated prediction model. By integrating various technologies and employing effective data handling techniques, we have established a system capable of delivering accurate toolkit recommendations tailored to user projects.

The comprehensive approach taken in the development and implementation phases ensures that the platform not only meets user needs but also maintains high standards of data quality and usability.

The next section, Results and Discussion, will delve into the outcomes of our methodology, analyzing the effectiveness of the prediction model and the overall performance of the Toolkit Data Warehouse. This analysis will provide insights into the system's strengths and areas for improvement, guiding future enhancements and applications.

III/ RESULTS AND DISCUSSIONS

This section provides an overview of the results obtained from the Toolkit Data Warehouse and the prediction model. We will analyze the performance metrics, evaluate the accuracy of the toolkit recommendations, and discuss the implications of these findings. Insights gained from user interactions and data analysis will be highlighted, along with recommendations for future improvements and enhancements to the system.

1. Results

The findings from the Toolkit Data Warehouse reveal valuable insights into the effectiveness of the prediction model. A detailed analysis of the model's performance metrics demonstrates its ability to provide accurate toolkit recommendations based on user inputs.

Next, we will focus on the processing model results, examining specific metrics and outcomes that highlight the strengths and areas for improvement in the model's predictive capabilities.

1.1. Processing model result

Building on the insights gained from the processing model results, we now turn our attention to the performance of the various models employed in the Toolkit Data Warehouse. This section evaluates their effectiveness in delivering accurate toolkit recommendations and compares the results across different modeling approaches.

1.1.1. Performance of Models

To determine the most suitable model and algorithm for our project, we experimented with various methods for handling missing data. Specifically, we focused on two primary models: LightGBM and Neural Networks. These models were selected based on their capabilities to handle missing data and their accuracy in predictions.

- **Performance of LightGBM**

As we said before, for datasets with missing values across multiple columns, two approaches can be employed:

- **LightGBM in Sequential Method:**

In the Sequential Method, two primary approaches were tested for handling missing values: a Column-by-Column approach and a Row-by-Row approach.

- **Column-by-Column Approach:**

The Column-by-Column method focused on processing each column of missing data independently. For each column with missing values, LightGBM was trained using all other columns as features to predict the missing values in the target column. The data was split into training and validation sets, with 80% used for training and 20% for validation.

During the training process, LightGBM's parameters were tuned to optimize the model, and early stopping was employed to prevent overfitting by stopping the training if the validation loss did not improve after 50 rounds.

However, the Column-by-Column approach had limitations when columns such as the 'Rate' column, which had weak correlations with other features, were used as a target. The model often struggled to predict accurately, leading to overfitting on certain columns and generally poor performance. For instance, using the 'Rate' column for prediction yielded unsatisfactory results due to its weak relationship with other columns. Despite tuning the model, the validation loss frequently indicated overfitting during training.

- **Row-by-Row Approach:**

Given the limitations observed in the Column-by-Column approach, the Row-by-Row method was tested. In this approach, LightGBM was applied to each row independently. Instead of using an entire column to predict missing values, we leveraged the available data within a single row to predict its missing values. This allowed the model to utilize the relationships within each row, which proved to be stronger compared to inter-column relationships.

To prevent overfitting and enhance generalization, an 80/20 train-validation split was used. The model was trained on complete data from other rows and evaluated on withheld validation data.

- **LightGBM in Simultaneous Prediction:**

In the Joint Modeling approach, all columns with missing values were predicted simultaneously. LightGBM was trained to predict missing values in multiple columns at once by utilizing the non-missing columns as input features. This method treated all available data together, allowing the model to learn patterns across multiple variables.

For training, the data was split into training and validation sets, with 80% of the data used for training and 20% for validation. LightGBM's parameters were tuned for optimal performance, and early stopping was applied after 100 rounds to prevent overfitting if the validation loss did not improve. The model's performance was monitored throughout training, with RMSE used as the key metric to evaluate accuracy.

- **Performance of Neural Network**
  - **Neural Network with Column-by-Column:**

During training, the model was monitored using early stopping, which halts the training process when the validation loss stops improving. The learning curve showed gradual decreases in both training and validation losses over time, although the model often plateaued before achieving significant improvements. The use of early stopping was crucial to prevent overfitting, especially when predicting a single column at a time.

However, even with early stopping, the model's performance did not yield satisfactory results, as evidenced by the metrics. Despite training for up to 50 epochs per column, the limited gains suggest that the model's architecture or the column-by-column approach might not have been sufficient to capture the underlying patterns in the data.

1.1.2. Comparison of Results and Evaluation Metrics

To assess the performance of the models, we used metrics such as RMSE (Root Mean Squared Error), MAE, and F1 Score.

Model	RMSE	MAE	R^2
LightGBM Column-by-Column	1.2886	0.9886	0.2486
LightGBM Row-by-Row	0.5310	0.4024	0.8652
LightGBM Simultaneous Prediction	1.2408	0.9500	0.2674
Neural Network Column-by-Column	1.4364	1.1698	0.0664

Table 2: Result of Processing Models

- **LightGBM Column-by-Column**

This approach evaluates each column independently when predicting missing values. While the model achieved moderate results, the relatively high RMSE and MAE values indicate significant deviations between the predicted and actual values. The R^2 score of 0.2486 also suggests that only about 24.86% of the variance in the target variable is explained by the model. This indicates a weaker fit and lower accuracy in prediction.

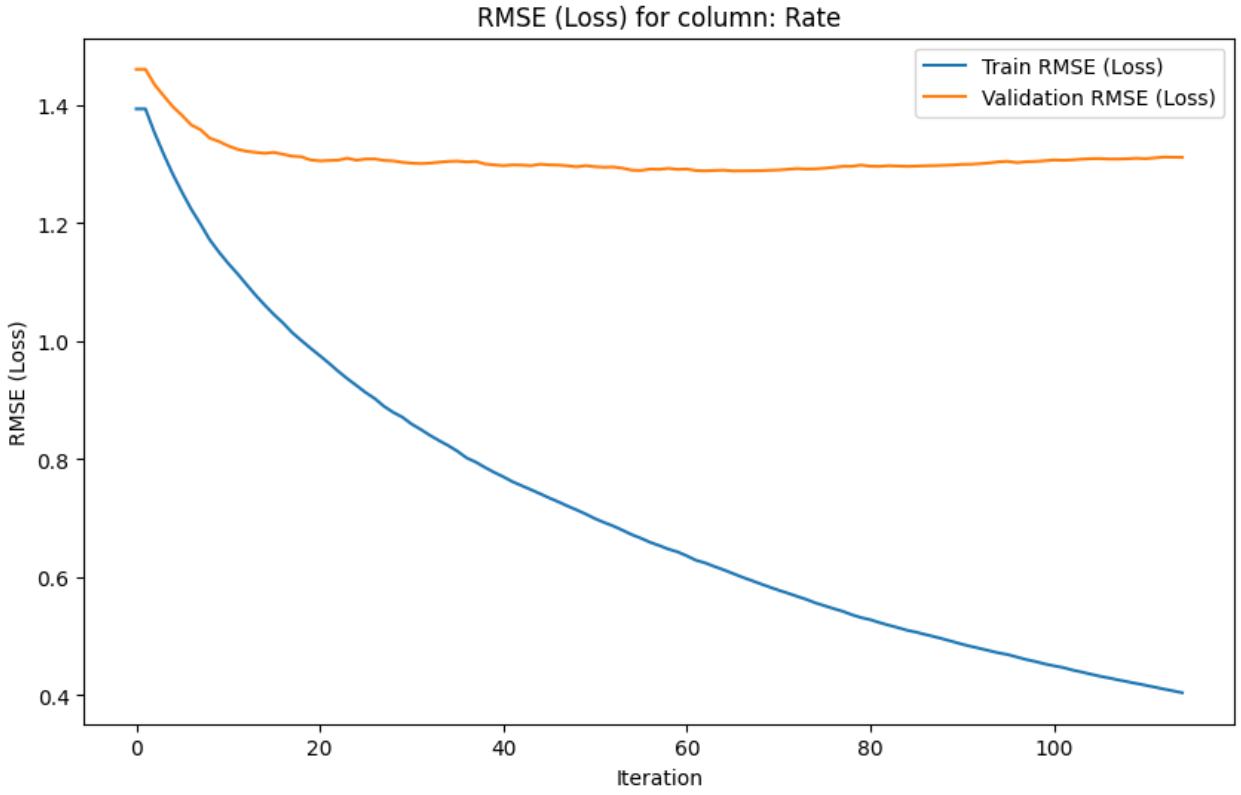

Figure 9: LightGBM Column-by-Column validation loss

- **LightGBM Row-by-Row**

The row-by-row approach predicts missing values while considering all available features in each row simultaneously. This method significantly outperformed the column-by-column approach, with an RMSE of 0.5310 and an MAE of 0.4024, indicating smaller prediction errors. The high R^2 value of 0.8652 reveals that 86.52% of the variance in the target variable is explained by the model, demonstrating a much better fit and higher predictive power.

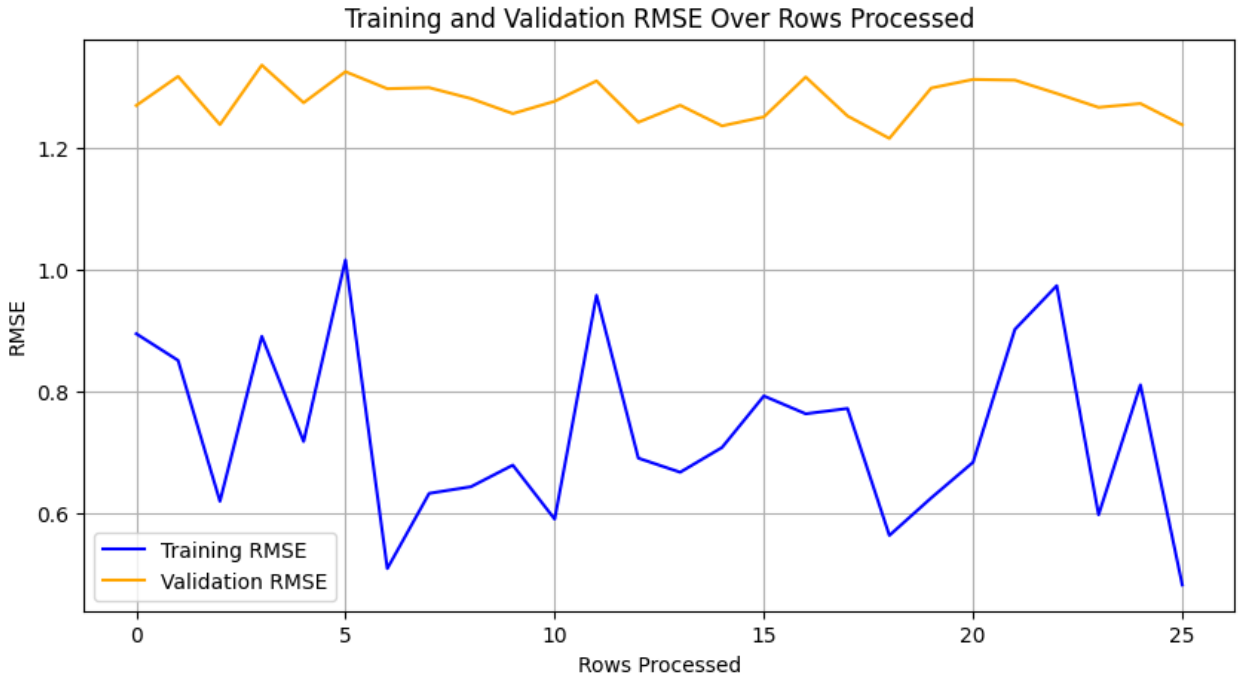

Figure 10: LightGBM Row-by-Row validation loss

- **LightGBM Simultaneous Prediction**

In this joint modeling approach, LightGBM was used to predict missing values across multiple columns simultaneously. The model achieved an RMSE of 1.2408 and an MAE of 0.9500, reflecting moderate prediction errors. However, the R^2 value of 0.2674 indicates that only 26.74% of the variance in the target column ('Rate') was explained by the model. This result shows that while the model performed slightly better than the column-by-column approach, it still struggled to capture enough relevant information to make highly accurate predictions.

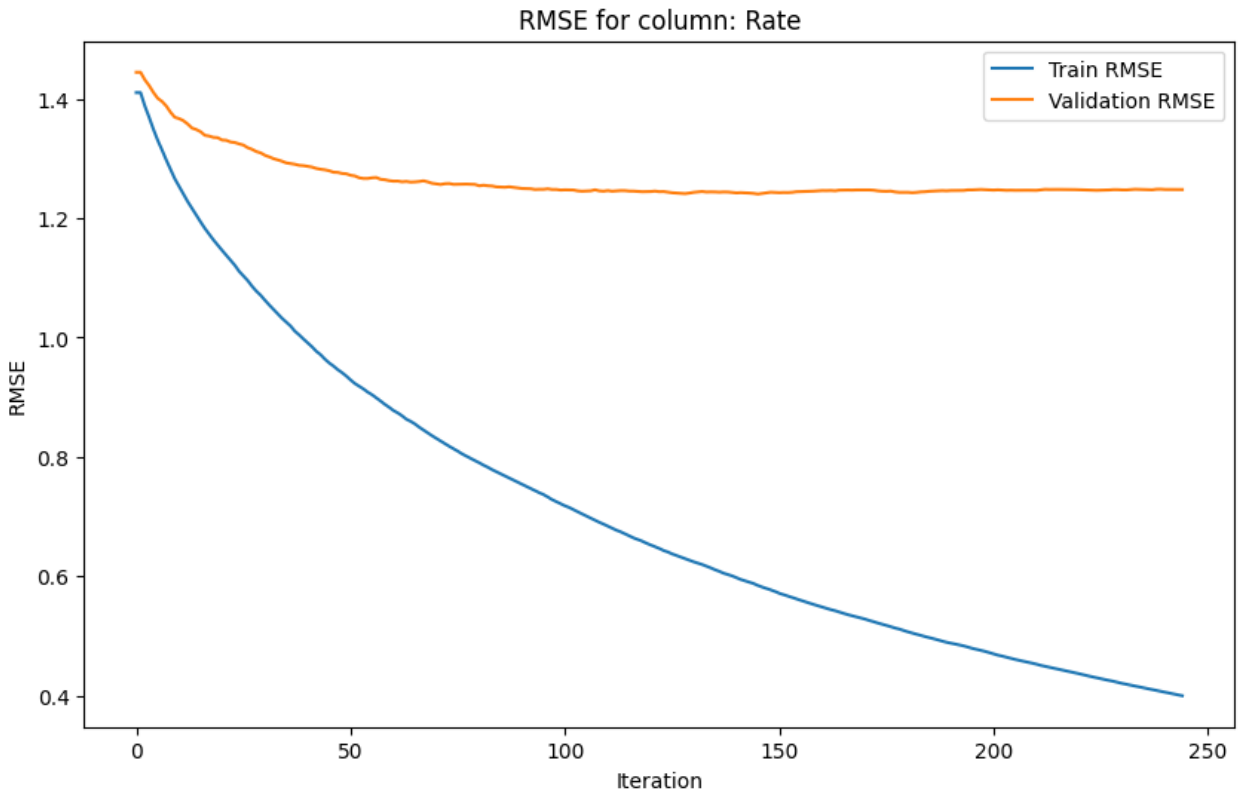

Figure 11: LightGBM Simultaneous validation loss

- **Neural Network Column-by-Column**

The column-by-column approach predicts missing values one column at a time, focusing on each column in isolation. This method resulted in an RMSE of 1.4315 and an MAE of 1.1492, indicating relatively larger prediction errors. The low R^2 value of 0.0727 shows that only 7.27% of the variance in the target variable is explained by the model, indicating poor fit and low predictive power. Despite training for multiple epochs with early stopping, the model struggled to achieve satisfactory results across the columns.

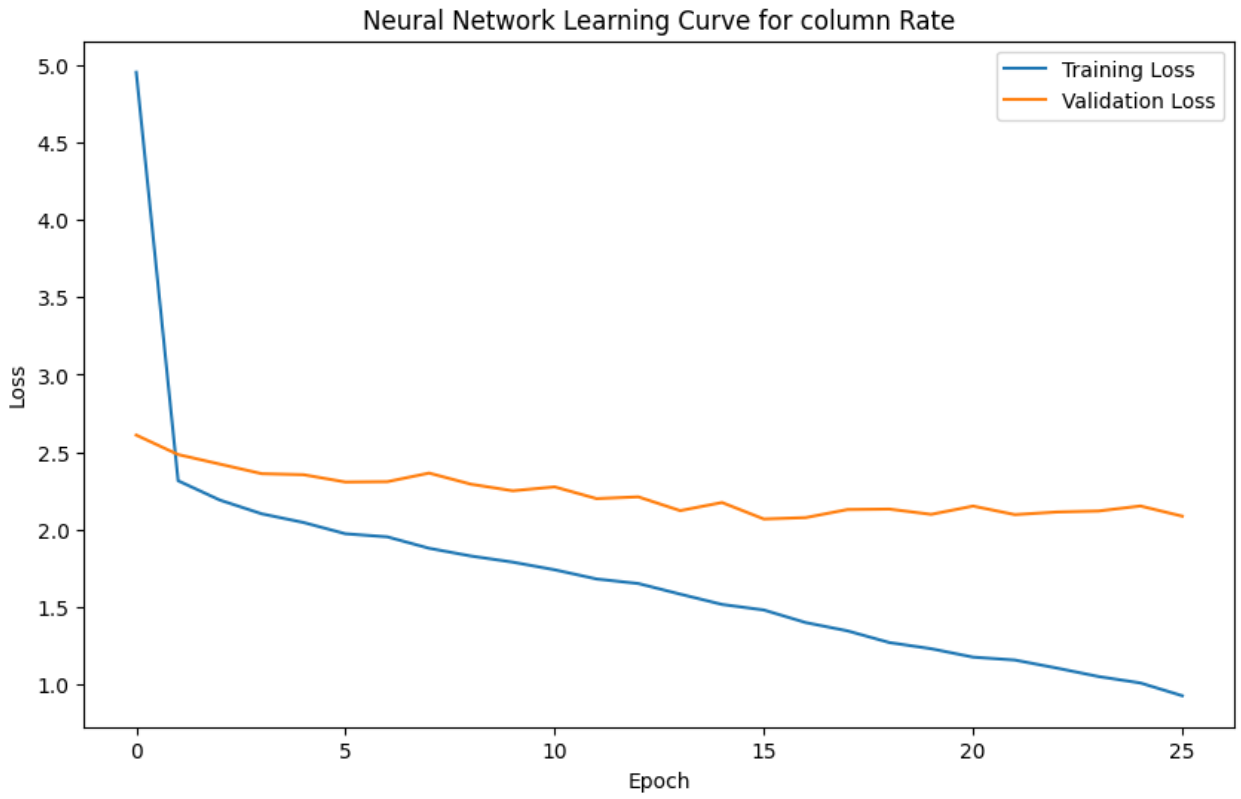

Figure 12: Neural Network Column-by-Column validation loss

The LightGBM Row-by-Row method gives the most accurate results with the lowest error values and the highest R^2, making it the best option. On the other hand, both LightGBM Column-by-Column and Neural Network Column-by-Column methods do not perform well, likely because they cannot handle interactions between features effectively. The Simultaneous Prediction method improves slightly but still falls behind the row-by-row approach.

In summary, while the models demonstrated promising performance in providing accurate toolkit recommendations, several challenges and difficulties emerged during the development and implementation phases. The next section will explore these challenges in detail, highlighting the obstacles encountered and the lessons learned throughout the project.

1.1.3. Challenges and difficulties

The development of the Toolkit Data Warehouse presented various challenges, particularly in the area of prediction methodologies. One significant aspect was the decision between sequential and simultaneous prediction approaches. This section examines the complexities associated with each method, discussing their implications for model performance and accuracy in generating toolkit recommendations.

1.1.3.1. Sequential vs. Simultaneous Prediction

Predicting missing values in datasets with incomplete information poses a significant challenge. Dealing with missing values in datasets can be complex, especially when they span multiple columns [1]. Two main methods can be used to address this:

- **Sequential Method (One-by-One Prediction):**

This approach involves predicting one column of missing values at a time. It assumes that each column can be predicted based on the data available in other columns. However, this method can be problematic because it may not consider the relationships between rows and columns comprehensively [2].

- **Simultaneous Prediction (Multi-Target Learning) [3] [4]:**

This method aims to predict missing values for all columns at once, taking into account the interdependencies between different columns. While it is more holistic, it requires complex models that can handle multiple missing values simultaneously.

1.1.3.2. Row-Based vs. Column-Based Approach

Another issue with the sequential method is the approach to data. There are two main ways to handle the data:

- **Column-Based Approach:**

This approach treats each column independently. It is suitable when there is a strong relationship between values in a single column but may overlook relationships between different columns.

- **Row-Based Approach:**

This method considers entire rows of data. It is useful when the values in a row are interrelated. However, it may fail to capture relationships within a single column, which can be problematic if the data has strong column-based dependencies.

1.1.3.3. Issues with Neural Networks in Sequential Methods [5]

In an attempt to impute missing values using a Neural Network (NN), we encountered several challenges that impacted the effectiveness of the approach. The primary issue was the presence of NaN values in the training data, which caused the model to skip rows and fail to produce reliable imputations. Despite preprocessing steps that included encoding categorical variables and normalizing data, the NN model struggled with missing values in both the features and target columns.

Due to these issues, performance metrics such as RMSE, MAE, and R-squared could not be accurately calculated for all rows. The inability to train the model effectively on rows with NaN values resulted in incomplete imputation and highlighted the need for alternative methods. Future work could explore different imputation strategies, such as using simpler models or advanced techniques like matrix factorization, to better handle missing data in similar contexts.

In summary, the analysis of the processing model results has shed light on the effectiveness of our prediction methodologies. By evaluating the various approaches used, we have gained insights into their strengths and areas for improvement.

Building on these findings, we now shift our focus to the Prediction Model Results, where we will assess the overall performance of the models and their ability to provide accurate toolkit recommendations for users.

1.2. Prediction Model Results

This section presents the results of the various prediction models employed within the Toolkit Data Warehouse. We will evaluate the performance metrics of each model, analyzing their accuracy in generating toolkit recommendations. Additionally, comparisons

will be drawn to identify the most effective approaches for optimizing the user experience and meeting project requirements.

1.2.1. Performance of Models

Similarly to data processing, we applied various models and algorithms to predict suitable toolkits. The general approach involves comparing and identifying commonalities between the new project and existing projects in the data repository. The toolkit with the most commonalities with the new project is considered the most suitable, with lower similarity results being ranked subsequently. We experimented with three models and algorithms: Cosine Similarity, Neural Networks, and K-Nearest Neighbors (KNN). LightGBM, while effective, is prone to overfitting with limited new project information.

Given that raw data cannot be directly processed by models and algorithms, we employed libraries such as TF-IDF and Label Encoding to convert the data into numerical format. Detailed discussions on model-specific library usage will be provided in the results section.

1.2.1.1. Cosine Similarity and TF-IDF

Cosine Similarity, paired with TF-IDF, was used to compare project descriptions and identify the most suitable toolkits. TF-IDF converts text into numerical form, emphasizing important terms. By calculating the cosine of the angle between project vectors, we ranked toolkits based on similarity to the new project. This method worked well for comparing textual data but has limitations in capturing deeper feature interactions. Despite this, it served as a reliable baseline for quick toolkit recommendations.

1.2.1.2. KNN and TF-IDF

KNN performed well with small datasets such as new project data. The use of TF-IDF and Label Encoding showed minimal differences, so the focus was on analyzing results with TF-IDF. However, results using both TF-IDF and Label Encoding will be included in the comparison section for completeness.

1.2.1.3. Neural Network and Label Encoding

We used a Neural Network model with Label Encoding to predict suitable toolkits. However, after testing, the model showed weak performance in this context, with higher error rates and lower accuracy compared to other methods. As a result, it was less effective for toolkit recommendations in this phase.

1.2.2. Comparison of Results and Evaluation Metrics

In this section, we present a detailed comparison of the models based on evaluation metrics such as Precision, Recall, F1 Score, and Accuracy. These metrics help assess the effectiveness of each model in predicting suitable toolkits for new projects.

1.2.2.1. Performance Analysis

Model	Precision	Recall	F1 Score	Accuracy
Cosine Similarity (TF-IDF)	0.0224	0.0244	0.0228	0.0244
KNN (TF-IDF)	0.0224	0.0244	0.0228	0.0244
NN (Label Encoding)	0.0292	0.040	0.0297	0.0400

Table 3: Result of Prediction Models

- **Cosine Similarity (TF-IDF):**

Cosine Similarity using TF-IDF has the lowest Precision and Recall among the models. The F1 Score, which balances Precision and Recall, also reflects this performance. While Cosine Similarity provides a straightforward method for ranking based on similarity, its effectiveness is limited when dealing with complex or nuanced project descriptions.

- **KNN (TF-IDF):**

KNN with TF-IDF shows similar performance to Cosine Similarity in all metrics. KNN is generally effective for small datasets but struggles with larger or more complex data where feature scaling and dimensionality can impact its performance. The model's reliance on distance metrics may not capture all the relevant features needed for accurate recommendations.

- **Neural Network (Label Encoding):**

The Neural Network model outperforms both Cosine Similarity and KNN in Precision, Recall, F1 Score, and Accuracy. This improvement suggests that Neural Networks, with their capacity for capturing complex patterns and relationships in the data, can better handle the intricacies of project descriptions. However, it's important to note that the performance might still be limited by the quality of the input data and the model's parameter tuning.

1.2.2.2. Comparative Analysis

- **Cosine Similarity vs. KNN:**

Both models exhibit comparable performance metrics. However, KNN's performance may improve with a larger dataset or when additional feature engineering is applied. The primary difference is in how each method calculates similarity - Cosine Similarity relies solely on text vector similarity, while KNN incorporates distance metrics, which may affect results in different scenarios.

- **Cosine Similarity/KNN vs. Neural Network**

The Neural Network model shows a clear advantage over both Cosine Similarity and KNN. Its higher Precision and Recall indicate better performance in accurately identifying relevant toolkits. Neural Networks are able to leverage complex data relationships and hidden patterns, leading to better overall accuracy. This model's ability to handle non-linear relationships and high-dimensional data provides a significant edge in predicting suitable toolkits.

1.2.2.3. Conclusion

The results suggest that while simpler methods like Cosine Similarity and KNN are useful for initial evaluations or smaller datasets, the Neural Network model provides more robust performance for handling complex and high-dimensional data. For applications where accuracy and nuanced understanding of project descriptions are critical, Neural Networks should be considered despite their complexity and potentially higher computational requirements.

- **Recommendations:**
    - **For Small or Simple Datasets: Cosine** Similarity or KNN may suffice and offer easier implementation with lower computational costs.
    - **For Complex or High-Dimensional Data:** Neural Networks are recommended due to their superior ability to model intricate patterns and relationships in the data.

In summary, the choice of model should be guided by the specific needs of the application, the nature of the data, and the desired level of accuracy. Further experimentation and tuning may be necessary to optimize model performance based on real-world requirements and constraints.

1.2.3. Challenges and Difficulties

1.2.3.1. Handling Multilingual Data

The current project operates entirely in English, but real-world datasets often include information in other languages, such as Vietnamese [7]. This presents several challenges:

- **Stopwords in Vietnamese:**

English and Vietnamese have different stopwords (common words that are often ignored in text processing). Developing a stopword list for Vietnamese is essential for accurate text analysis, but this task can be complex due to the differences in language structure and usage.

- **Punctuation Differences [6]:**

English text typically lacks punctuation, whereas Vietnamese text often includes various punctuation marks. This difference affects text processing and can impact the accuracy of NLP models.

1.2.3.2. Validation of Vietnamese Text

Currently, there is no effective method for validating the accuracy of text analysis models with Vietnamese data. This limitation means that the model's performance may be less reliable when applied to datasets that include Vietnamese text.

1.2.3.3. Adaptation to Mixed Language Data

The project needs to handle data that may include both English and Vietnamese text. Adapting models to process and analyze mixed-language data is challenging, as it requires integrating different language processing techniques and ensuring that the model can handle variations in text formats and structures.

In conclusion, the analysis of the prediction model results highlights the effectiveness of our methodologies in delivering accurate toolkit recommendations. The insights gained from these evaluations not only demonstrate the models' strengths but also provide direction for future enhancements.

With a solid understanding of the prediction capabilities, we now turn our attention to the Website Development and System Integration. This section will explore how the results from the prediction models have been integrated into the website's architecture, ensuring a seamless user experience while accessing toolkit recommendations.

1.3. Website

This section outlines the development and integration of the Toolkit Data Warehouse website, designed to provide users with an intuitive platform for accessing toolkit recommendations. It details the technologies employed, the architecture of the system, and how the predictive capabilities of the models have been seamlessly integrated into the user interface to enhance user experience and facilitate efficient toolkit discovery.

1.3.1. Performance

The website was effectively implemented, meeting basic requirements such as input fields for users, displaying recommended toolkits in order, and providing detailed information about each toolkit, along with a function to send tickets to the toolkit authors. However, there are still many issues that need further optimization to ensure stable performance.

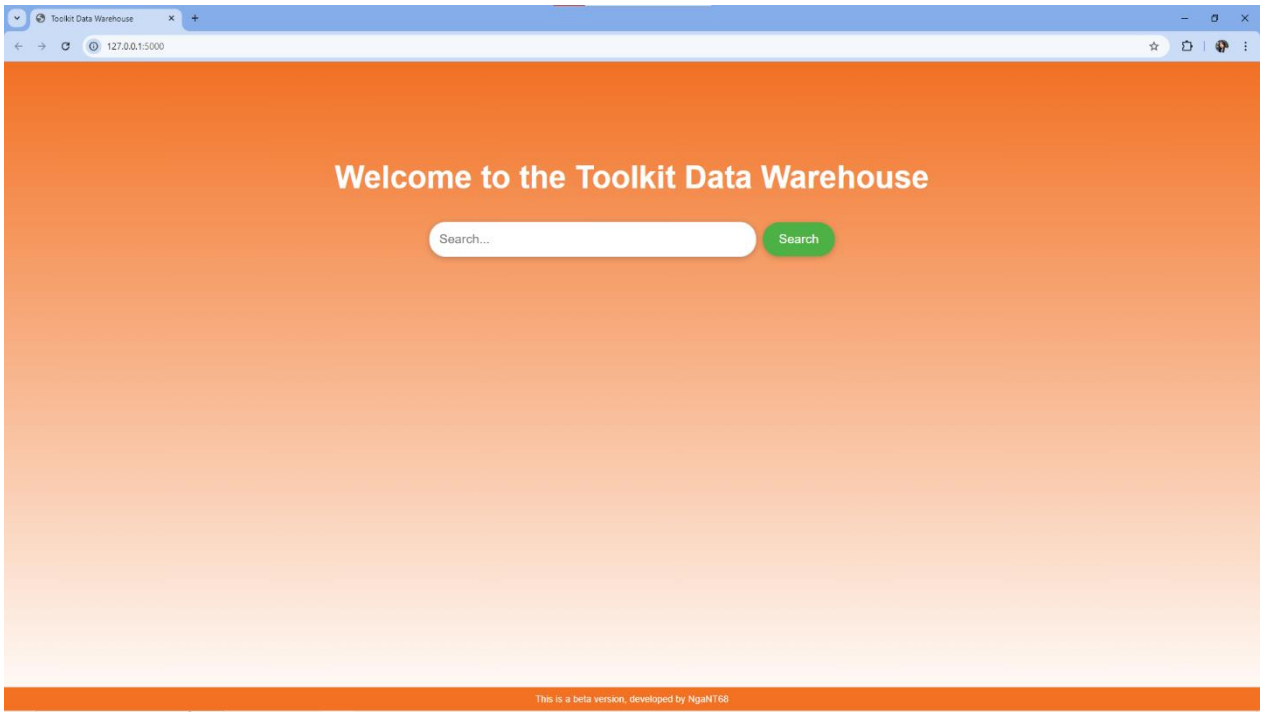

Figure 13: Platform First User Interface

1.3.2. Challenges and Difficulties

Several challenges arose during the website's development. Integrating the machine learning models into the web interface was complex due to differences in data formats and model outputs. Additionally, ensuring compatibility across various browsers and devices required extensive testing and adjustments.

A significant challenge was the limitation on the number of toolkits displayed; currently, the UI supports showing only six toolkits. If the number exceeds six, it causes errors in the interface. Moreover, maintaining real-time responsiveness, particularly when the system had to process larger datasets, affected user experience.

In summary, the development and integration of the Toolkit Data Warehouse website have established a user-friendly platform that effectively leverages the predictive models for toolkit recommendations. The thoughtful design and implementation ensure that users can easily navigate the system and access the information they need.

As we move forward, the Discussion section will delve into the broader implications of these developments, examining how the results and user feedback can inform future enhancements, address potential limitations, and guide the evolution of the Toolkit Data Warehouse.

2. Discussions

This section explores the implications of the findings from the Toolkit Data Warehouse, focusing on the effectiveness of the prediction models and their integration into the website. Key insights will be drawn from user interactions and model performance, guiding future improvements.

To begin, we will examine the data processing methods utilized in the project, discussing their impact on the overall functionality and accuracy of the toolkit recommendations.

2.1. Data Processing

Data processing is vital for the Toolkit Data Warehouse, impacting the accuracy of toolkit recommendations. This section details the methods used to handle missing and duplicate data, select prediction models, and evaluate their performance, ensuring efficient system operation.

2.1.1. Processing Model

The handling of missing and duplicate data was effectively managed using the described methods. LightGBM processed missing data using the Row-by-Row approach with favorable results, while Cosine Similarity was employed to detect and eliminate duplicate data. The use of TF-IDF and Label Encoding improved processing and prediction capabilities, reducing errors in the input data.

2.1.2. Prediction Model

K-Nearest Neighbors (KNN) was a primary model used in the toolkit recommendation system. Its ability to effectively identify similar projects made it well-suited for smaller datasets, allowing for quick and straightforward predictions. KNN ranked toolkits based on

proximity in the feature space, which enabled it to capture relevant similarities between new and existing projects.

However, the model faced challenges as dataset size increased. KNN's computational efficiency declined with larger datasets, leading to longer prediction times and decreased accuracy. This indicated that while KNN is effective for small-scale applications, it may not scale well for more complex or extensive data scenarios.

Overall, KNN served as a valuable tool in the recommendation process, especially in initial tests, but highlighted the necessity for more robust models to handle larger datasets effectively.

2.1.3. Evaluation Methods

The analysis of evaluation metrics such as RMSE and F1 Score indicates that the Row-by-Row method with LightGBM achieved the best results for handling missing data and predicting suitable toolkits. A lower RMSE reflects better prediction accuracy, while a higher F1 Score signifies improved classification performance. These metrics provide a clear understanding of model effectiveness and help in identifying the best model for our application.

In summary, the data processing methods have effectively improved data quality and prediction accuracy. The combination of LightGBM and K-Nearest Neighbors has established a robust foundation for the recommendation system, validated by evaluation metrics that highlight the need for ongoing refinement and enhancement.

2.2. Recommendation Platform

The Recommendation Platform serves as a crucial enhancement to the Toolkit Data Warehouse, offering users an improved experience in finding suitable toolkits. This section compares the new system with the old one and highlights its usability features, showcasing how these advancements facilitate better decision-making for users.

2.2.1. Comparison with old system

Compared to the old system, the new system enables predicting suitable toolkits based on past projects and predictive information. The old system only provided a list of toolkits without ranking them by relevance. The new system not only generates a list but also ranks toolkits based on their similarity to the new project, thereby assisting users in selecting the most effective toolkit.

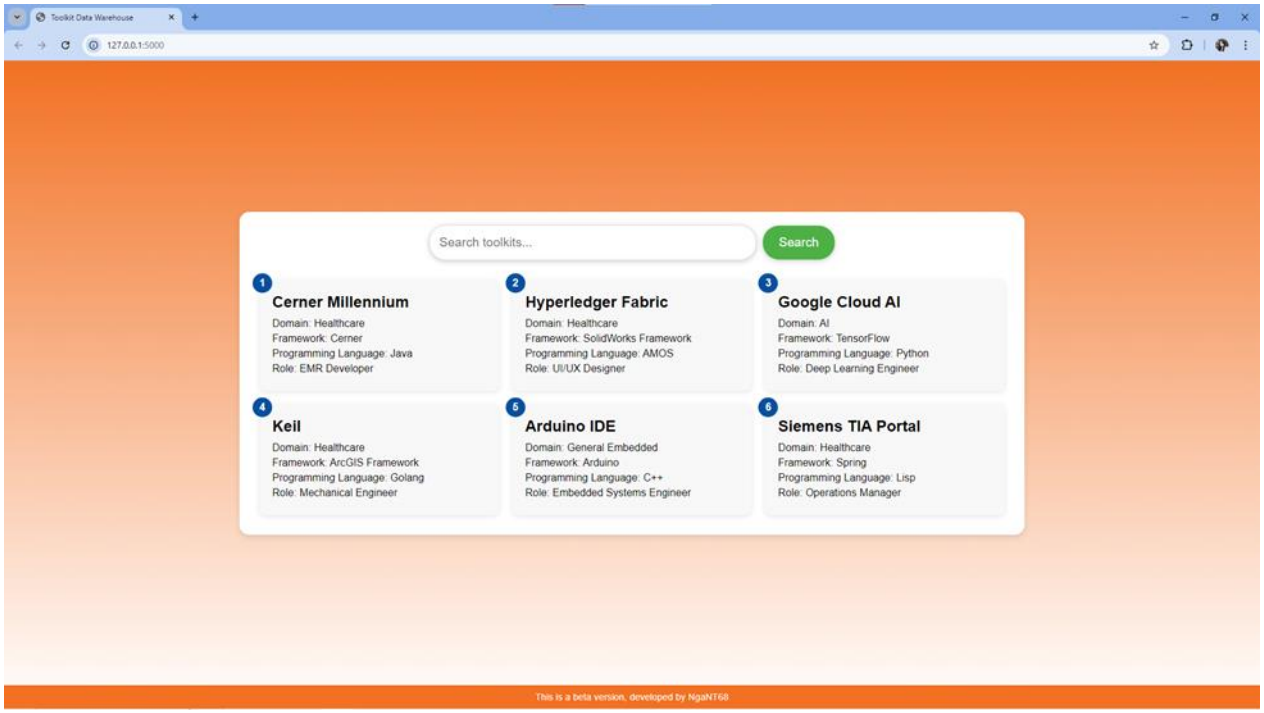

Figure 14: List of Recommendation Toolkits

2.2.2. Website Usability

The website was designed to be as simple and effective as possible. Users can easily input project descriptions and receive recommended toolkits. By separating the detailed view from the main search results, the system provides both a broad overview and in-depth information when required. The platform's color scheme and layout contribute to a professional and intuitive design that aids navigation.

In conclusion, the Recommendation Platform significantly outperforms the old system by not only providing a ranked list of toolkits based on project similarity but also ensuring a user-friendly interface. These improvements empower users to make informed choices efficiently, enhancing the overall functionality and effectiveness of the Toolkit Data Warehouse.

2.3. Impact, limitations and improvements

This section evaluates the impact of the Toolkit Data Warehouse on toolkit selection, highlighting its strengths and contributions. It also discusses the limitations encountered and identifies opportunities for future improvements, ensuring the platform remains responsive to user needs.

2.3.1. Limitations and Improvements

Current limitations include LightGBM's tendency to overfit and the unstable performance of Neural Networks. Future improvements could involve experimenting with additional models like XGBoost or Random Forest and applying better parameter tuning techniques. Additionally, enhancing data collection and cleaning processes to minimize errors and gaps is crucial for further development.

2.3.2. Business Impact

The implementation of the new system is expected to positively impact the business by optimizing the process of selecting toolkits for new projects. The new system not only reduces the time required to find and select tools but also enhances the accuracy of toolkit predictions, leading to increased efficiency and reduced costs for projects.

In summary, the results from the Toolkit Data Warehouse demonstrate significant advancements in predicting suitable toolkits for new projects. The combination of robust data processing methods and effective prediction models has enhanced the accuracy and relevance of recommendations, providing users with valuable insights for their project

needs. However, challenges remain, particularly regarding the scalability of models and the handling of complex data scenarios.

Addressing these challenges will be crucial for the platform's continued success and user satisfaction. Future improvements may include optimizing algorithms for larger datasets and refining the user interface to enhance usability. By focusing on these areas, the Toolkit Data Warehouse can further solidify its position as a vital resource for project toolkit selection, ultimately improving the overall experience for its users.

IV/ CONCLUSION

1. Summary

This thesis has explored the development of a comprehensive toolkit recommendation system that leverages machine learning and natural language processing (NLP) techniques to predict and suggest the most suitable toolkits for new projects. The system integrates various models such as LightGBM, KNN, and Cosine Similarity to intelligently identify relationships between project features and previously used toolkits, resulting in an enhanced recommendation engine that empowers users to make informed decisions.

The Toolkit Data Warehouse was developed as the platform to house this recommendation engine, combining a user-friendly interface with advanced backend architecture. The website, built with Flask and backed by PostgreSQL for data storage, allows users to search and filter through toolkits while receiving accurate, ranked recommendations. The system's data pipeline is managed using Apache Airflow, ensuring that data is efficiently processed before being fed into the predictive models.

The research focused on key challenges such as data processing, model evaluation, and recommendation accuracy. Performance metrics, including F1 scores and RMSE, were used to validate the system’s effectiveness in real-world conditions. The thesis also includes a detailed comparison of the old and new systems, showcasing significant improvements in the recommendation process, usability, and accuracy.

2. Future works

While the current system provides a solid foundation for recommending toolkits, there are several areas for future development:

- **Expanding Dataset Coverage:**

Future work could focus on enriching the dataset by including a wider range of project types, domains, and toolkits. This would allow the system to become more versatile and cover a broader spectrum of use cases.

- **Enhancing NLP Techniques:**

Further refinement in Natural Language Processing, such as using more advanced models like transformers, could improve the accuracy of project description parsing and feature extraction. This would lead to better identification of relevant toolkits.

- **User Feedback Integration:**

Incorporating a feedback loop where users can rate and comment on the suggested toolkits would improve the recommendation accuracy. This feedback could be used to train the system and continuously adapt the recommendations to user preferences.

- **Real-Time Data Processing:**

Transitioning to a real-time recommendation system by integrating streaming data pipelines could be a significant improvement. This would allow the system to offer up-to-date recommendations as new projects and toolkits are added to the database.

- **Model Optimization:**

Continued optimization of machine learning models and experimentation with other algorithms like deep learning could further enhance the system's predictive capabilities, providing more accurate and personalized recommendations.

In conclusion, the work conducted in this thesis presents a significant step towards building a more intelligent, responsive, and user-friendly toolkit recommendation platform. While there is still room for improvement, the foundation laid here offers strong potential for future enhancements and real-world application.

REFERENCES

[1] K. G.-O. Stef van Buuren, "Multivariate Imputation by Chained Equations (MICE) in R," 2011.

[2] M. W. Alsuhaibani, "Handling Imbalanced Data and Missing Values in Machine Learning: A Comparison of Approaches," 2020.

[3] E. S. Christian Rendl, "Multi-output Prediction and its Applications," 2020.

[4] E. S. Christian Rendl, "Simultaneous Prediction of Multiple Targets Using Multivariate Decision Trees," 2020.

[5] M. A. I. K. Ismail Ahmedy, "Missing Data Imputation Using Deep Learning," 2019.

[6] N. T. K. L. Kieu Tuan Anh, "Vietnamese Text Processing for NLP Applications," 2019.

[7] J. B. Nicolas Garrette, "Challenges in Multilingual NLP: A Case Study of Low-Resource Languages," 2020.

[8] J. S. Murray, "Multiple Imputation: A Review of Practical and Theoretical Findings," 2018.

[9] E. K. E. L. Steven Bird, NLP Techniques for Text Classification and Information Retrieval, O'Reilly Media, 2019.

APPENDIX

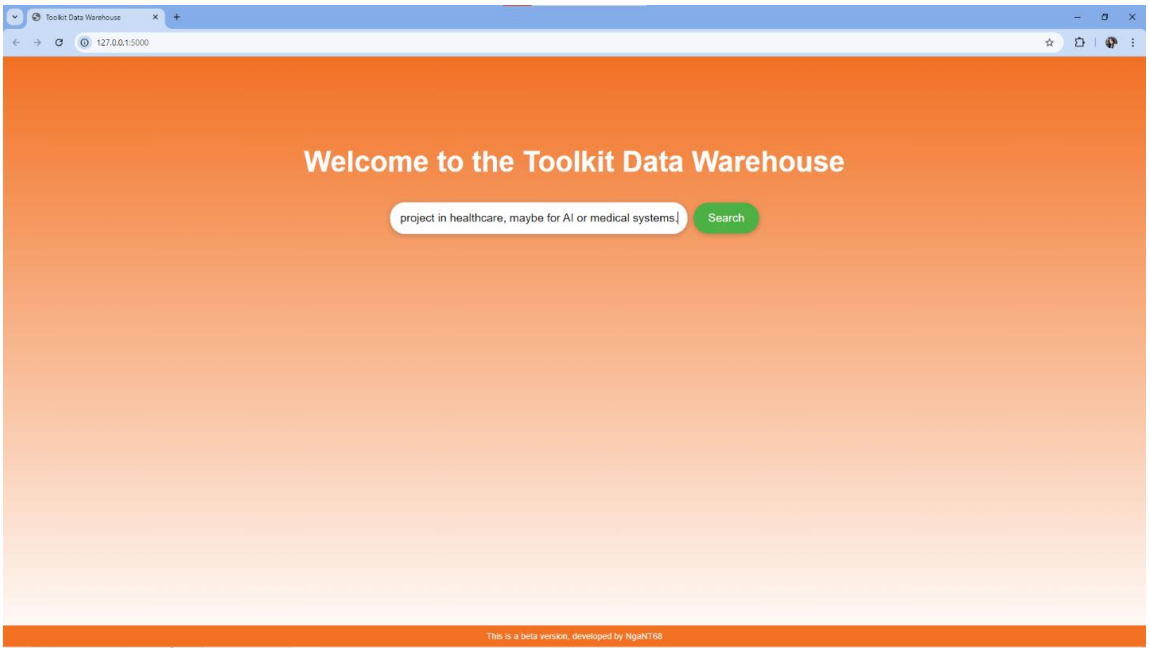

Figure 15: Enter a new project information

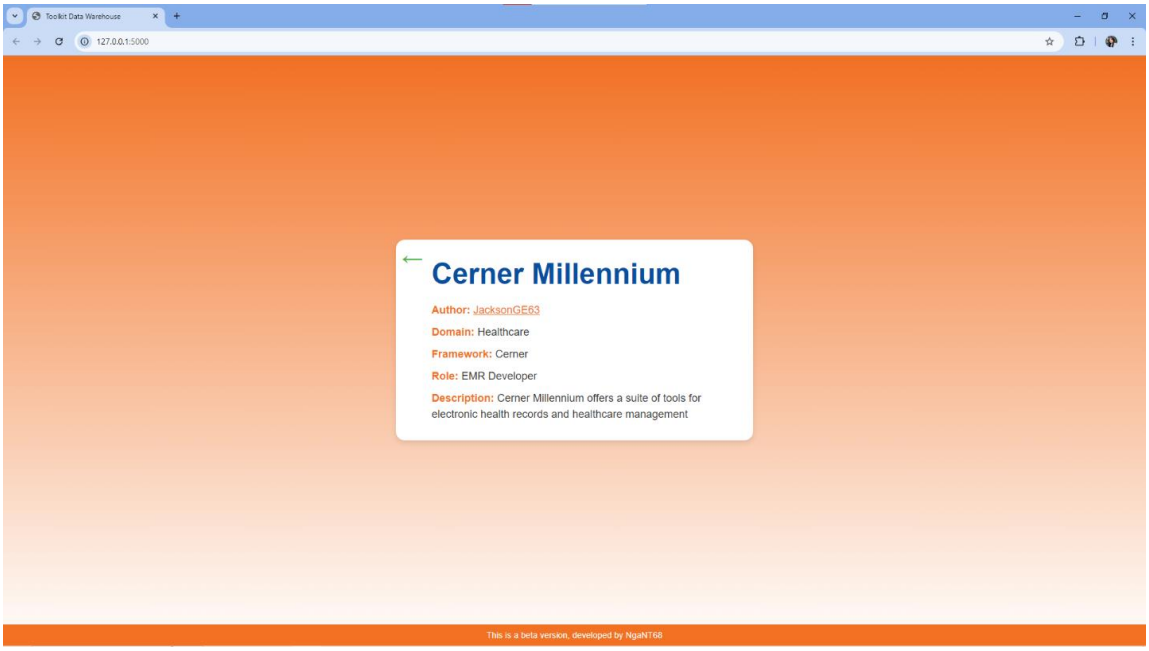

Figure 16: Toolkit detail information

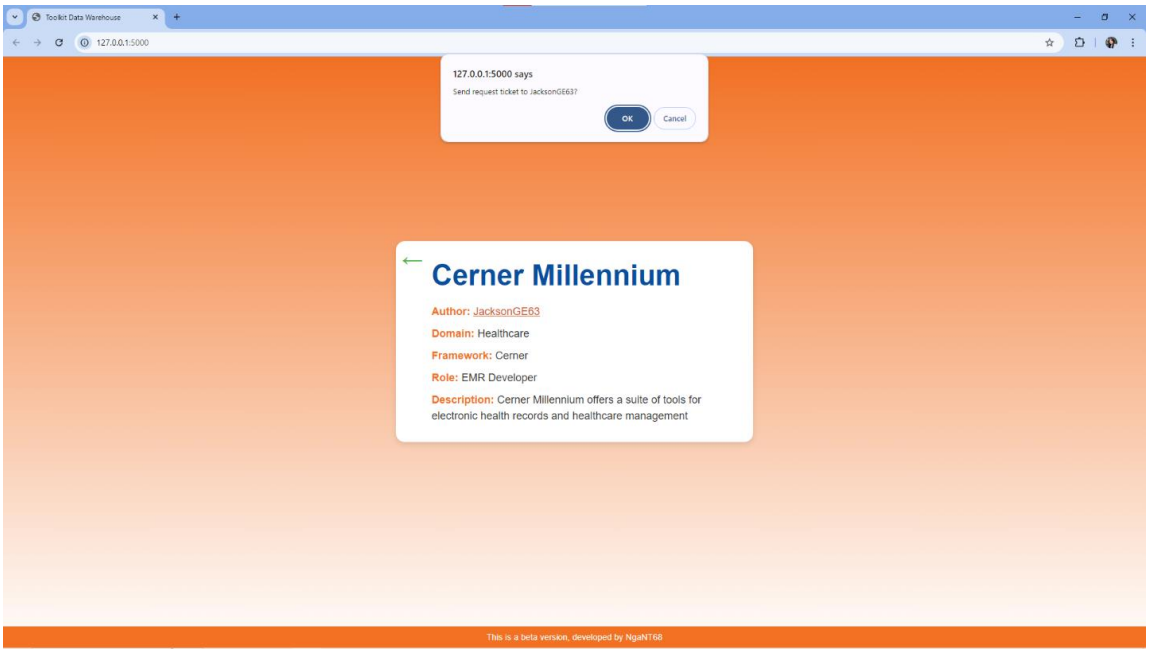

Figure 17: Send ticket action