**Final Project Report**

**NLP based Duplicate Bug Report Detection using Supervised Machine Learning Algorithms**



**Project Supervisor**

**Saad Ahmed**

**Submitted By**

**F22024F062**

**Muhammad Umer Mansoor BC190400681**

**Software Projects & Research Section,**

**Department of Computer Sciences,**

**Virtual University of Pakistan**

|  |
| --- |
|  |



**CERTIFICATE**

This is to certify thatMuhammad Umer Mansoor (BC190400681) has worked on and completed his Software Project at Software & Research Projects Section, Department of Computer Sciences, Virtual University of Pakistan in partial fulfillment of the requirement for the degree of BS in Computer Sciences under my guidance and supervision.

In our opinion, it is satisfactory and up to the mark and therefore fulfills the requirements of BS in Computer Sciences.

**Supervisor / Internal Examiner**

Saad Ahmed

Supervisor,

Software Projects & Research Section,

Department of Computer Sciences

Virtual University of Pakistan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**External Examiner/Subject Specialist**

<<External Supervisor Name>>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**Accepted By:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

(For office use)

**EXORDIUM**

**In the name of Allah, the Compassionate, the Merciful.**

**Praise be to Allah, Lord of Creation,**

**The Compassionate, the Merciful,**

**King of Judgment-day!**

**You alone we worship, and to You alone we pray for help,**

**Guide us to the straight path.**

**The path of those who You have favored,**

**Not of those who have incurred Your wrath,**

**Nor of those who have gone astray.**

**DEDICATION**

I dedicate this project first and foremost to my **loving parents** who have always encouraged me to follow my passions. Their unwavering love and support have been a source of strength through all my endeavors.

I also dedicate this work to **my** **supervisor** **Saad Ahmed** who sparked my interest in technology and instilled in me a curiosity to learn. His guidance and mentorship inspired me to strive higher.

**ACKNOWLEDGEMENT**

First of all, I am grateful to the **ALLAH Almighty** the most merciful and beneficent who guides us in darkness and helps us in difficulties.

Profound thanks to our supervisor **“Saad Ahmed”**, who helped throughout the completion of my project.

My humble gratitude to **my Parents and brothers** for their implacable love, prayers encouragement & best wishes, which helped me to face the difficulties for life & enabled me to complete my education. Due to their prayers, valuable guidance, suggestions, and encouragement I am able to complete this project in successful manners.

Finally, I appreciate the countless **researchers and developers** whose work formed the foundation for this project. Standing on the shoulders of giants enables us to push forward the boundaries of knowledge.

**PREFACE**

Pursuing this final year project has been an immensely enriching experience for me. It provided the opportunity to apply my academic knowledge to build an end-to-end machine learning system.

One of the most valuable lessons was the importance of defining the problem clearly. I spent a good amount of time researching duplicate bug detection, understanding pain points in the domain, and articulating project objectives. This solid foundation guided all subsequent efforts.

Preprocessing the textual data using NLP techniques was highly educational. It gave me insight into working with unstructured data and the power of NLP in extracting signals from text. Experimenting with different algorithms to build the classification model was also a key takeaway.

Finally, I would say, this entire final year project experience has levelled up my skills in this exciting and vast field of Data Science.

**TABLE OF CONTENTS**

|  |
| --- |
|  |

[CHAPTER NO. 1](#CH1)

[gathering & Analyzing info 09](#CH1)

* 1. [introduction ………………………………………. 10](#Intro11)
  2. [purpose ……………………………………………… 10](#pur12)
  3. [scope ……………………………………………....... 10](#sc13)
  4. [definitions, acronyms and abbreviations …… 11](#def14)
  5. [Project requirements ………………………….. 11](#pro15)
     1. [Functional Requirements …………...... 11](#func151)
     2. [Non-Functional Requirements ………. 12](#non152)
  6. [use cases and usage scenarios ..................... 13](#use16)
     1. [Use Case Diagram ..................................... 13](#use161)
     2. [Usage Scenarios ....................................... 14](#usage162)
  7. [Development Methodology ......................... 19](#dev17)
     1. [Chosen Methodology ............................. 19](#cho171)
     2. [Reasons for Chosen Methodology .............. 19](#rea172)
     3. [Work Plan (Gantt Chart) ............................. 20](#work173)

[CHAPTER NO. 2](#CH2)

[designing the project 22](#CH2)

* 1. [Introduction ................................................ 23](#intro21)
  2. [purpose ........................................................ 23](#purp22)
  3. [scope ........................................................... 23](#sco23)
  4. [definitions, acronyms and abbreviations ..... 24](#def24)
  5. [Methodology .............................................. 25](#meth25)

[Chapter no.3](#CH3)

[DEvelopment 28](#CH3)

* 1. [Development plan ................................. 29](#dev31)
  2. [Programming Languages ....................... 29](#pro32)
  3. [Machine Learning Models ..................... 30](#mach33)
  4. [Model Evaluation Metrics .................... 31](#mod34)
  5. [Analysis Of Model Evaluation Results . 33](#ana35)

**CHAPTER 1**

Gathering & Analyzing Info

* 1. **INTRODUCTION**

The objective of this project is to classify duplicate bug reports using natural language processing (NLP) techniques and supervised machine learning algorithms. Duplicate bug reports refer to multiple issue tickets created for the same bug in a software project. They waste developer time and resources in triaging and debugging the same issue multiple times.

* 1. **PURPOSE**

The purpose of this project is to build an NLP classification machine learning algorithm that can automatically identify duplicate bug reports. This will help improve the efficiency of the bug fixing process.

* 1. **SCOPE**

The scope of this project includes:

* Setting up the Python environment with libraries like Pandas, NumPy, Matplotlib, Seaborn, Scikit-Learn, NLTK and spaCy.
* Loading the bug report dataset.
* Exploratory data analysis on the dataset.
* Preprocessing the textual data in the bug reports using NLP techniques like tokenization, stopword removal, stemming and lemmatization.
* Building a corpus from the processed text.
* Splitting data into training and test sets.
* Training machine learning models like Naive Bayes, SVM, Random Forest on the corpus.
* Evaluating model performance using metrics like accuracy, precision, recall.
* Comparing results of different models.
* Retraining models to improve accuracy if needed.
  1. **DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

NLP - Natural Language Processing

SVM - Support Vector Machine

TF-IDF - Term Frequency Inverse Document Frequency

BOW - Bag of Words

* 1. **PROJECT REQUIREMENTS**
     1. **FUNCTIONAL REQUIREMENTS**

The following are the functional requirements of the project separated by system and user requirements:

1. System must be set the environment online/offline (If required)
2. System applies different data processing techniques (Tokenization, Stop word removal, Lemmatization, etc.)
3. System must Build Corpus
4. System must be split the given dataset into testing and training.
5. System must train the specified model.
6. User must be evaluating mentioned models in the form of Confusion Matrix, Accuracy, Precision, Recall
7. User must be discussed the results of given algorithms (Naïve Bayes, Support Vector Machine, Random Forest)
8. User must retrain the model if accuracy is not good (less than 60%) by changing different training parameters (If required)
   * 1. **NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements for this project/system may include the following:

**Performance:** The system must be able to classify a bug report within a specific time frame, for example, within a few seconds, to ensure that the bug triaging process is not delayed.

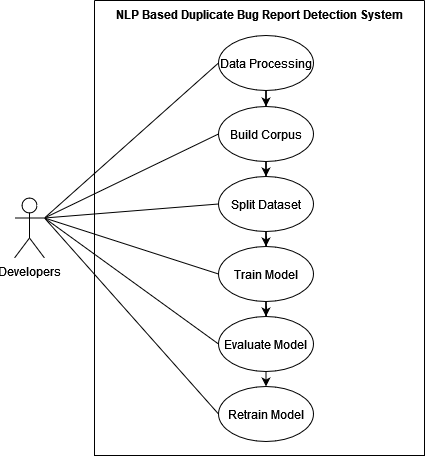
**Scalability:** The system must be able to handle an increasing number of bug reports without a significant decrease in performance.

**Reliability:** The system must have a high level of reliability, with minimal downtime, to ensure that bug reports are classified and assigned to the appropriate developer in a timely manner.

**Usability:** The system must be user-friendly, with an intuitive interface, to ensure that developers and customer support representatives are able to use it effectively.

* 1. **USE CASES AND USAGE SCENARIOS**
     1. **USE CASE DIAGRAM (OPTIONAL):**

The use case diagram is as follows:



* + 1. **USAGE SCENARIOS (OPTIONAL):**

Here are the usage scenarios of the use cases for this project:

**Usage Scenario#1:**

|  |  |
| --- | --- |
| Use Case Title | Data Processing |
| Use Case ID | UC-01 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to apply data processing techniques. 3. The user chooses the techniques to apply such as Tokenization, Stop word removal, Lemmatization, etc. 4. The system applies the selected techniques on the data. |
| Description | The user selects the NLP techniques to be applied on the bug report data. These techniques help in preparing the data for further processing, such as building a corpus and training machine learning models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The system environment must be configured. |
| Post-Conditions: The data has been processed using the selected NLP techniques. |
| Author(s) | User |
| Exceptions | The system may not work correctly if there is an error in the implementation of the techniques. |

**Usage Scenario#2:**

|  |  |
| --- | --- |
| Use Case Title | Build Corpus |
| Use Case ID | UC-02 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to build the corpus. 3. The system builds the corpus using the processed data. |
| Description | The system builds a corpus using the processed data to train machine learning models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The data must be processed using NLP techniques. |
| Post-Conditions: The corpus has been built and is ready to use for training machine learning models. |
| Author(s) | User |
| Exceptions | The system may not work correctly if there is an error in the implementation of building the corpus or if the data is not in the correct format. |

**Usage Scenario#3:**

|  |  |
| --- | --- |
| Use Case Title | Data Splitting |
| Use Case ID | UC-03 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to split the data. 3. The user inputs the percentage of data to be used for testing. 4. The system splits the data into training and testing sets. |
| Description | The user splits the data into training and testing sets. The training data is used to train the machine learning models, while the testing data is used to evaluate the performance of the models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The corpus must be built. |
| Post-Conditions: The data has been split into training and testing sets. |
| Author(s) | User |
| Exceptions | The system may not work correctly if the percentage of data for testing is set to a value that is too low or too high. |

**Usage Scenario#4:**

|  |  |
| --- | --- |
| Use Case Title | Model Training |
| Use Case ID | UC-04 |
| Action(s) | 1. User selects the specified model to be trained. 2. User sets the training parameters. 3. System starts the training process. 4. System saves the trained model. |
| Description | After splitting the dataset into testing and training sets, the user selects the model to be trained from the available options. The user sets the training parameters and initiates the training process. The system uses the training data to train the selected model and save the trained model for future use. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The dataset must be split into training and testing sets. |
| Post-Conditions: The system must save the trained model. |
| Author(s) | User |
| Exceptions | None |

**Usage Scenario#5:**

|  |  |
| --- | --- |
| Use Case Title | Model Evaluation |
| Use Case ID | UC-05 |
| Action(s) | 1. The user selects the algorithm they want to evaluate and compare the results for. 2. The system presents the evaluation results for the selected algorithm, including confusion matrix, accuracy, precision, and recall. 3. The user discusses the results and compares them with the results of other algorithms. 4. The user decides on the algorithm to use for the bug report classification. |
| Description | This use case represents the step where the user evaluates the performance of different algorithms and compares their results to make a decision on which algorithm to use for the classification of duplicate bug reports. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The system must have the evaluation results for the algorithms. The user must have knowledge of the performance metrics. |
| Post-Conditions: The results of the evaluation are documented for future reference. |
| Author(s) | User |
| Exceptions | None |

**Usage Scenario#6:**

|  |  |
| --- | --- |
| Use Case Title | Retrain Model |
| Use Case ID | UC-06 |
| Action(s) | 1. The user checks the accuracy of the current model and finds it to be less than 60%. 2. The user selects the option to retrain the model. 3. The system suggests different training parameters to adjust. 4. The user selects the training parameters they want to adjust. 5. The system re-trains the model with the updated training parameters. 6. The system evaluates the new model. 7. The user checks the accuracy of the new model and finds it to be satisfactory. |
| Description | This use case represents the step where the user re-trains the model to improve accuracy if the accuracy of the current model is less than 60%. |
| Alternative Paths | 1. If the user does not want to retrain the model, they can choose to use the existing model with less accuracy. 2. If the adjusted training parameters do not improve accuracy, the user can adjust other parameters or try a different algorithm. |
| Pre and Post Conditions | Pre-Conditions: The user must check the accuracy of the current model and find it to be less than 60%. |
| Post-Conditions: The system has a new model with improved accuracy. The user can use the new model for bug report classification. |
| Author(s) | User |
| Exceptions | If the accuracy of the model is greater than or equal to 60%, no retraining is required, and the system will continue with the next step(s). |

* 1. **DEVELOPMENT METHODOLOGY**
     1. **CHOSEN METHODOLOGY**

I’ve chosen the VU Process Model which is a combination of the Waterfall and Spiral models. It can be used to manage the development of this project. This methodology is chosen because it allows for the integration of the best practices of both models, resulting in a more efficient and effective development process.

* + 1. **REASONS FOR CHOSEN METHODOLOGY**

Reasons for choosing VU Process Model methodology are as follows:

**Allows for detailed requirements gathering:**

The Waterfall model aspect provides a structured way to carry out detailed requirements gathering and analysis in the initial stages. This is critical for this NLP project to clearly understand the problem scope, data needs, and algorithms to be used.

**Supports an iterative development approach:**

The Spiral model aspect allows for iterative development, testing and prototyping of solutions. This supports experimentation with different NLP techniques and machine learning algorithms in an incremental way.

**Reduces risk through incremental development:**

The iterative approach helps divide the project into smaller chunks and gather regular feedback. This reduces risks and catches issues early.

**Accommodates changes:**

The Spiral model aspect accommodates changes to requirements or design based on feedback from testing phases. This is beneficial due to the exploratory nature of comparing NLP and ML approaches.

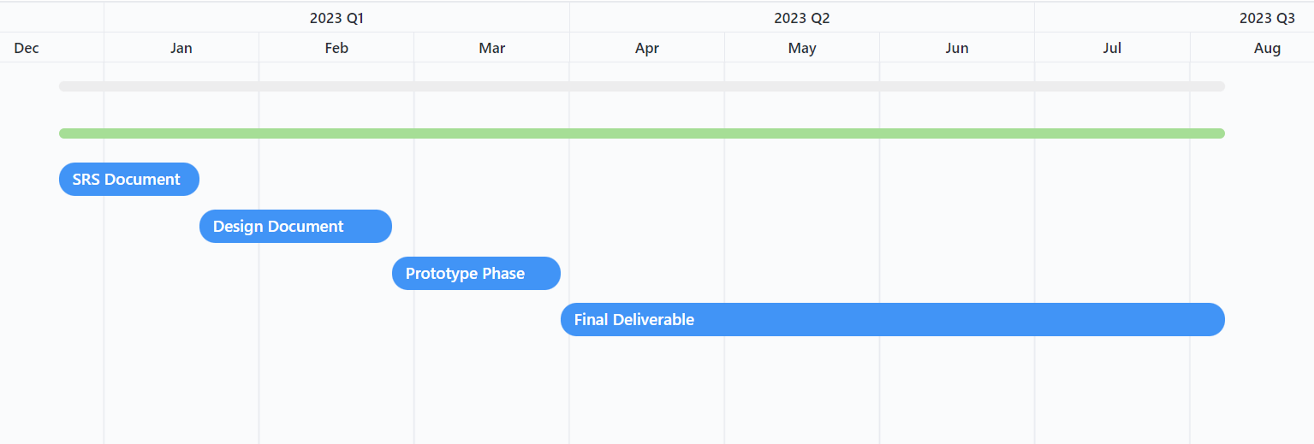
**Enables continuous improvement:**

The cyclic approach ensures lessons learned in one cycle can be applied to the next to continuously refine and improve the solution. This allows the model accuracy to be progressively enhanced.

**Facilitates better time and cost estimates:**

The defined phases provide clear milestones for time and resource estimation. This improves predictability, especially for a research-oriented project.

* + 1. **Work Plan (Gantt Chart)**



The Gantt chart shows the schedule for completing the NLP duplicate bug detection project over different stages from December 2022 to August 2023.

In the first stage from 23 December 2022 to 19 January 2023, the focus is on requirements gathering and analysis to produce the SRS document. This involves defining the project scope, objectives, functional requirements, use cases etc.

The second stage is creation of the design document from 20 January 2023 to 24 February 2023. The key tasks here are coming up with the system architecture, finalizing the modeling techniques, algorithms, framework tools to be used.

The next phase is prototype development from 25 February 2023 to 29 March 2023. The main goals are implementing the pipeline designed in the previous stage, developing models, and conducting initial testing.

The final deliverables are produced from 30 March 2023 to 7 August 2023. This involves activities like:

* Refining the models and optimizing performance.
* Integrating the solution with the bug tracking system.
* Comparing model results.
* Retraining models if required.
* Documenting the project in the final report.
* Creating a presentation with project outcomes.

In summary, the Gantt chart outlines a structured schedule for incrementally advancing through the planning, design, development, and delivery stages of the NLP project over an 8-month timeframe. The key milestones are the SRS, design, prototype, and final report/presentation.

**CHAPTER 2**

Designing the Project

* 1. **INTRODUCTION**

The design document outlines the system architecture, models, algorithms, and key technical design decisions made for this project. It serves as a blueprint for the development process.

* 1. **PURPOSE**

The purpose of this design document is to provide outlines of the system's design, features, and functionalities, and will serve as a guide for me to build and implement the system efficiently.

The design document will also benefit me by enabling me to identify potential design flaws or issues early on in the development process, ensuring the system's final version meets the requirements.

The system will operate by processing textual data through different NLP techniques such as tokenization, stop-word removal, and lemmatization, and then classifying the bug reports using machine learning algorithms.

* 1. **SCOPE**

The scope of this project encompasses building an end-to-end machine learning system for duplicate bug report classification using natural language processing and supervised learning algorithms.

The stages involved include preprocessing the textual data in the bug reports, extracting informative features, training different classification models, evaluating model performance, and determining the optimal approach.

Specifically, the project scope covers tasks such as loading the dataset of bug reports, performing exploratory analysis to understand data characteristics, cleaning the text data using NLP techniques like tokenization, stopword removal, lemmatization.

Building a corpus from the processed text and splitting into train-test sets are also included. Vectorizing the corpus using BOW and TF-IDF techniques.

Using the vectorized corpus, different supervised learning models like NB, SVM, RF, DT, KNN etc. are trained and evaluated on metrics like confusion matrix, accuracy, precision, recall, and F1-score.

The results are analyzed to find the best performing model. If the accuracy is unsatisfactory initially, the models are retrained with different parameters to improve performance.

The end goal is developing an NLP system that can effectively distinguish between new and duplicate bug reports. This has the potential to appreciably improve efficiency in software bug triaging and resolution.

The techniques explored can also provide insights into applying NLP and ML to textual data analysis in other domains.

* 1. **DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

NLP – Natural Language Processing

ML – Machine Learning

NB – Naïve Bayes

SVM – Support Vector Machine

RF – Random Forest

DT – Decision Tree

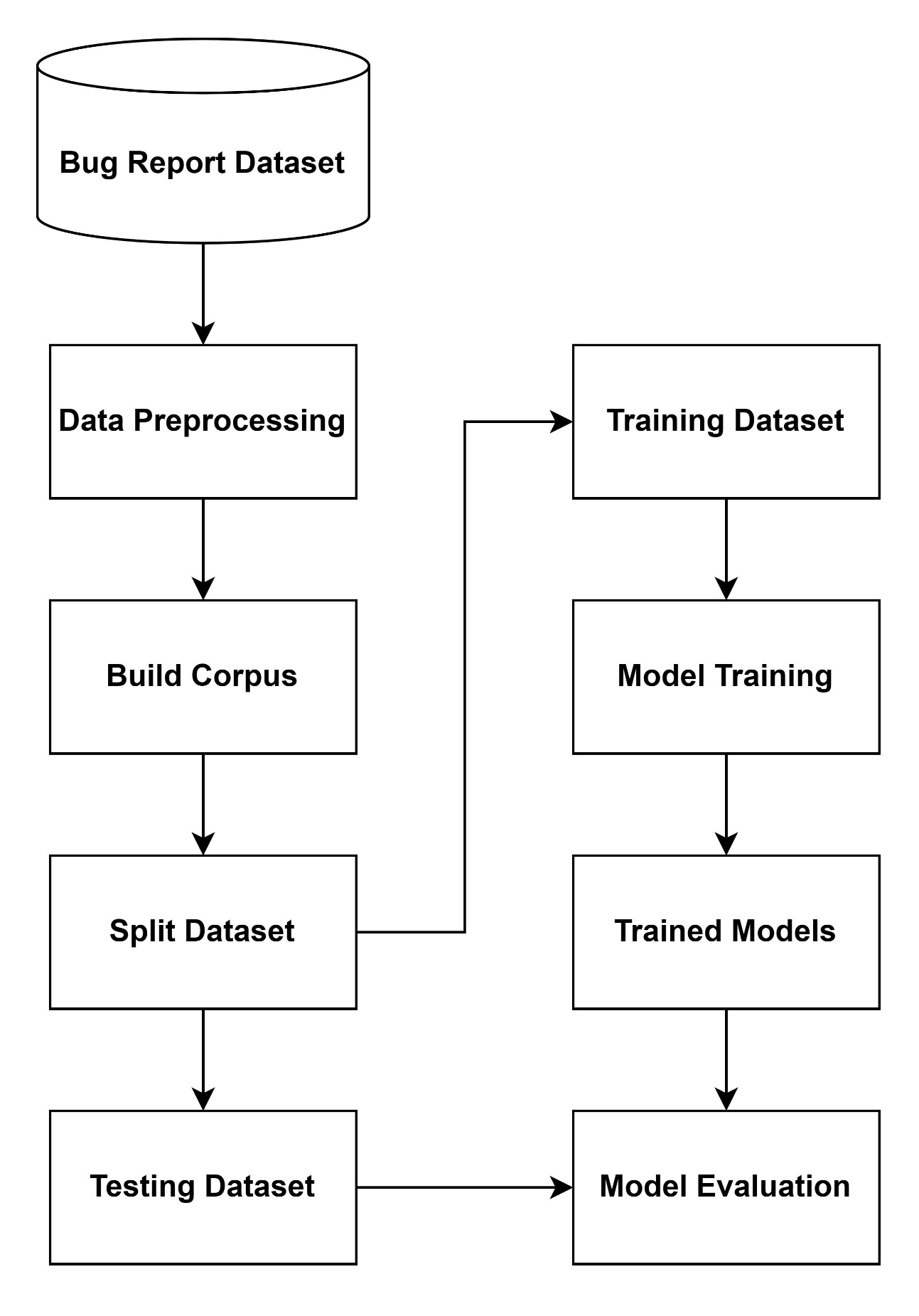
KNN – K-Nearest Neighbors

BOW – Bag of Words

TF-IDF – Term Frequency Inverse Document Frequency

* 1. **METHODOLOGY**

**Diagram:**

****

**Details:**

The methodology for developing an NLP-based duplicate bug report detection system using supervised machine learning involves several key stages.

First, a dataset of bug reports is collected from relevant sources to be used for training and testing models. This data then undergoes preprocessing including tokenization, removal of stopwords, stemming, and lemmatization in order to clean the text and prepare it for splitting into train, test datasets and feature extraction.

Next, the dataset is splitted into train and test datasets for training and testing ML models. Vectorization is then to applied to both training and testing datasets. The vectorizer is fitted only on the training data. This learns the vocabulary and feature mapping.

Then training data is transformed into vector representations using the fitted vectorizer. But test data isn’t touched when fitting the vectorizer. To vectorize the test data, we use the same fitted vectorizer from training dataset.

Useful features like TF-IDF vectors, word/phrase frequencies, and bag-of-words representations are then engineered from the preprocessed and vectorized text.

The next step is selecting appropriate supervised machine learning algorithms to train such as NB, SVM, RF, DT, KNN etc. The models are trained on the preprocessed and vectorized bug report data in order to tune them for optimal duplicate detection performance.

Once trained, the models are evaluated on a held-out test set using metrics like accuracy, precision, recall and F1-score. This reveals how well each model is able to identify duplicate bug reports compared to ground truth labels.

Finally, the top performing model is integrated into a live bug tracking system to flag incoming bug reports as potential duplicates in real-time.

In summary, this structured NLP and machine learning methodology allows effective duplicate detection through careful data preprocessing, feature engineering, model training, and evaluation.

**CHAPTER 3**

Development

* 1. **DEVELOPMENT PLAN**

The project was developed incrementally in the following phases:

**Environment Setup:** The Python environment was set up with libraries like Pandas, NumPy, Scikit-learn, NLTK. Jupyter notebook was used for development.

**Data Loading and EDA:** The bug report dataset was loaded and analyzed to understand the data distribution, correlations, etc.

**Data Preprocessing:** NLP techniques like tokenization, lemmatization, stopword removal were applied to clean and standardize the text.

**Dataset Splitting:** The dataset was splitted into the training and testing set. The total of 80% random data was splitted into the training set and rest of 20% was splitted to the testing dataset.

**Feature Extraction:** Bag of Words and TF-IDF models were used to vectorize the textual data.

**Model Building:** Classification models like Naive Bayes, SVM, Random Forest were trained on the extracted feature vectors.

**Model Evaluation:** The models were evaluated using metrics like accuracy, precision, recall.

* 1. **PROGRAMMING LANGUAGES**

The project was developed in Python 3.10.12. The key libraries used were:

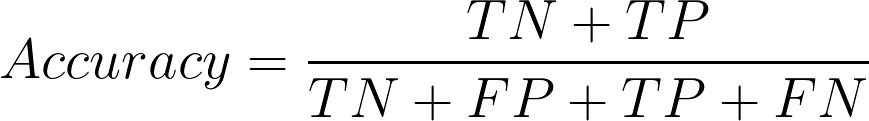
* **Pandas, NumPy** - For data manipulation.
* **Scikit-learn** - For machine learning models.
* **NLTK, spaCy** - For text processing and NLP.
* **Matplotlib, Seaborn** - For visualization and EDA.
  1. **MACHINE LEARNING MODELS**

The following machine learning models were developed and evaluated in this project:

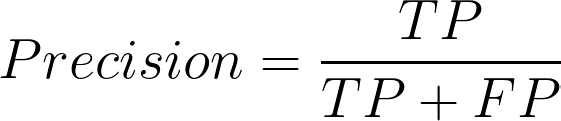
1. **Naive Bayes Classifier - Multinomial Naive Bayes:** A probabilistic classifier that applies Bayes' theorem to model feature distributions and make predictions. Works well with high-dimensional text data.
2. **Support Vector Machine (SVM) -** Two implementations were evaluated:
   1. **SVC:** Non-linear SVM using kernel trick to project data into higher dimensions to find optimal separating hyperplane. Effective for complex datasets.
   2. **LinearSVC:** Linear SVM that finds a maximum margin hyperplane in original feature space. Faster performance for large datasets.
3. **Random Forest Classifier:** Ensemble method that aggregates predictions from multiple decision trees. Helps reduce overfitting on training data.
4. **Logistic Regression:** Regression model that uses logistic function to predict binary class probabilities. Simple yet powerful baseline classifier.
5. **K-Nearest Neighbors (KNN):** Non-parametric classifier that predicts labels based on distance similarity with K datapoints in training set.
6. **Decision Tree Classifier:** Non-parametric model that uses rule-based decisions on feature values to classify data.
   1. **MODEL EVALUATION METRICS**

The models were evaluated using the following key metrics:

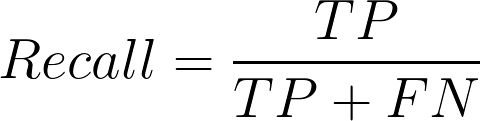
**Accuracy:** Percentage of correctly classified bug reports. It tells us out of all the predictions we made, how many were true.



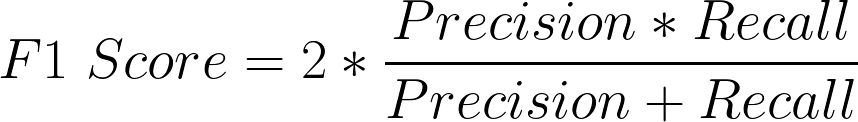
**Precision:** Percentage of duplicate predictions that were actual duplicates. It is a metric that gives us the proportion of true positives to the number of total positives that model predicts.



**Recall:** Percentage of actual duplicates correctly predicted. Recall focuses on how good the model is at finding all the positives.



**F1-score:** F1 score is the harmonic mean of precision and recall and is a better measure than accuracy. It is a measure that combines recall and precision.



**Confusion Matrix:** Summary of correct and incorrect predictions for each class. It is a performance measurement for machine learning classification problems where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



**What is TP, FN, TN, and FP? Let’s learn by the examples in context of this duplicate bug detection project!**

**TP - True Positive:** A true positive occurs when the model correctly predicts a bug report as a duplicate. For example, if there are two reports describing the same software crash, and the model labels the second report as a duplicate, that is considered a true positive classification.

**FN - False Negative:** A false negative occurs when the model incorrectly predicts a duplicate bug report as non-duplicate. For instance, if two bug reports describe the same UI layout issue, but the model fails to identify the second report as a duplicate, it is a false negative error.

**TN - True Negative:** A true negative happens when the model correctly predicts a new bug report as non-duplicate. For example, if a bug report describes a software crash that has not been reported before, and the model labels it as a new issue, this is a true negative classification.

**FP - False Positive:** A false positive arises when the model wrongly predicts a non-duplicate bug as a duplicate. If a bug report describes a new printing failure, but the model mistakenly classifies it as a duplicate, that is considered a false positive result.

* 1. **ANALYSIS OF MODEL EVALUATION RESULTS**

Based on the evaluation results, the Random Forest classifier performs the best among all the models, with the highest accuracy and F1 scores. Here is a more detailed analysis:

Random Forest achieves the highest accuracy of 75% on stemmed data with BOW, compared to 60-72% for other models. It also has strong macro and weighted F1 scores of 0.75.

On lemmatized data with BOW, Random Forest has 74% accuracy, again outperforming other models in the 60-72% range. The precision, recall and F1 scores are balanced for both classes.

With TF-IDF vectorization, Random Forest still performs very well with 73-74% accuracy on stemmed and lemmatized data, better than the other models. The precision and recall are reasonably good for both classes.

In contrast, models like Naive Bayes, SVM, Logistic Regression and KNN achieve 60-72% accuracy consistently across vectorization methods, lower than Random Forest. Their F1 scores are also not as strong except for SVM and Logistic Regression.

SVM is the closest competitor to Random Forest with 68-72% accuracy. But Random Forest leverages an ensemble of trees to reduce overfitting and improve results compared to a single tree.

Naive Bayes consistently scored around 66% accuracy across all configurations. The TF-IDF vectors improved the recall for the duplicate class compared to BOW. But overall, NB was outperformed by SVM and Random Forest.

Logistic Regression and Decision Tree were on par with each other, with 66-67% accuracy. Nothing stood out as their strengths or weaknesses.

KNN was the weakest model, with accuracy of 60-63%. It had low recall, meaning it failed to detect many of the duplicate bugs.

In terms of stemming vs lemmatization, the results are quite close for Random Forest on both versions of the dataset. The stemmed version has slightly higher accuracy in some cases. But both achieve good results overall.

For the other models like Naive Bayes, SVM, Logistic Regression etc., their accuracy is also very close on both stemmed and lemmatized data. There is no significant difference.

Concluding this, I would say that lemmatization does not really improve results over stemming for this dataset. Both versions give comparable model performance. This suggests the context needed for lemmatization does not provide much benefit here.

**My Final Remarks:**

Random Forest would be my choice as the best model for this dataset based on its high accuracy, good F1 score, balance between precision and recall, and robust performance across vectorization methods. I think that some hyperparameter tuning may further improve its effectiveness.