Selecting Software Development Life Cycle (SDLC) Models from Requirement Documents Using NLP

By

Mahira Gul

00000329238

Supervisor

Asst Prof Dr. Yawar Abbas Bangash

A thesis submitted to the Department of Computer Software Engineering, Military College of Signals (MCS), National University of Sciences and Technology (NUST), Islamabad, Pakistan, in partial fulfilment of the requirements for the degree of MS in Software Engineering.

February 2023

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS thesis written by Ms **Mahira Gul**, Registration No. **00000329238** of Military College of Signals has been vetted by undersigned, found complete in all respect as per NUST Statues/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: _____

Name of Supervisor: **Asst Prof Dr. Yawar Abbas Bangash**

Date: _____

Signature (HOD): _____

Date: _____

Signature (Dean): _____

Date: _____

Declaration

I, Mahira Gul declare that this thesis titled **"Selecting Software Development Life Cycle (SDLC) Models from Requirement Documents Using NLP"** and the work presented in it is my own and has been generated by me as a result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a Master of Software Engineering degree in MCS NUST.

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other qualification at NUST or any other institution this has been clearly stated.

3. Where I have consulted the published work of others, this is always clearly attributed.

4. Where I have quoted from the published work of others the source is always given. With the exception of such quotations this thesis is entirely my own work.

5. I have acknowledged all main sources of help.

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

_____

Mahira Gul

00000329238

MSSE27

Dedication

“Starting with the name of Allah, who is the most Beneficent and the most Merciful.”

I dedicate this thesis to my parents, supervisor, and co-supervisor, who supported and guided me in each step of this journey of thesis.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Yawar Abbas Bangash for his patience, motivation, and immense knowledge. His guidance carried me through all the stages of research and writing this thesis.

My sincere thanks also go to my co-supervisor Lt. Col. Khawir and committee member Dr. Hammad Afzal who had their continuous support throughout my research and thesis. Without the guidance of these gentlemen, I would not have been able to conduct this research.

Last, but not least, I would like to thank my parents and siblings who helped me maintain the motivation to complete this thesis.

Abstract

The success of software system depends on many factors among which the selection of most suitable Software development life cycle (SDLC) model is the most significant. SDLC represents a framework to develop a software system through planning, analysis, design, implementation, testing, deployment, and maintenance. These activities are carried out in different series of steps and depend on the context and characteristics of the software project [1]. In this research, we will provide a view of different SDLC models with their important factors that need to be considered for their selection. Then we will propose a system to analyze the software charter document to extract useful information using NLP techniques like regular expressions. At the end, the most suitable SDLC model will be recommended for software practitioners to carry out the development process by using machine learning algorithms. We have applied 11 machine learning algorithms and achieved the highest accuracy of 90.909% with Naïve Bayes Algorithm.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Overview

Software engineering is an approach to apply systematic procedures in design and implementation of a quality and reliable software product as per the need of the customers. The main constraints to consider for developing a software project are time, budget, and requirements. The project should be on time, within the specified budget and as per the requirements mentioned by the clients in project scope [45]. To manage a software project, a proper methodology is needed that helps manager to divide the task and to assign the roles. Before resource allocation, it is important to understand the project characteristics and to divide in into different phases. These phases may vary based on the type of the project. Software development life cycle (SDLC) model is a methodology that is followed by the project managers to plan and execute a project from start to end. The main phases of SDLC are requirement identification and analysis, software design, software implementation, testing, deployment, and maintenance. The model selected by project manager specifies the sequence of steps carried out in the project development. No one model is fit for all projects as it varies based on the conditions.

With the advancement in information technology, software applications have now become a part of everyday life. Millions of applications and software products exist, but still the success rate is low. One of the many reasons of the failed projects is the inappropriate methodology being followed by the software companies. With new problems arising these methodologies evolve from traditional to more iterative methods. At first, the traditional waterfall model was used for all kind of projects in which the requirements of the project were collected, analyzed, and locked at the start of the project. Once the requirements are fixed, the implementation was carried out. No change in the requirements was supported by this model. Later, with the urge to deliver more quality products and to satisfy end users, iterative methodologies were introduced to incorporate changes. Over the time, bundle of SDLC models came into existence. Due to the factor of diversity and multiple methodologies, it might would have been difficult for the project to select the appropriate

methodology for their projects. Using the most suitable SDLC model can help to increase the success rate of the project and decrease the time and effort needed for the development. With suitable model, the risks related to the project and uncertainty are minimized and managed, respectively. Along with this, quality of the project is improved, and project can be better tracked and controlled.

This research will help analysts to perform documents analysis task more efficiently. For document analysis, software project charter is selected. To select this document as an input, we conducted a survey from students and employees working in different software houses and companies. Based on this survey, we have identified that charter is the first document that is created at the project initiation phase. Based on this document, the stakeholder and the project manager specify if the project needs to be carried out or not. So, this research is the contribution in the field of Software project management where the most suitable SDLC model will be selected, according to the project characteristics that are derived from the software project charter.

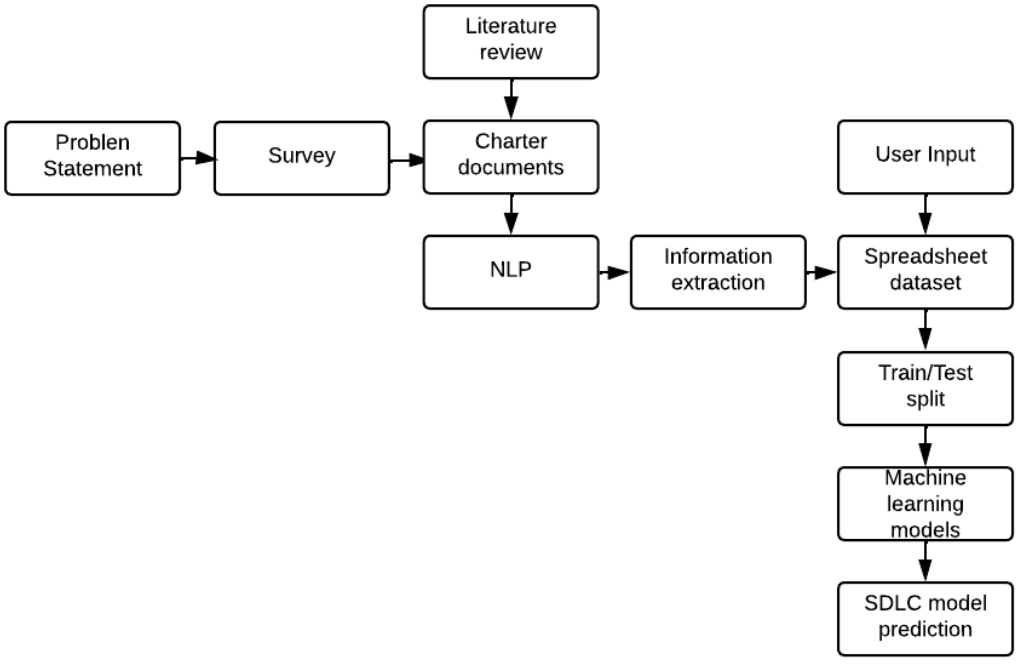


Fig 1. 1: A detailed block diagram to illustrate the high-level scope and workflow of the thesis, starting from problem statement to the software development life cycle model prediction for any project.

1.2. Motivation and Problem Statement

In software engineering, one of the major issues is the selection of best SDLC model as it depends on various factors. It may affect the success of the project as all stages of software development process are based on the type of model selected. For this, not only a wide range of knowledge is required, but also the input from experience experts, software practitioners and developers are considered. This research will help to minimize the manual efforts and time needed for the prediction of SDLC model. It will focus on text analysis of charter document to identify the important factors that can aid in this selection process.

1.3. Objectives

Following are the main objectives of this research:

- To process and evaluate Software project charter document to extract useful information needed for the SDLC model prediction.
- To recommend the most suitable SDLC models based on the extracted project characteristics.
- To minimize the time and efforts wasted for the manual documents analysis.

1.4. Thesis Contribution

As per the best of our knowledge, no prior work is done by the researchers to extract features from software project charter, that can help to predict the software development life cycle model for that project.

The main contributions of this work are stated below:

- We conducted a survey to get experts opinion on the dataset creation. Based on the survey results, we selected software project document as an input.
- We developed a dataset comprising of 71 software project charter documents. Prior to this, no dataset for software project charter exists.
- We have cross-verified the dataset from two software houses before implementation.
- We have worked on information extraction and applied 11 different machine learning algorithms to compare the results.

• Finally, we developed a semi-automatic system to predict SDLC models based on software charter document.

1.5. Thesis Organization

We have organized our thesis into the following main sections:

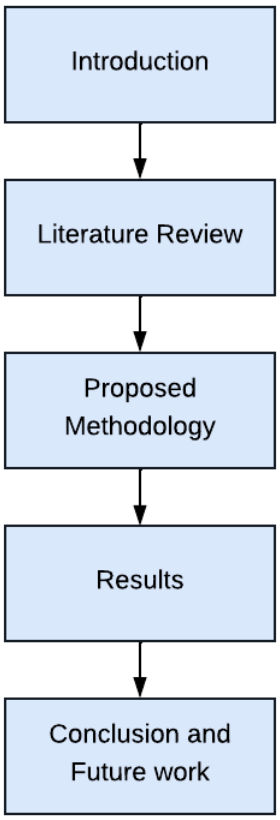


Fig 1.5. 1. Thesis organization in five chapters, comprising of Introduction, Literature Review, Proposed Methodology, Results, Conclusion and Future work.

- Chapter 1 contains the introduction of the thesis which contains the problem statement, our contribution in this thesis along with the proposed objectives of the thesis.
- Chapter 2 contains literature review that is carried out during this research work. In this chapter, we have included the Software development life cycle models along with the work done in past to predict suitable SDLC model. The NLP techniques and ML algorithms that are used in this research are discussed.

- Chapter 3 includes the proposed solution and the methodology used to predict the suitable SDLC model along with the working. The dataset that is used in this research is also explained in detail.
- Chapter 4 covers the brief description of each ML algorithm and the achieved results.
- Chapter 5 contains the conclusion of the work done, challenges faced while working on it, followed by the future work.

Chapter 2

Literature Review

2.1. Overview

Software development life cycle (SDLC) model is a methodology that is followed by the project managers to plan and execute a project from start to end. There are various approaches of SDLC that exist in the literature. One of them is Traditional models where sequential series of steps are carried out. These traditional models are linear, rigid, and less flexible. In Iterative models, all the stages are designed in a way that they shall be revisited again to adapt any required changes. Some of the important SDLC models are Waterfall, Agile, V-Model, Extreme Programming, Evolutionary and Incremental [2].

2.1.1. Waterfall model:

Waterfall is a traditional, structural, and static approach in which the development is carried out in a linear manner. One activity needs to be completed before the next activity starts which means that one cannot go back to the previous phase as it does not support overlapping. To make one adjustment, all phases need to be changed, making it more expensive. It supports extensive documentation and is used in critical projects where changes are not welcomed, and all the requirements are specified and locked before the implementation. One of the major disadvantages of this system is that, if the client is not sure about the system requirements at the start, it does not support amendments and enhancements [8].

In the first phase of requirement identification and analysis, requirements are collected from the customers and are refined. These requirements are stored in the repository and act as an input for the later stages of implementation. In second phase, system architecture design is created for the actual implementation of the system. Third phase is system development, where the actual implementation of the project occurs. Software developers perform unit testing of the modules before the testing of the system occurs in forth phase. In this phase, software testers check the quality and check if the system is implemented as per the specifications or not. Test cases are developed to check the functional aspects of

the system as per the checklist and to measure expected and actual outcomes. In the last phase, the system is deployed into the real world for the actuals users. Maintenance is the last phase; the product maintenance is given by the company. In this phase, the customers report any problem that is faced by them to the development team, and they fix it. Along with this, system updates are also included in Maintenance [7].

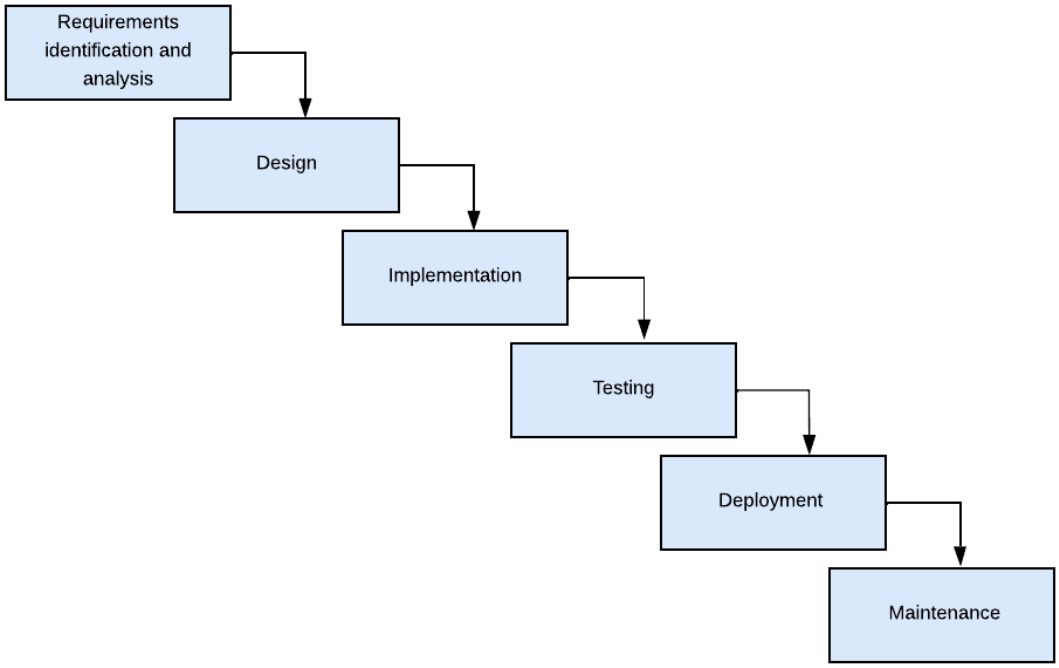


Fig 2.1. 1: Waterfall Model in which software development is carried out in a linear way which means one activity needs to be completed before the next activity starts.

2.1.2. Incremental model:

Incremental model is the modification in waterfall model to improve the functionality. In incremental model, the functionality that is more important, core or that is risky, is developed first. Project is divided into different increments, combining linear model with iterative model; thus, risk is also distributed across. In the first increment, the basic and main requirements are fulfilled. Other supplementary requirements are addressed in the next increment [5]. The results and feedback of one increment are used as feedback for the next increment along with the customer's input. Their involvement can help to identify the risks in advance as customers perform a detailed review. For incremental project development, a proper planning is needed and the functionality that is more needed is

delivered first. It does not support frequent change in functionalities but accept seldom or slow changes. Customer involvement is high and complete project is delivered at the end. It is iterative in nature and can need fewer people at the start of the project [12].

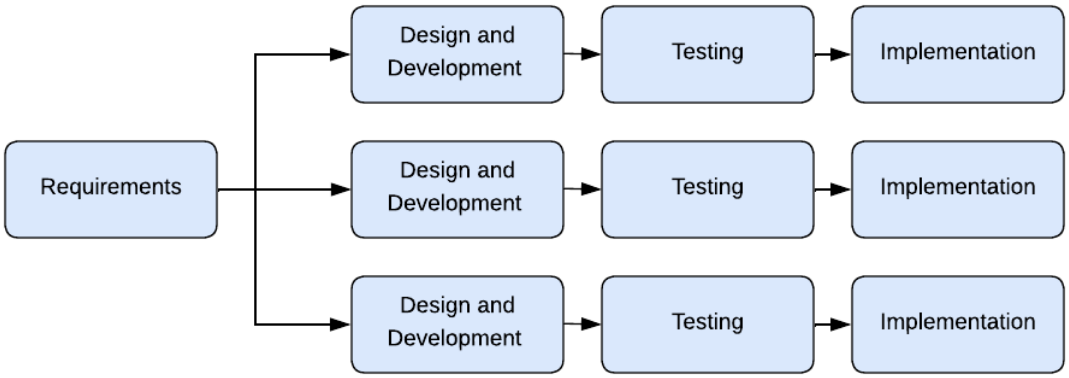


Fig 2.1. 2: Incremental model in which project is divided into multiple increments with the goal to complete the most important and core functionality first.

2.1.3. Evolutionary model:

Evolutionary model is like iterative model but except it does not expect a product that is useable for the users in each iteration. Development of the product per iteration is based on the specified categories instead of the importance of features. If the technology is not well understood by the practitioners, then this model is preferred. If the requirements are not well understood at the beginning of the project, then this technology is preferred. With user involvement and feedback, the system features can be improved. A strong management is needed in this model as developers are not much sure about the architecture and algorithm so, it involves high risk [13]. Thus, evolutionary model is used when details of the project are not well defined and can be understood later based on the basic objectives specified by the customer. The system is divided into the smaller work products or chunks, and then it is delivered to the user which in return validates the system. Change is requirements are welcomed and can be handled well [14].

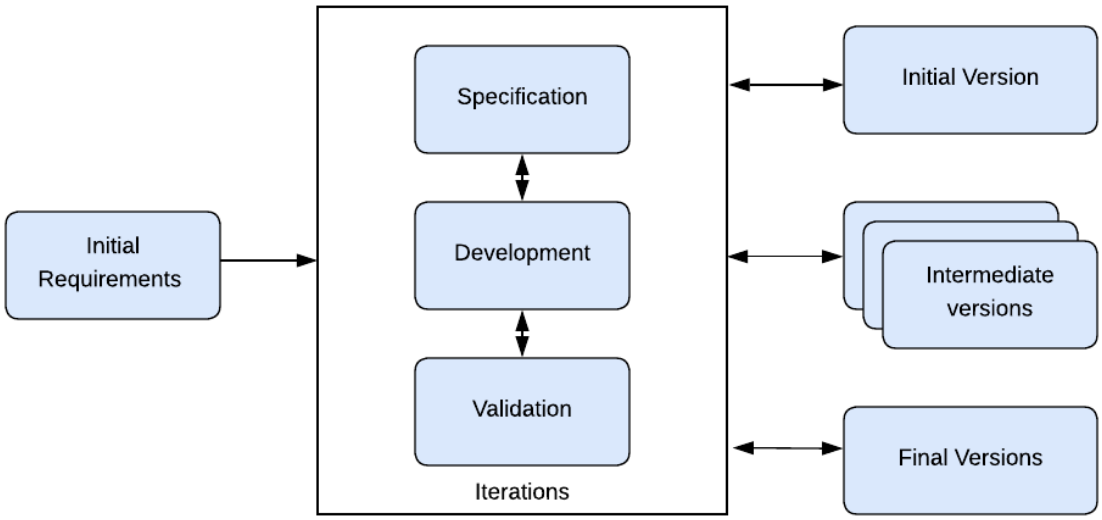


Fig 2.1. 3: Evolutionary model in which system is divided into the smaller work products and the feedback given by customer as well as the change is welcomed.

2.1.4. Hybrid Model:

Two or more Software development life cycle models are combined to create a Hybrid model. According to the customer requirements and need, these models can be adjusted. If we need the combined effect of two models to be implemented in our system, we go for hybrid model. It can be used for any project, either its small, medium, or large. Following are the two major SDLC models that can be used together as Hybrid model:

1. Prototype and Spiral model

   When there is dependency and requirements are specified in each phase as customer or developer is new to the market and is unfamiliar with the technology and software requirements, then this combination of models is used. Customers specify the requirements per module, and prototype of the first module is created. Testing is done for that prototype, and after testing, customer approval is needed. Once the customer approve, design is created followed by development, and testing by the tester. This process continues till all the modules are developed.

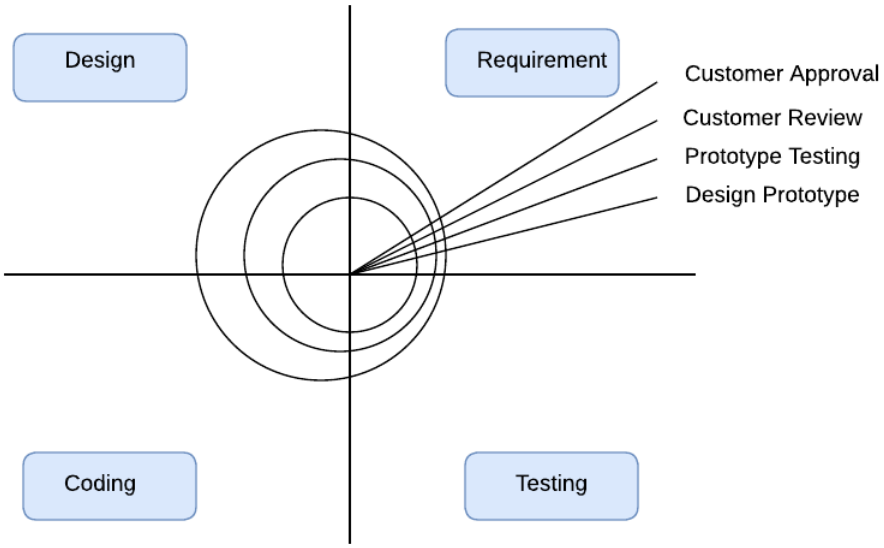


Fig 2.1.4. 1: Prototype and Spiral model as Hybrid model in which customer specify requirements in each module which is further divided into design, coding, and testing

2. Verification and Validation model and Prototype model

   This Hybrid model is used when the customer and developer are not much familiar with the technology, but the customer expects a high-quality product. In it, developers and testers work in parallel to deliver the high-quality product. In this model, business needs are recorded in CRS which is then evaluated by tested along with acceptance testing. This CRS is then converted into a proper SRS document followed by creating the Prototype designs. Tester will then check SRS document and make test cases for the system testing. They also prototype to identify bugs if exists and send back the bug report to the developers. Customers are also involved here for the review of prototypes. After their approval, High level system design is generated, and integration testing documents are generated by testing team. In next step, low level design of the system is created by the developers and reviewed by testers with functional test cases. Finally, the developer do the implementation per prototype and perform white box testing before delivering it to the testing team. This process continues until the prototypes are complete and stable [15].

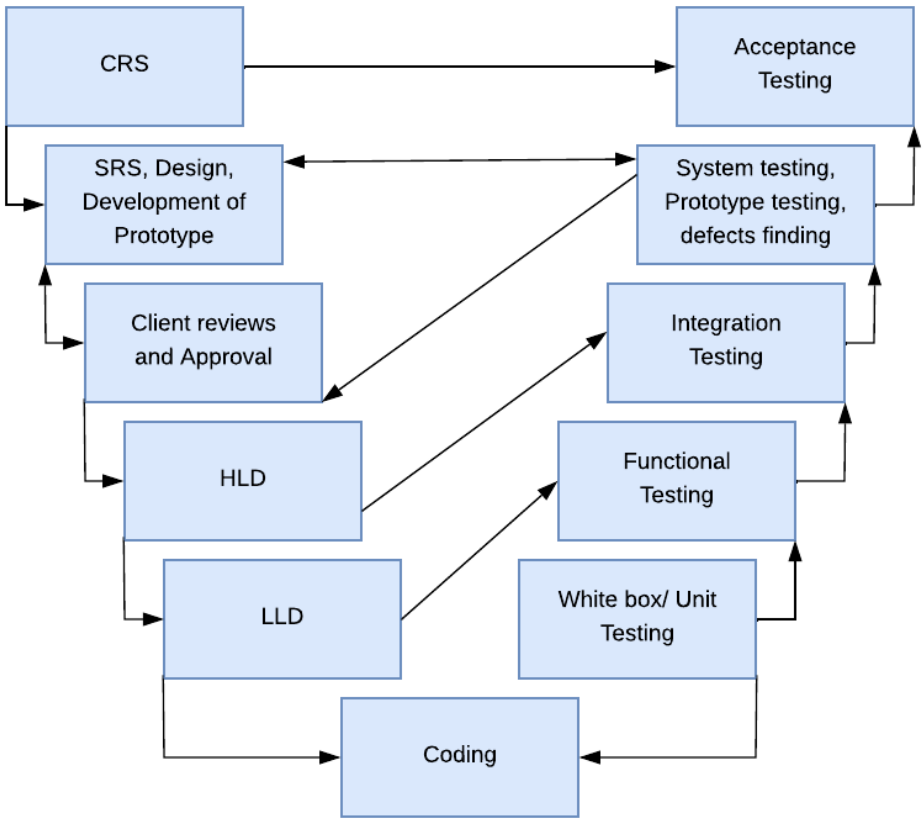


Fig 2.1.4. 2: V&V and Prototype model in which developers and testers work in parallel to deliver the high-quality product

2.2. Selection of SDLC model:

In literature, various approaches exist to select the most suitable SDLC model based on the current project characteristics. Kuldeep et al. proposed a rule-based recommendation system that recommend the most suitable SDLC model based on various product's characteristics [3]. Various questions to the developer regarding size, risk, complexity, standard, reliability, customers, networks, and time are asked to the developer. Production rules are defined as a set of 'If…Then' rules and are stored in rule repository. Another module of this system is knowledge acquisition module that acquire new rules form the expert if required. Models that are being discussed by this study are iterative waterfall, XP, Scrum, RAD, prototyping, incremental and spiral. Based on the answers given by the

developer, the system makes recommendation for the best suited SDLC model. This system is semi-automatic where the input from developer is mandatory for the model prediction.

Mumtaz et al. proposed a method for the selection of suitable Software development life cycle models using Analytic Hierarchy Process. Different selection criteria's that can influence the decision of choosing the most suitable SDLC model are identified. These factors are set of requirements, development time and cost, changes in requirements and complexity of the system. At next step, a hierarchical structure of SDLC model incorporating these factors into Traditional, agile and hybrid models, was presented. After that a decision matrix is needed to calculate ranking values, and a Binary search tree needs to be constructed to apply the in-order tree traversal technique to get the prioritized list of SDLC models. In this method, for pair wise comparison, decisions like which criteria is more important and should have more value are taken manually [4].

A Alshamrani et al. compared three SDLC models waterfall, spiral and incremental considering Requirement specification, Time, complexity, change in requirements, cost, flexibility, simplicity, risk, customer involvement, testing, maintenance, and ease of implementation [5]. Their strengths, weaknesses and where they should be used is also discussed. In waterfall model, the requirement specifications are known at the beginning. They are inflexible, with less customer involvement. This model is inappropriate when the project is long, complex and where the requirements change frequently. In this model, Testing is not done frequently, but is carried out at the end. In Spiral model, not all requirements are known at the beginning. This model is appropriate when the project is long, complex and where the requirements change frequently. It is a little bit more flexible than waterfall model. Customer involvement is low, but after each iteration and testing is done at the end of each phase. In incremental model, not all requirements are known at the beginning. This model should be used when the project is long, complex and where the requirements change frequently. It is more flexible than Spiral model. Customer involvement is high and is after each iteration. In this model, testing is done at the end of each iteration.

P.M.Khan et al. [6] proposed a selection matrix to choose the best SDLC model for different Projects. They classified various models under the umbrella of Traditional and Agile methodologies and identified the risks that can occur due to the wrong SDLC model

in business-critical software Projects. They considered Waterfall, 2I-Process models (Iterative/incremental) and V-Process in traditional models and XP and RUP in Agile process. A survey was conducted from 59 IT professionals to explore the real-life experience of management over the wrong selection of SDLC model. Along with this, they also studied 11 real life projects from various organizations to evaluate right selection of SDLC model selection and consequences of choosing the wrong one. Based on the perceived results, decision support matrix was developed to aid in the suitable selection of SDLC model.

2.3. Natural Language processing (NLP):

Natural language processing (NLP) is an approach of artificial intelligence that is widely used to extract useful information from plain text documents and raw text [39]. To make machine understand human language, NLP came into existence. It is widely used in Artificial intelligence, data mining, information retrieval, Linguistics, and text analysis. It helps to ease the human computer interaction by learning the syntax and context of natural language. A quick and efficient information retrieval is possible by natural language processing techniques and tools from a repository of millions of documents, text and images. One of the areas where Natural language is mostly used is to perform tasks with natural language. As natural language is ambiguous, semantic based textual information retrieval is now also supported by NLP algorithms [48]. Output of NLP systems can be text as well as image, depending upon the user requirements. Some of the text processing algorithms used in NLP are Seq2Seq Model, named entity recognition Model, User preference graph, Word Embedding, Phrase Based Machine Translation, and Neural Machine Translation (NMT) [47].

Following are the main NLP techniques that are frequently used in text analysis and data mining.

- **Sentence Segmentation definition:**

  To understand human language better, sentence segmentation is the first step where whole text, comprising of different paragraphs are broken down into sentences [40]. The breakdown is done by identifying the boundaries between the sentences, which is mostly the punctuation mark [41].

Example:

Paragraph:

Selection of SDLC model is important for the success of the project. No one model is fit for all types of projects. The choice of SDLC selection varies according to the characteristics of the project.

Sentence Segmentation:

1. Selection of SDLC model is important for the success of the project.
2. No one model is fit for all types of projects.
3. The choice of SDLC selection varies according to the characteristics of the project.

- **Word tokenization definition:**

  After the sentence segmentation, the next mostly used step is word tokenization where the sentence is further split into separate words, referred as tokens. These tokens are identified by word boundaries instead of sentence boundaries and are also termed as word segmentation.

  Example:

  Sentence:

  No one model is fit for all types of projects.

  Tokens:

  “No”

  “one”

  “model”,

  “is”

  “fit”

  “for”

  “all”

  “types”

  “of”

  “projects”

- **Stemming definition:**

Stemming in natural language processing refers to the conversion of a word into its root word or base form. It helps in analyzing the token to understand what the text is about to improve indexing and searching. Context or part of speech of the sentence is not understood by stemming [42].

Example:

Words:

change, changes, changing

Stemming:

chang

- **Lemmatization definition:**

  It returns the lemma, comparing to stemming, which is an actual word. Text with the same meaning is connected through lemmatization [43].

  Example:

  Words:

  plays, play, playing

  Lemma:

  play

- **Stop words analysis definition:**

  The words that appear more often or more frequently are of least importance and are terms as stop words. They are removed from the text to do the processing to the only important terms.

  Example:

  Sentence:

  Selection of SDLC model is important for the success of the project.

  Stop words:

  of, is, for, the

- **Dependency parsing definition:**

  This NLP technique is used to analyze how each word in the sentence is related to the other word. It is done by tree structure named as dependency tree, where a word can be assigned as a parent term and the relationship of all other terms is identified [44].

- **Part of speech tagging definition:**

  In this approach, each term is assigned the part of speech that is belonged to it, to understand its meaning.

  Example:

  Sentence:

  Islamabad is the capital and most crowded city.

  POS tagging:

  Islamabad -> Proper noun

  is -> verb

  the -> determiner

  capital -> noun

  and -> conjunction

  most -> adverb

  crowded -> adjective

  city -> noun

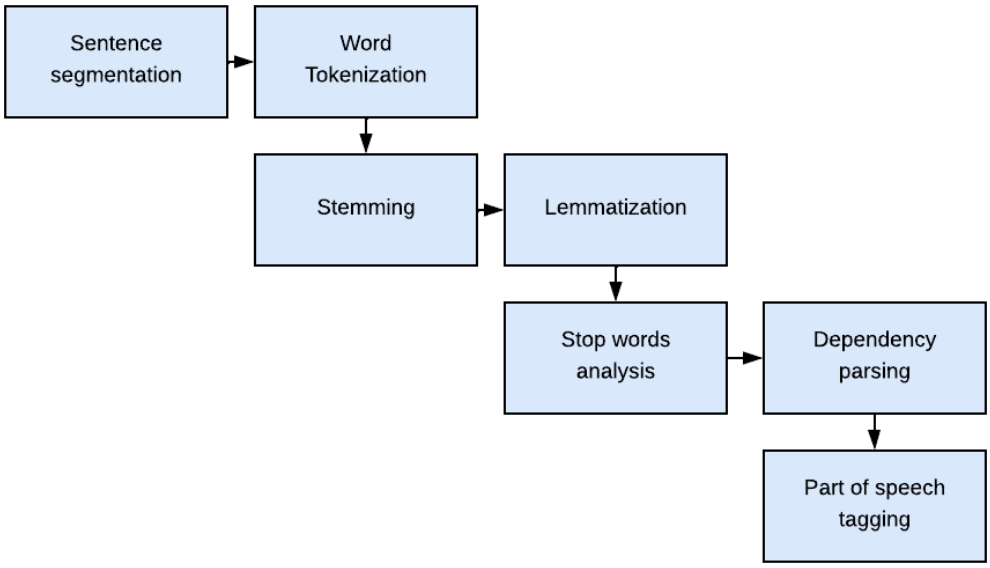


Fig 2.3. 1: A pipeline of Natural language processing tasks that helps to understand human language in artificial intelligence

2.4. NLP in Software engineering:

Right requirement engineering led to the right software project development. Different NLP tools and techniques are used in Software Engineering to extract the elements of interest, that can help to accelerate and improve the processes. We can classify NLP approaches into symbolic NLP and statistical NLP.

In Symbolic NLP, deep linguistic analysis is done through algorithms and approaches that use rule-based systems or semantic networks. Issue of Symbolic NLP is this, that is it is inflexible to adopt new languages since explicit human handwritten rules are used in it. They do not work well with unfamiliar inputs also. POS tagger is an example of the NLP tool that is based on Statistical NLP.

In Statistical NLP, a large linguistic corpus is used along with various Machine learning algorithms. It is simple, robust and helps to develop probabilistic model. As these models learn through data and trained over the large set data so they produce good results NLP can solve the issues in requirement specifications that are ambiguous, unnecessarily complicated, missing, wrong, duplicated, or conflicting. In this regard, researchers have proposed different NLP tools, libraries and techniques that improve Software engineering processes. Table 2.1 shows how different practitioners use NLP in various Software engineering phase.

Table 2. 1: Literature review of NLP tools, Techniques, and development characteristics in Software engineering

Sr. No	**Research paper's name**	**Observations**	**NLP Techniques**	**Development**
1	**Generating UML Class Diagram from Natural Language Requiremen**	-Aids in requirement analysis and Design. -Create UML class diagram from textual requirements using Natural Language processing,	Sentence Tokenization, Word Tokenization, stop words removal,	**System Name:** Requirement engineering analysis and design (READ) **Language:** Python

	ts: A Survey of Approaches and Techniques	historic rules, and domain ontology techniques. -Class diagram generated as a result includes class name, attributes, methods, and relationships [16].	Stemming, POS Tagging	**IDE:** Visual Studio **Library:** NLTK **GUI:** Tkinter library
2	**A Novel Framework to Automatically Generate IFML Models from Plain Text Requirements**	-Automate the development of Interaction Flow Modeling Language (IFML) models. -IFML Domain and Core models are generated from initial plain text requirements by using NLP. -Identified set of rules to obtain important elements and information from the requirement document [17].	POS Tagging, sentence splitting, tokenization.	**Tool name**: Text to IFML (T2IF) **IDE:** Visual Studio **Language:** C# **Database:** SQL Server 2008 **Framework:** SharpNLP **Library:** Regular expression library
3	**Generating UML Class Diagram using NLP Techniques and**	-Generate class diagram from Software requirements Specifications -Classes, attributes, methods, association, aggregation, composition,	Sentence and Word Tokenization, POS Tagging, Lemmatization and Stemming, Parse Trees and	**Framework/Tool:** Stanford CoreNLP **Library:** NLTK

	Heuristic Rules	dependency, and recursive relationships are extracted. -Use NLP and set of heuristics. -Reduce cost and time required for manual and design processes [18].	Type dependencies, Open Information Extraction	
4	**SDLC Model Selection Tool and Risk Incorporation**	-This tool consists of two main components. -The first component is called "Comparison metric generation", in which past organization's projects data is used, SDLC model are derived and then a comparison metric is generated. -The second component is called "SDLC Model selection", that fetch the project priorities, calculate score of SDLC models and then select suitable SDLC model [19].	Not specified	Not specified

5	**Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches**	-Automatic approach to identify and classify five Non-Functional Requirements from Software Requirement documents. -NLP techniques and ML algorithms are used with semantic and syntactic analysis. -CNN Approach and Word embedded models are best among others for NFR identification/classification and requirement sentence representation, respectively. -Classification accuracy can be improved by the fusion of multiple NLP techniques [20].	Tokenization, Punctuation Removal, Stop word Removal, Case folding, Parts of Speech tagging(POS) , Lemmatization, TF-IDF, Word2Vec, BERT	**Library:** NLTK **Dataset**: PURE Environment **Cloud service:** Google Colab 9 Pro-Language Python 3.7

In IT industry, the skilled and experience software project managers and developers are costly human resources and dependency on such resources should be less. One way to cope with this factor and to save cost is a step towards automation of various tasks related to Software engineering. Globalization of the IT industry has increased the pressure on business enterprises [55]. Over the past few years, the demand of minimum time spent on manual processing and analysis of the requirement documents is of great importance in the IT industry. There is a strong need for the techniques that use different levels of detail in the requirement specification to bring out the required results.

In software engineering, one of the major issues is the selection of best SDLC model as it depends on various factors. It may affect the success of the project as all stages of software development process are based on the type of model selected [56]. For this, not only a wide range of knowledge is required, but also the input from experience experts, software practitioners and developers are considered. This research will help to minimize the manual efforts and time needed for the prediction of SDLC model. It will focus on text analysis of available documents to identify the important factors that can aid in this selection process. This research will also be a good step towards automation and will help software practitioner in decision making process. It will also produce the initiative among graduates to produce and develop expert systems that can help to catch up the pace with rest of the world in software engineering domain.

Chapter 3

Proposed methodology

3.1. Proposed Architecture:

We have divided this project into the four major sections. The first section is document selection for input. For this purpose, we have conducted an online survey from software practitioners and researchers. The result of this survey is used in the second section which is the dataset generation. At this section, 71 software charter documents are created manually. The third section is information extraction, where python language is used for the required data extraction from the dataset. The last section is for training different Machine learning models on the dataset. The broader overview of the project is specified as follows: The system will take software project charter as an input. Tokenization is performed to split the document into sentences and regular expressions are applied to extract the characteristics needed. After that, those extracted information is fed into the excel file along with the expert input. Different Machine learning algorithms are trained over it to predict most suitable SDLC model for the Project.

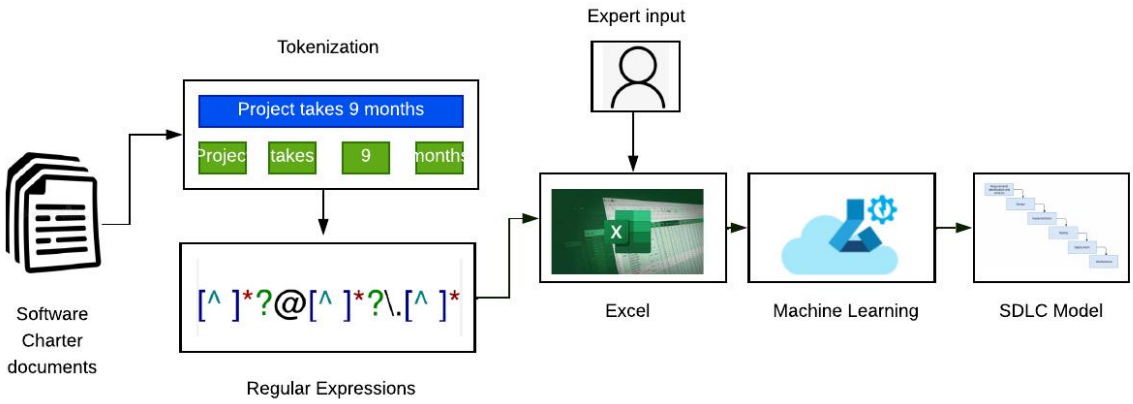


Fig 3. 1: Proposed Architecture diagram of the proposed work in which software charter documents are fed as an input, then regular expressions are used to extract information which is stored in excel file along with the user input. After that, machine learning models are applied to predict SDLC models.

3.2. Survey:

To select which document should be considered as an input, we have conducted a survey from students and employees working in different software houses and companies. This was an important step as, as per the best of our knowledge, no literature exists where the researchers have specified the document that should be created before SDLC model selection. This selection process in still done manually by organizations even with the excessive documentation. As documentation is mandatory for a successful software project, it is also important to consider a document as a base for SDLC model. That document will be helpful to extract project characteristics based on which the selection of model depends. The survey that we have conducted was solely for research purpose and a brief description of the project was also specified in it.

3.2.1. Research Questions:

We prepared a survey form, comprising of 5 questions that we asked from the Software experts and the students of software engineering field. The detail of each question is stated as follows:

- Profession
  - Student
  - Employee
- Company/ Institute name
- Email
- Company's CMM Level (Software Capability Maturity Model)
  - CMM Level 1
  - CMM Level 2
  - CMM Level 3
  - CMM Level 4
  - CMM Level 5
  - Not specified

  Note: Leave this part if you are filling this survey as a student
- Which document/documents do you consider before selecting any particular SDLC model for your Project?

- Software project charter
- Domain Analysis Document
- Feasibility Study
- Scope Document
- Risk Management Plan
- Stakeholder Analysis Document
- Business Requirement Document
- Cost Benefit Analysis document
- Concept note.
- Software project plan.

3.2.2. Participants:

The total of 84 participants were involved in this research, out of which 46 participants were students with the major of software engineering and 38 responses were collected from the employees of different software houses. We have collected this data by creating the online survey in Google forms and distributed it across different online channels including Facebook, WhatsApp, Instagram, and LinkedIn.

3.2.3. Inclusion and exclusion criteria:

We have included only those participants in this research who have an IT background. The survey form is distributed among students and researchers who are enrolled in the discipline of Software engineering, computer science, Information technology or computer engineering, in a reputed university.

Secondly, we have distributed it across the employees who are working in any software house or are a part of any IT project. All other individuals and professionals are excluded from this study.

3.2.4. Statistical analysis:

The data from the google forms was exported in .xlxs file. Then we used excel tools and techniques to generate graphs. Some of the graphs are directly exported as image from the from the Summary section in google forms.

According to the data analysis, 45.2% of the responses were collected from Employees, working in different software houses across Pakistan and 54.8% of the responses are

collected from the students of different reputed universities. The response is illustrated in Appendix A.

45 responses were collected from the companies regarding Software capability maturity model level of their organization. In 48.9% of the companies, CMM is not specified, or the employees are unfamiliar with it. 8.9% of the companies are at CMM level 1, 6.6% of the companies are at CMM level 2, 17.8% of the companies are at CMM level 3, 8.9% of the companies are at CMM level 4 and 8.9% of them are at CMM level 5. The response is illustrated in Appendix B.

For the main question for which we conducted this survey that which document should act as input for SDLC model for your project, we received 84 responses. 77.4% of the participants believe that Software project charter should be the document that we should consider for selecting SDLC model. 20.2% selected Domain analysis document, 32.1% voted for Business requirement document,34.5% of them selected Scope document, 31% selected Risk management plan, 40.5% preferred software project plan, Feasibility study was selected from 29.8% of the respondents, 25% of the participants selected Cost benefit analysis document, and 11.9% of them went with concept note. Most of the people selected Software charter document, so we selected it as a base document. The detailed illustration of the response from this survey question is shown in Appendix C.

To make our survey more helpful and realistic, we took the response of just employees and filtered out the responses of students. The reason of applying this filter was to get the response from the people who are working in IT industries, instead of the people who are a bit naïve to the practical work. 81.5% of the Employees suggested to take software charter documents as an input to the SDLC model selection process. 26.3% of them agreed on Domain analysis document, 31.6% selected Feasibility Study, 44.7% of them selected Scope Document, 26.3% of them selected risk management plan, 36.8% of them selected Stakeholder analysis document, 39.5% of them selected the business requirement document, 26.31% of them selected the cost benefit analysis document, 7.8% of them selected the concept note and 34.2% of the experts selected software project plan. According to this statistic, the document with the major outputs was also software project charter. The bar graph representing these values is shown is Appendix E.

3.3. Software Project charter:

Based on the studies and survey results, Software project charter is selected as an input document for SDLC model selection. The first phase to start any project is Identification phase [22]. In this phase, business problem is stated, and opportunities are identified in a project charter. Project charter and Risk matrix are the two main deliverables of Project charter activity in the first phase, after project charter is developed, the Stakeholder analysis and Project plan are created simultaneously.

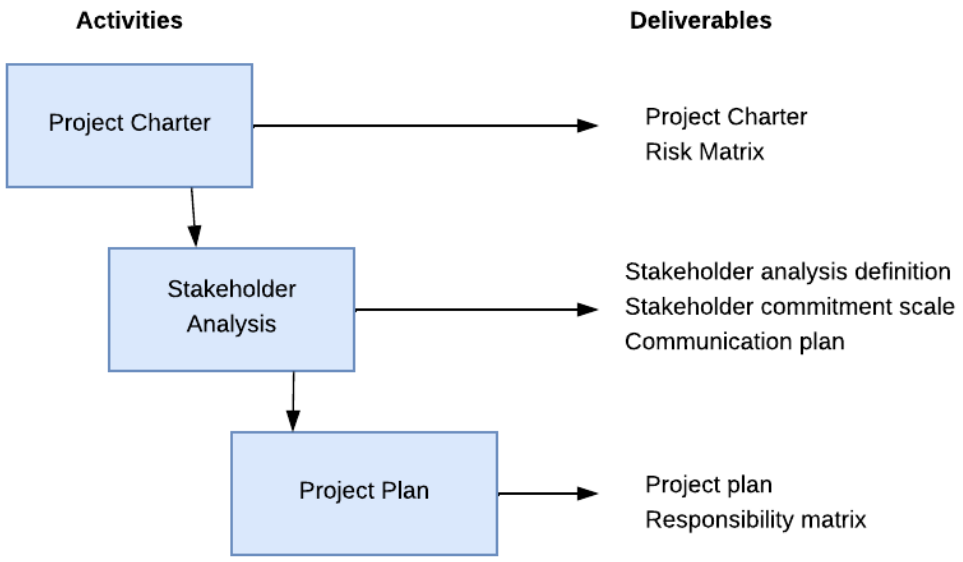


Fig 3. 2: Software project identification phase, where software project charter creation is the first activity before proceeding further with the project.

Software Project charter is the high-level document with the project overview which states what needs to be done in a project, why it should be done and how it will be done. Business case of the Project is also presented in it. Project charter helps to make it clear to the business developers and sponsors to have a view on the project goal dates, its estimated cost and other important factors before proceeding forward [21]. All the stakeholders agree on the project on this charter document. It is different from project plan where the project and its characteristics, resources and estimations are specified in detail. Charter is created at the start of the project and is not changed through out the project unless al stakeholders are agreed to. It is a foundation to structure and kickoff any software project [46].

Some of the main elements of project charter are:

- Project name
- Purpose of the project
- Project Team
- Roles and Responsibilities
- Project sponsor
- Project scope
- Objectives of the project
- Goals of the Project
- Milestones
- Project completion date
- Cost assumptions
- Communication plan
- Approval from stakeholders

3.4. Dataset:

This dataset consists of Software charter documents that are used as basis to start a new project. The dataset is created from the public Software requirements related documents that are open to use and are available online. We extracted Software scope documents, Software requirement specification documents (PURE Dataset) and some documents are collected from the NCSAEL (Cyber security research lab, Military college of Signals, NUST). Then we convert those documents into Software charter document. We collected 71 requirement documents after all the research and created our dataset of pdf files.

This dataset can be used in various Natural language processing, Software project management, and Software Requirement engineering tasks. We have included multiple domains in this dataset and different project charter templates. We do not claim that we have used all the documents that are there in PURE dataset [9]. The characteristics that are needed to select most suitable SDLC model are filled by referring to the work done by Linda C. Alexander et al. who presented a criterion that aids to select most suitable software development life cycle model [10].

Some statistics on the content are stated as follows:

- Name: Each document is identified by the unique name given to it
- Number of pages: Number of pages per document are shown in Figure.1. The document with a greater number of pages is 7 and we have minimum of 3 pages. Average no of pages is 4.

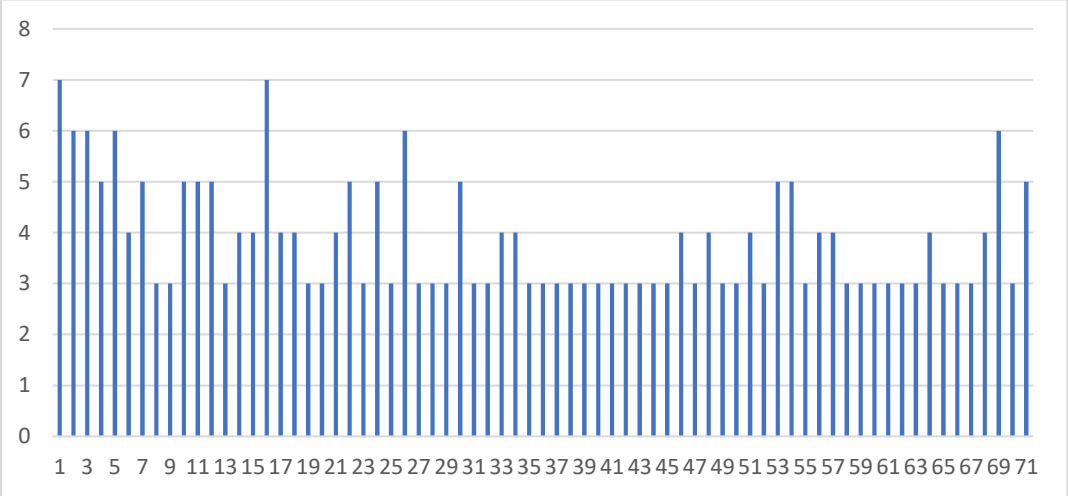


Fig 3.3. 1: Length of the charter documents in dataset with respect to the number of pages. Maximum length of 7, minimum 3 and average of 4 pages exist in this dataset.

- Language: As requirements are expressed in Natural language, so the language of this dataset is English in compliance with the domain specific acronyms.
- Format: The document in this dataset is in both word and pdf format, with the .docx and .pdf extension, respectively. For this project, we used documents with .pdf extension.
- Source: It indicates from what source the base document is the retrieved, based on which the new dataset is created. Letter P indicates the Pure dataset, letter N indicates NCSAEL, and letter O indicates other online resources.

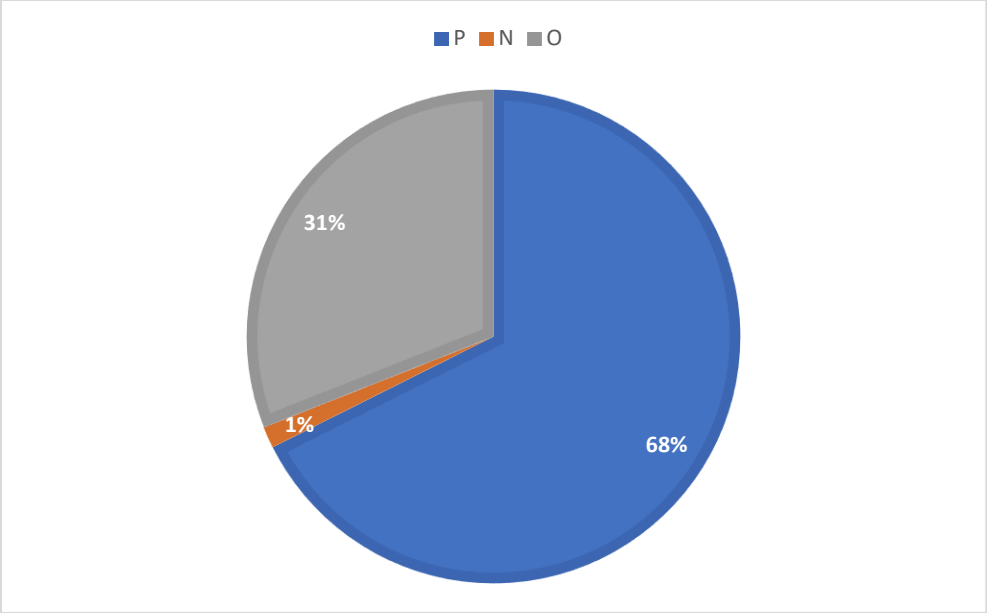


Fig 3.3. 2: Source documents that are referred for the project charter creation where P stands for Pure dataset, N stands for NCSAEL and O stands for other online resources.

3.5. Dataset cross-evaluation:

We performed dataset evaluation from two software houses before working on the project. They checked the project characteristics, their values and model predictions based on those project characteristics. The proof of dataset evaluation is attached for reference.

3.5.1. NCSAEL:

NCSAEL is a cyber security lab that is sponsored by Planning Commission and Higher Education Commission of Pakistan. We have contacted Maliha Safdar, who is working as a team lead and got our dataset verified. She checked the characteristics that we have used as an input and verified the output model per characteristic. This helped us to use supervised learning algorithms at our dataset.

The certificate of verification by NCSAEL is attached in Appendix E.

3.5.2. TechEase:

TechEase is a computer software company, providing services since past 6 years. We have contacted Allaudin, who is working as a Chief Technology officer in TechEase and got our dataset verified. He checked the characteristics that we have used as an input and verified

the output model. This helped us to use supervised machine learning models at our dataset. The certificate of verification is attached in Appendix F.

3.6. Data in Charter to support SDLC models:

We refer the work done by Alexander et al. to define the criteria on the basis of which a practitioner can choose the most suitable Software development lifecycle model for their Project [10]. They identified the following criteria for SLDC model selection where V1, V2 and V3 specifies Simple, complex, and difficult for Problem complexity, Seldom, Slow and Rapid for Frequency of change, Small, Large and medium for Project size and so on.

For Waterfall model:

In waterfall model, one activity needs to be completed before the next activity starts which means that one cannot go back to the previous phase as it does not support overlapping. Entry 1 means, for that value and characteristic against it, the project model is suitable and 0 means the SDLC model is in-appropriate for that entry. Factors that are necessary to consider for selecting Waterfall model are specified in the Table 3.5.1.

Table 3.5. 1: Criteria to select Waterfall model based on project characteristics where V1, V2 and V3 specifies the characteristic to be Simple, Complex and difficult.

Criteria	**V1**	**V2**	**V3**
Problem Complexity	1	0	0
Frequency of change	1	0	0
Product size	0	0	0
Interface requirements	1	0	0
Funds availability	0	0	0
Staff availability	0	0	0
Time schedule	1	1	0
Developer experience	1	1	1

For Incremental model:

In incremental model, the functionality that is more important and core or that is risky, is developed first. This model divides the project into different increments. Entry 1 in the table means, for that value and characteristic against it, the project model is suitable and 0 means the SDLC model is in-appropriate for that entry. Table 3.5.2 specified the factors that are necessary to consider for the selection of incremental model.

Table 3.5. 2: Criteria to select Incremental model based on project characteristics where V1, V2 and V3 specifies the characteristic to be Simple, Complex and difficult.

Criteria	**V1**	**V2**	**V3**
Problem Complexity	1	1	0
Frequency of change	1	1	0
Product size	1	1	1
Interface requirements	1	1	0
Funds availability	0	1	1
Staff availability	0	1	1
Time schedule	1	1	0
Developer experience	0	1	1

For Evolutionary model:

In Evolutionary model, development of the product per iteration is based on the specified categories instead of the importance of features. Entry 1 in the table means, for that value and characteristic against it, the project model is suitable and 0 means the SDLC model is in-appropriate for that entry. Factors that are necessary to consider for selecting Evolutionary model are specified in the Table 3.5.3.

Table 3.5. 3: Criteria to select Evolutionary model based on project characteristics where V1, V2 and V3 specifies the characteristic to be Simple, Complex and difficult.

Criteria	V1	V2	V3
Problem Complexity	1	1	1
Frequency of change	1	1	1
Product size	1	1	1
Interface requirements	1	1	1
Funds availability	1	1	1
Staff availability	1	1	1
Time schedule	1	1	1
Developer experience	0	1	1

For Hybrid model:

Hybrid model is the combination of two models. Entry 1 in the table means, for that value and characteristic against it, the project model is suitable and 0 means the SDLC model is in-appropriate for that entry. Factors that are necessary to consider for selecting Hybrid model are specified in the Table 3.5.4.

Table 3.5. 4: Criteria to select Hybrid model based on project characteristics where V1, V2 and V3 specifies the characteristic to be Simple, Complex and difficult.

Criteria	V1	V2	V3
Problem Complexity	1	1	1
Frequency of change	1	1	0
Product size	0	0	0
Interface requirements	1	1	1
Funds availability	0	0	0
Staff availability	0	0	0
Time schedule	1	1	1

Developer experience	0	1	1

3.7. Feature Extraction:

Once the dataset is generated, the next step is feature extraction from pdf files. Natural language processing (NLP) is used along with the Regular expressions to extract the elements of choice. We have extracted the following characteristics:

Month calculation:

We focused on three main formats to calculate the total no of months required to complete any project.

Syntax 1: If the time is specified in months.

This supports the words that end with any of the following words (month, months, months., month.)

Following can be some of the possible sentences that can match this regular expression:

- The project will be finished in 3 month
- The project will be finished in 14 months
- The project will be finished in 14 months.
- The project will be finished in 3 month.
- The project will be completed in 3 month
- The project will be completed in 14 months
- The project will be completed in 14 months.
- The project will be completed in 3 month.
- The project needs 3 month to finish.
- The project needs 14 months to finish.
- Total time required for this project is 14 months
- Total time required for this project is 14 months.
- Total time required for this project is 3 month.
- Total time required for this project is 3 month
- This project has the total duration of 14 months
- This project has the total duration of 14 months.
- This project has the total duration of 3 month.

- This project has the total duration of 3 month
- For this project, 14 months are required.
- For this project, 3 month is required.

Syntax 2: If the project starts and end date is specified.

For start date, it supports the words like start on, start at, starts on, and starts at. For end date, it supports the words like finish on, finish in, finished on, finished in, complete on, and complete in.

Some of the possible formats are:

- The project will start on 18/06/2020 and will finish on 18/04/2022.
- The project will start on 18/06/2020 and will finish in 18/04/2022.
- The project will start on 18/06/2020 and will be finished on 18/04/2022.
- The project will start on 18/06/2020 and will be finished in 18/04/2022.
- The project will start on 18/06/2020 and will be complete in 18/04/2022.
- The project will start on 18/06/2020 and will be complete on18/04/2022.
- The project will start at 18/06/2020 and will finish on 18/04/2022.
- The project will start at 18/06/2020 and will finish in 18/04/2022.
- The project will start at 18/06/2020 and will be finished on 18/04/2022.
- The project will start at 18/06/2020 and will be finished in 18/04/2022.
- The project will start at 18/06/2020 and will be complete in 18/04/2022.
- The project will start at 18/06/2020 and will be complete on18/04/2022.
- The project starts on 18/06/2020 and will finish on 18/04/2022.
- The project starts on 18/06/2020 and will finish in 18/04/2022.
- The project starts on 18/06/2020 and will be finished on 18/04/2022.
- The project starts on 18/06/2020 and will be finished in 18/04/2022.
- The project starts on 18/06/2020 and will be complete in 18/04/2022.
- The project starts on 18/06/2020 and will be complete on18/04/2022.
- The project starts at 18/06/2020 and will finish on 18/04/2022.
- The project starts at 18/06/2020 and will finish in 18/04/2022.
- The project starts at 18/06/2020 and will be finished on 18/04/2022.
- The project starts at 18/06/2020 and will be finished in 18/04/2022.

- The project starts at 18/06/2020 and will be complete in 18/04/2022.
- The project starts at 18/06/2020 and will be complete on18/04/2022.

Calculate Experience:

For this, we need developer experience. The words use for Resources can be Software developers, Senior developer, Senior Programmer, Junior developers, Junior Programmer, Developer, and Programmer.

It supports the five table header formats from where information can be extracted which is specified in the Table 3.6.1.

Table 3.6. 1: Five supported formats/templates for table header in software charter document to extract developer experience

Resources Required	Average Experience
Resources Required	Average Experience in years
Role	Average Experience
Roles	Experience (in years)
Roles	Experience

Some of the possible formats with examples are shown in Table 3.6.2:

Table 3.6. 2: Sample examples to extract developer experience from Software project charter document with regular expressions

Resources Required	**Average Experience**
Software developers	13 years
Resources Required	**Average Experience in years**
Software developers	11
Role	**No of resources**
Senior Programmer	10
Junior Programmer	7
Role	**Average Experience**
Senior developer	10 years
Junior developers	7 years

Roles	**Experience (in years)**
Developer	10 years
Roles	**Experience**
Programmer	10

Calculate team members availability:

Table 3.6.3 and table 3.6.4 specify the header formats that are supported by the project to calculate team members availability.

For example, the total number of team required in this case is 78.

Table 3.6. 3: Tables format # 1 that is supported by our code to calculate availability of team members

Resources Required	**No of resources**	**Average Experience**
Software developers	35	21 years
Software Tester	23	03 years
Software Analyst	10	05 years
Quality Assurance Lead	5	02 years
Training Leader	5	04 years3

Total number of team members required in this case are 21.

Table 3.6. 4: Table format # 2 that is supported by our code to calculate availability of team members

No.	**Role**	**Average Experience**	**Personals**	**Responsibility**
1	Project Manager	11 years	1	Lead the project
2	Senior Programmer	6 years	4	Will do coding
3	Junior Programmer	4 years	7	Will do coding
4	Software Architect	3 years	3	Will design the system
5	Quality assurance engineer	3 years	2	Ensure quality
6	Software analyst	7 years	4	Analyze the system

Cost:

It supports any number followed by the word lac or lacs. Some of the possible supported lines are:

- The project needs 12 lacs to finish.
- The approved budget for the Project is 82 lacs
- Total budget of the Project is 86 lacs.
- Total specified budget for the project is 8 lac.
- The cost to complete and deploy this project will be 22 lacs.
- According to the Estimate, The Overall cost of the Project is 20 lacs which involve the cost of human resources, development cost and other required material cost.

After the required features are extracted from this dataset, our system will generate a dataset.xlsx file which contain different projects along with their extracted features in the Excel file. This document will consist of 6 columns which are stated as below:

- Column 1: This column consists of Project name from which the project characteristics are extracted. Datatype of this column in categorical.
- Column 2: This column contains the Time schedule which is categorized as Short, Enough and Plenty. It is extracted as numeric datatype from the pdf (Number months =3). After that, it is converted to either of the category. The project which requires <6 months are categorized as Short, the project with the duration in between 6-12 months are categorized as Enough and the project with >12 months duration is categorized as Plenty [11].
- Column 3: This column contains Developer Experience in years. Developer can be either Software developers, Senior developer, Senior Programmer, Junior developers, Junior Programmer, Developer, or Programmer. It is divided into the the categories of 1-5,5-10 and 15+ and its data type is numerical.
- Column 4: This column contains availability of staff members, with numeric data type. The Staff members are categorized as 0-10, 10-50 and 50+ employees. These employees include Project managers, developers, Software quality assurance

engineer, Tester, Testing lead, Project lead, System analyst or any other member involved in the project.

- Column 5: This column contains project size, which is classified as small, medium, and large. The data type of this column is categorical.
- Column 6: This column contains the extracted information of the funds available for the Project. These funds are divided into three ranges, 1-10, 10-30 and 30+ lacs. Datatype of this field is Numeric.

Once these features are extracted, we will add the user input. User can be project manager, senior software developer or project lead. The characteristics that are needed in by the user per software project are:

- Problem complexity specifies the degree of complexity of the project. It can be either simple, complex, or difficult, based on the Project scope. The datatype of this column is categorical.
- Frequency of changes specifies the degree of extend to which the problem might change in the future. It can be categorized as seldom, slow, and rapid and its datatype is Categorical.
- Interface requirements specifies whether the user interface is heavy or simple to use. Interface requirements can be minor, significant, critical and its datatype is Categorical.

After that, some preprocessing is done before applying Machine learning models. The document name column is deleted, and other columns are re-arranged. The final excel file before the ML implementation contains the information specified in the Table 3.6.5.

Table 3.6. 5: Features to select SDLC models in excel file with their column no and Data type

Column no	**Data/Parameters**	**Data type**
1	Time schedule	Categorical
2	Developer Experience	Numeric
3	Problem Complexity	Categorical
4	Frequency of change	Categorical

5	Project Size	Categorical
6	Interface requirements	Categorical
7	Funds	Numeric
8	Availability of staff members	Numeric
9	Model	Label

Chapter 4

Results

4.1. Ordinal Encoder:

As our dataset contained categorical features, so they need to be converted into the numerical columns for scikit-learn classifiers to work. With string data, machine learning models tries to find any hierarchical relationship or preference and lead to the misinterpretation [49]. So, we have applied Ordinal encoder to convert our categorical data of Time schedule, Problem Complexity, Problem Complexity, Project Size, and Interface requirements in numeric form [50].

Sample dataset:

Short	8	Simple	Seldom	Large	Minor	82	69
Short	20	Simple	Seldom	Small	Minor	86	37
Enough	8	Simple	Seldom	Small	Minor	62	43

Dataset after Ordinal Encoding at first three column:

[‘0’, ‘8’,’0’,’0’,’2’, ‘0’,’82’,’69’],

[‘0’,’20’,’0’,’0’,’0’,’0’,’86’,’37’],

[‘1’,’8’,’0’,’0’,’0’,’0’,’62’,’43’]

In the above example, the first column which specifies Project time had the categorical values. After one hot encoding, value 0 is given to Short, 0 to Simple, 0 to Seldom, 2 to Large, and 0 to Minor. We applied the same technique to other column values as well.

4.2. Training dataset:

We split our dataset into training and testing data. The ratio of train, test split is 70:30 where 70% of the data is used for training and 30% of data is used for testing. For splitting, we used train_test_split library from sklearn.

4.3. Machine Learning Models

4.3.1. KNN:

K-Nearest Neighbors algorithm is used for classification and regression problems and is a nonparametric method with lazy learning. K closest examples in each space are used as an input for a given N training vectors and neighbors are those objects with the closest or same value. Distance function is used so each feature needs to be scaled in a same way and Euclidean distance between two points are calculated with the following formula:

$$d(X, Y) = \sqrt{(a1-a2)2+(b2-b1)2}$$

Its performance mainly depends on training set [24]. Our accuracy at this model is 77.27%. Misclassified sample in this algorithm are 4. Figure 4.1.1 shows the confusion matrix of KNN with predicted and true labels.

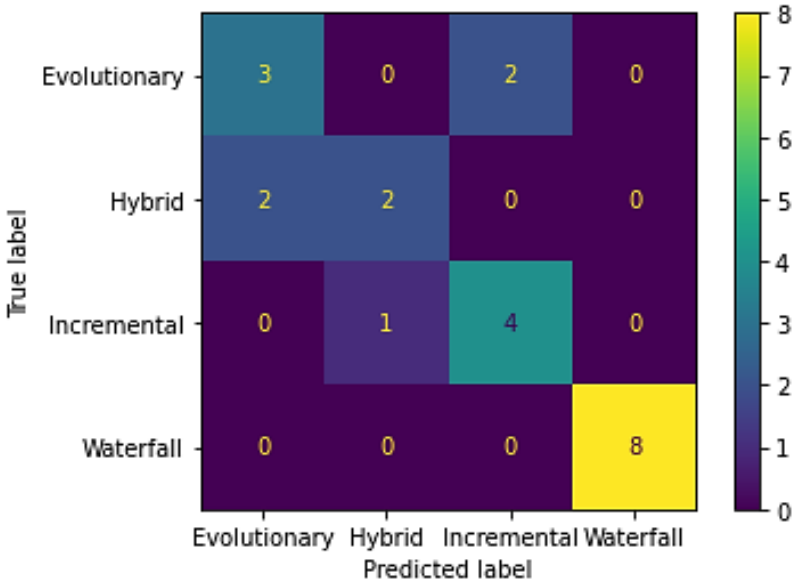


Fig 4.1. 1: Confusion matrix to display true labels and predicted labels for KNN, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.2. Gradient Boost classifier:

It contains a group of Machine learning algorithms that combine different machine learning model to improve the strength. “Sklearn” machine learning library is used to implement this algorithm. A weak hypothesis is chosen to make tweaks repeatedly to change it to a strong model. It reduces the loss [25]. We got the accuracy of 86.363% in our problem by

using Gradient Boost classifier. Misclassified sample in this algorithm are 3. Figure 4.1.2 shows the confusion matrix of Gradient boost classifier with predicted and true labels.

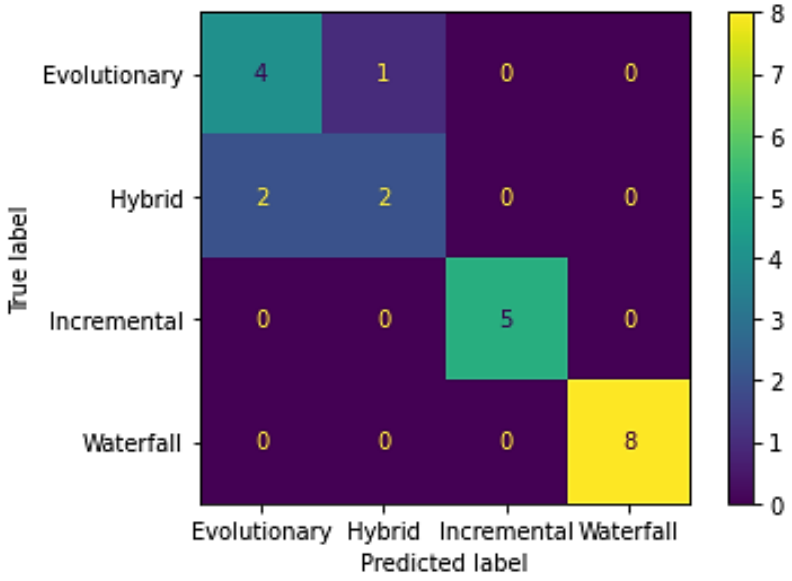


Fig 4.1. 2: Confusion matrix to display true labels and predicted labels for Gradient boost classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.3. SVM:

Support vector machine (SVM) is used for classification, regression and to detect outliers. It supports both binary as well as multiclass classification [26]. Decision boundary is created by SVM to segregate classes into n-dimensional space Extreme points, also known as support vectors, are used to create the best decision boundary which is known as hyperplane [27].

While using Support vector machine at our dataset, we got the accuracy of 77.272%. Misclassified sample in this algorithm are 6. Figure 4.1.3 shows the confusion matrix of Support Vector Machine with predicted and true labels.

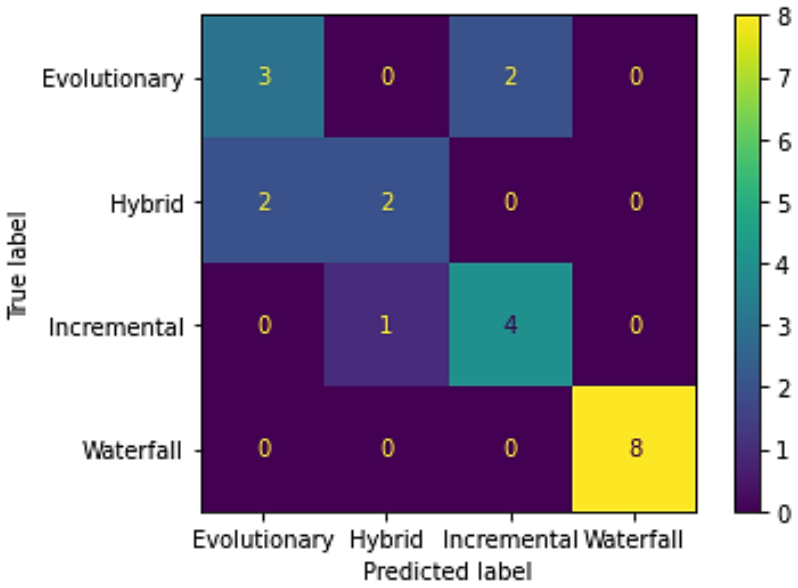


Fig 4.1. 3: Confusion matrix to display true labels and predicted labels for Support vector machine (SVM), where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.4. Naïve Bayes:

It is the most used algorithm in data mining that is used for classification problems. It assumes that one feature that exists in the class is independent of the other features [23]. At out dataset, Naïve Bayes got the accuracy of 90.909%. Misclassified sample in this algorithm are 4. Figure 4.1.4 shows the confusion matrix of Naïve Bayes with predicted and true labels.

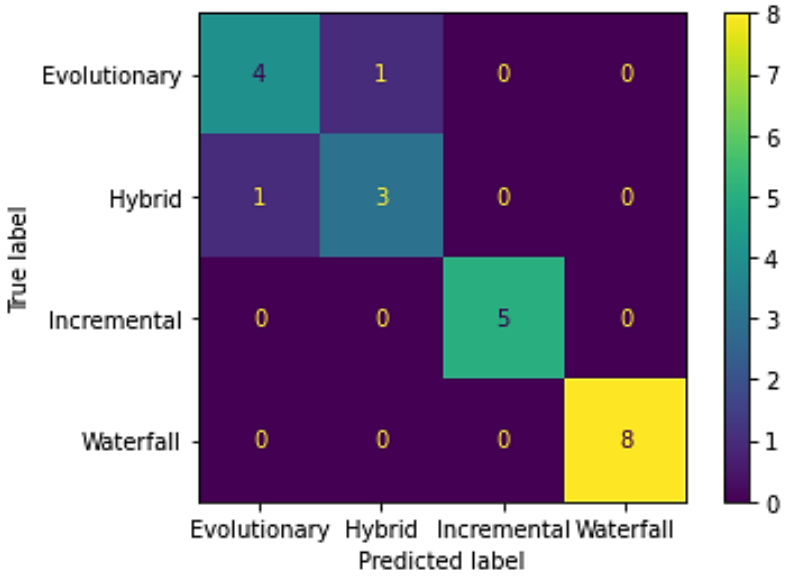


Fig 4.1. 4: Confusion matrix to display true labels and predicted labels for Naive Bayes, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.5. Random Forest Classifier:

Random forest classifier is used for classification, prediction, and regression problems. It is a combination of different tree classifiers, in which each tree vote for the most suitable class and result is found after combining the results of all [28]. Its classification accuracy is high and at our dataset, we got the accuracy of 86.363%. Misclassified sample in this algorithm are 3. Figure 4.1.5 shows the confusion matrix of Random Forest classifier with predicted and true labels.

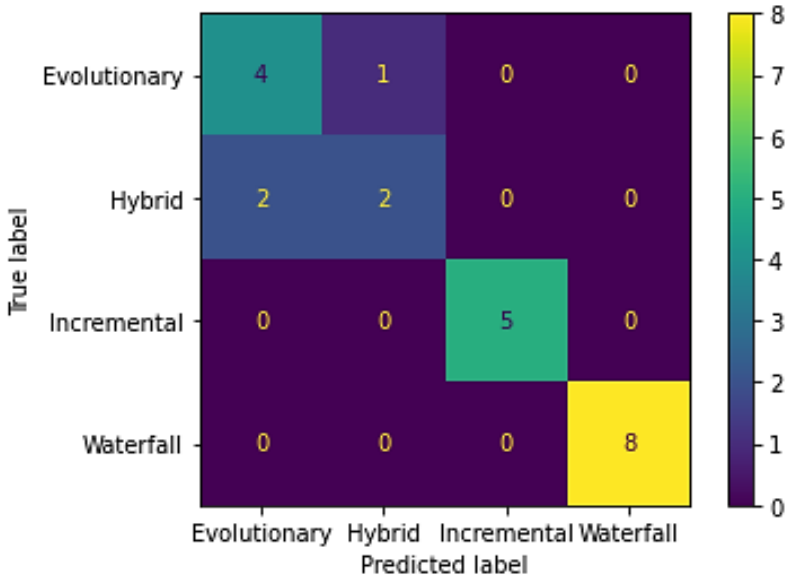


Fig 4.1. 5: Confusion matrix to display true labels and predicted labels for Random Forest classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.6. Ada Boost classifier:

In this algorithm, linear combination of member classifiers is used to reduce the changes of error in each cycle in the process of training the dataset [54]. This combination of weak classifiers is then used to make a stronger classifier, by adjusting the weights per iteration. The weights of the sample with are classified correctly are decreased and misclassified training data samples weights are increased [29]. In our project, we got the accuracy score of 77.272%. Misclassified sample in this algorithm are 3. Figure 4.1.6 shows the confusion matrix of Ada boost classifier with predicted and true labels.

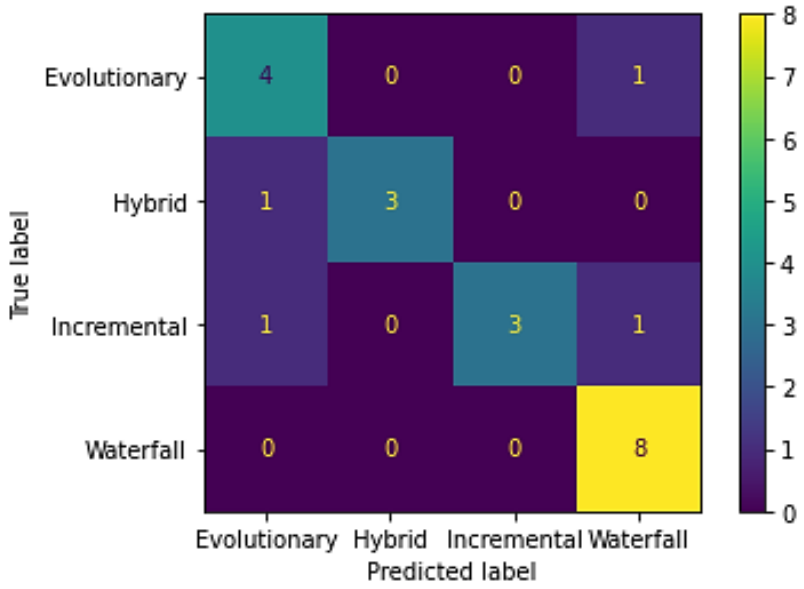


Fig 4.1. 6: Confusion matrix to display true labels and predicted labels for Ada boost classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.7. Linear Discriminant analysis:

In linear discriminant analysis, maximum separability is achieved as the algorithm increase the variance between and within class ratio. Its major application is speech recognition and majorly used in data classification [53]. It draws the decision region between the classes and the location of the original dataset does not changes [30]. We got 77.272% accuracy by using this algorithm. Misclassified sample in this algorithm are 4. Figure 4.1.7 shows the confusion matrix of Linear Discriminant analysis with predicted and true labels.

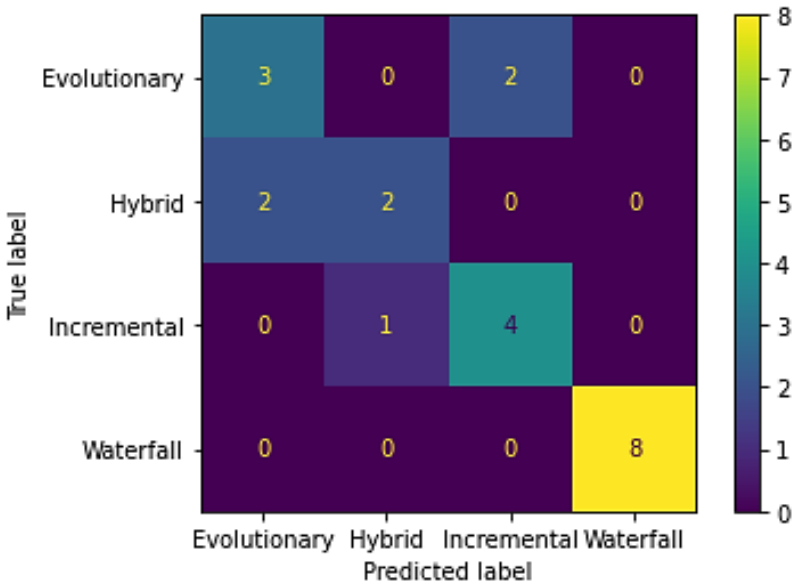


Fig 4.1. 7: Confusion matrix to display true labels and predicted labels for Linear discriminant analysis, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.8. Ridge Classifier:

It is a supervised classifier which is based on Ridge regression and is used to analyze linear discriminant model [31]. It converts the sample data in [-1,1] form and reduce overfitting by penalizing coefficients and reduce complexity [32]. To improve classification, and to reduce variation, we have specified the value of alpha parameter to 10. The total accuracy for Ridge classifier is 72.727%. Misclassified sample in this algorithm are 4. Figure 4.1.8 shows the confusion matrix of Ridge classifier with predicted and true labels.

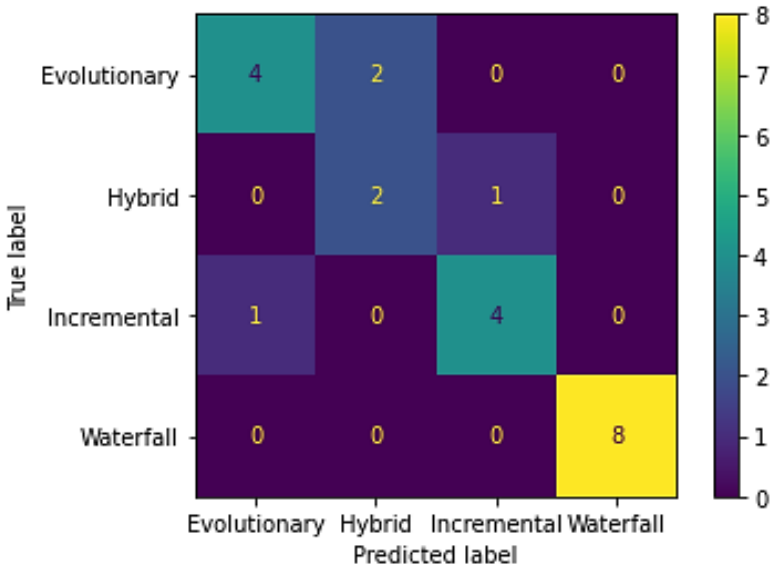


Fig 4.1. 8: Confusion matrix to display true labels and predicted labels for Linear discriminant analysis, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.9. Decision tree classifier:

It is a supervised learning technique, mostly used for classification problems. Decision trees use decision functions to classify an unknown sample in a class. It consists of a root node, and interior nodes that represents features, decisions rules are specified at the branches and the terminal nodes that describe final classification or outcome [33]. Layer is the nodes at a particular level with the same distance from the root node. Decision nodes contains multiple branches while Leaf nodes represent output and are not used for decision [34]. The accuracy of 81.818% is achieved by this algorithm. Misclassified sample in this algorithm are 6. Figure 4.1.9 shows the confusion matrix of Decision tree classifier with predicted and true labels.

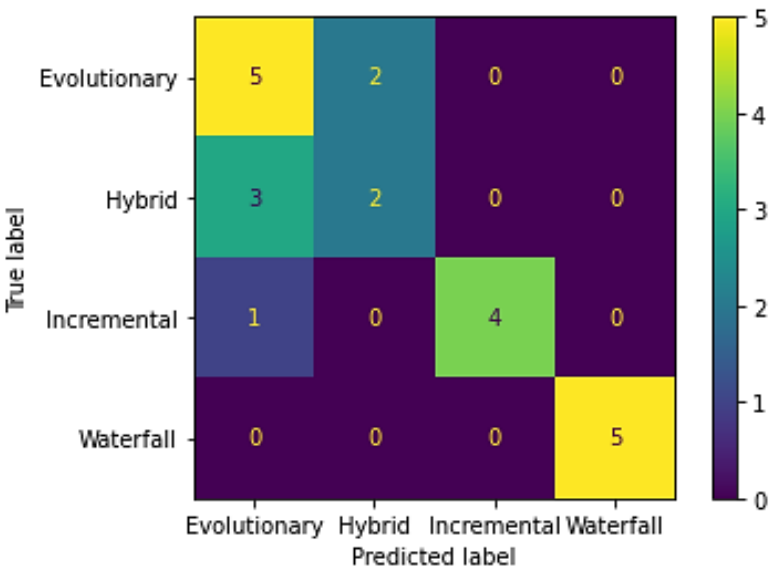


Fig 4.1. 9: Confusion matrix to display true labels and predicted labels for Decision Tree classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.10. Light gradient boost classifier:

Tree based learning algorithms are used by light gradient boost classifier with low memory usage and high efficiency. It can easily handle large scale data [35]. It splits the tree by leaf wise instead of level wise and expects less loss as compared to the level-wise algorithms [36]. With our dataset, Light gradient boost classifier obtained the accuracy score of 81.818%. Misclassified sample in this algorithm are 5. Figure 4.1.1 shows the confusion matrix of Light gradient boost classifier with predicted and true labels.

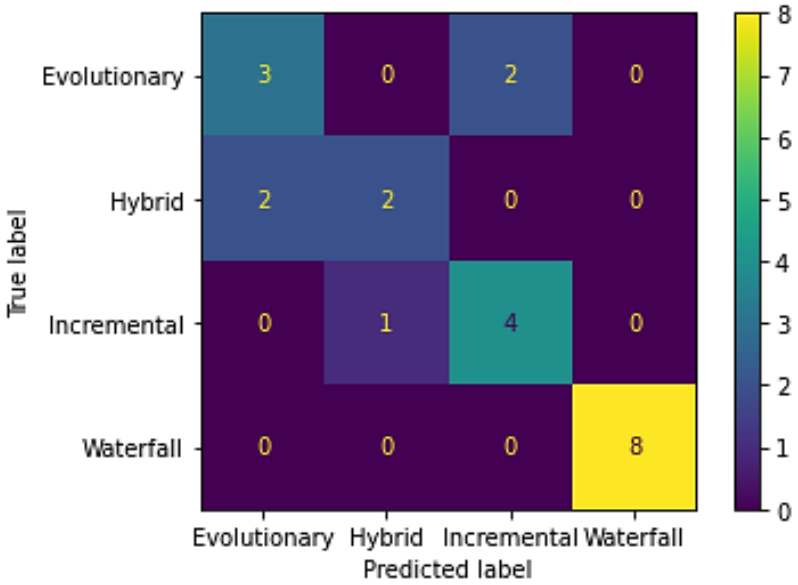


Fig 4.1. 10: Confusion matrix to display true labels and predicted labels for Light gradient boost classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.3.11. Extra Tree classifier:

Extra Tree classifier is a machine learning algorithm that works on randomization of decision trees to reduce reduction and it merge the results of various decision trees [37]. This algorithm is good to control overfitting problem. One of the parameters used in it is n_estimator which refers to the number of trees in the algorithm and whose value ranged from 10 to 100 [38]. With this algorithm, we got the accuracy of 81.818%.

In testing dataset, out of 22 samples, only 2 samples are misclassified by this algorithm. Figure 4.1.1 shows the confusion matrix of Extra tree classifier with predicted and true labels.

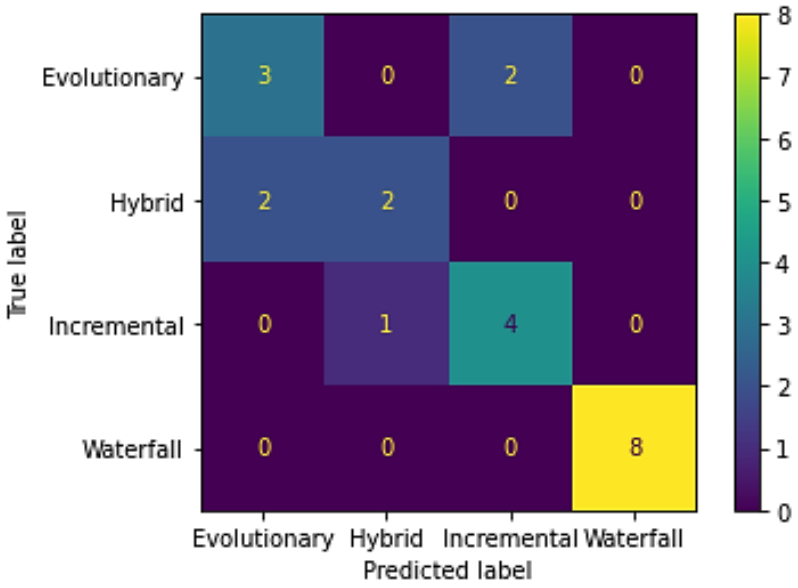


Fig 4.1. 11: Confusion matrix to display true labels and predicted labels for Extra Tree classifier, where the values in diagonal represents the elements that are predicted true by the classifier.

4.4. Comparison Table:

To compare the results of eleven algorithms that we have applied at our dataset, we created a comparison table. This table illustrates the Model name, Accuracy, precision, recall, F1 score and Accuracy with K-fold validation. For K-fold validation, we have used the value of K=5 [52]. Without K-Fold validation, we get the best results with Naïve Bayes classifier and with K-fold validation, we get the best results with SVM. The detail results are specified in Table 4.2.1

Table 4.2. 1: Comparison of 11 different Machine learning models based on Accuracy, Precision, Recall, F1 score and Accuracy score with K-fold cross validation

	ML Model	**Accuracy (in %)**	**Precision**	**Recall**	**F1 Score**	**Accuracy score-K-fold cross validation**
1	**Gradient Booster Classifier**	86.363	90.454	86.3636	86.338	77
2	**Linear Discriminant Analysis**	77.272	83.901	77.272	77.301	82

3	**Naïve Bayes**	90.909	93.506	90.909	91.207	78.06
4	**Random Forest Classifier**	86.363	90.454	86.363	86.338	78
5	**KNN**	77.27	81.186	77.272	75.719	80.99
6	**Light Gradient Boosting Machine**	81.818	84.415	81.818	81.060	67.99
7	**Ada Boost Classifier**	77.272	76.641	77.272	76.648	74
8	**Ridge Classifier**	72.7272	74.242	72.7272	73.195	67.99
9	**Decision Tree classifier**	72.727	74.494	72.727	72.916	70
10	**SVM- Linear kernel**	77.272	82.323	77.272	76.082	85
11	**Extra Trees Classifier**	81.818	82.207	81.818	81.722	82

Chapter 5

Conclusion and Future work

5.1. Challenges:

During this research, following are the major challenges that we faced:

- **Document selection:**

  As per the best of our knowledge, no such literature exist that specify which document can act as an input to select SDLC model. Multiple software project documents exist like Project scope document, Domain Analysis document, Stakeholders Analysis document, and Business Requirement Documents. After extensive study and conducting a survey, we selected Software project charter as an input document for SDLC model prediction.

- **Survey response collection:**

  We have created a survey form with five questions and distributed it across various platforms to gather the results. We found it difficult to collect the response from the target market and only managed to get 84 responses.

- **Dataset availability:**

  No prior dataset of Software project charter was available. For algorithm to predict the most suitable SDLC model, we needed a charter dataset on which our Machine learning model had to train and predict the needed results.

- **Dataset generation:**

  Creating charter documents from the scratch was a challenging task. We referred to PURE dataset, NCSAEL, and online resources to generate documents from the content of their SRS and other project documents.

- **Dataset verification:**

  We found it difficult to verify our dataset from Software houses. At the end, we managed to verify it from 2 Software houses named, NCSAEL and TechEase.

5.2. Future Work:

Software engineering is a vast field and continuous improvements can be done based on expert opinions and user's input. This work can be extended to incorporate the following more things:

- **Improvement in dataset:**

  The current dataset is for experimental purpose. It can be extended and further enhanced with the help of industry survey and experts' opinion. Successful and failed software projects across the globe can be considered for dataset improvement.

- **Better data extraction:**

  Data extraction from the documents can be improved. We are using regular expressions to extract the information of interest. The current regular expressions cannot cover the typing mistakes that can be there in the charter document.

- **Fully automation:**

  The project can be fully automated by applying more software engineering approaches.

- **SDLC models addition:**

  We are currently working on four SDLC models that are Waterfall, Incremental, Evolutionary and Hybrid model. In future, we can add more SDLC models like Agile, extreme programming, and Kanban etc.

- **SDLC model selection characteristics:**

  We have used 8 project characteristics for the selection of SDLC model. In future, after extensive study, we can add more characteristics that can be considered to select SDLC model.

5.3. Discussion:

In this research, we have developed a semi automatic system that extract some information from the charter document and consider manager input regarding the project for SDLC model prediction. We have created the dataset of charter document after the response collected from the industry survey as well as after the extensive study. We have applied

eleven ML models. With Extra Trees classifier, the accuracy to predict right SDLC model is 81.818%. Along with this, we have used SVM with accuracy of 77.272%, Decision tree classifier with accuracy of 77.72%, Ridge classifier with accuracy of 77.727%, Light Gradient Boosting Machine with an accuracy of 81.818, KNN with an accuracy of 77.27, Random Forest classifier with an accuracy of 86.363%, Naïve Bayes with an accuracy of 90.909%, Linear Discriminant Analysis with an accuracy of 77.272%, and Gradient Booster Classifier with an accuracy of 86.363%.

5.4. Conclusion:

An appropriate SDLC model contributes majorly to the project success, yet its selection is still a challenging task for the project managers. This thesis specifies NLP and machine learning techniques to propose SDLC model for a project, based on its characteristics. We have generated seventy-one software charter documents that act as an input to machine learning algorithms. Then we have applied eleven different ML algorithm and found the highest accuracy of 90.90% with Naïve Bayes classifier. The values of features that are extracted and obtained as an input, are not definitive. This information is collected from research and is verified by two software houses. The actual correlation between these characteristics and their contribution in the SDLC selection can be done by formal surveys and considering a large amount of successful and unsuccessful software projects.

References

[1] Ragunath, P. K., Velmourougan, S., Davachelvan, P., Kayalvizhi, S., & Ravimohan, R. (2010). Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC). International Journal of Computer Science and Network Security, 10(1), 112-119.

[2] Ruparelia, N. B. (2010). Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3), 8-13.

[3] Kumar, K., & Kumar, S. (2013). A rule-based recommendation system for selection of software development life cycle models. ACM SIGSOFT Software Engineering Notes, 38(4), 1-6.

[4] Khan, M. A., Parveen, A., & Sadiq, M. (2014, February). A method for the selection of software development life cycle models using analytic hierarchy process. In 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT) (pp. 534-540). IEEE.

[5] Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. International Journal of Computer Science Issues (IJCSI), 12(1), 106.

[6] Khan, P. M., & Beg, M. S. (2013, April). Extended decision support matrix for selection of sdlc-models on traditional and agile software development projects. In 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT) (pp. 8-15). IEEE.

[7] Petersen, K., Wohlin, C., & Baca, D. (2009, June). The waterfall model in large-scale development. In International Conference on Product-Focused Software Process Improvement (pp. 386-400). Springer, Berlin, Heidelberg.

[8] Adenowo, A. A., & Adenowo, B. A. (2013). Software engineering methodologies: a review of the waterfall model and object-oriented approach. International Journal of Scientific & Engineering Research, 4(7), 427-434.

[9] Ferrari, A., Spagnolo, G. O., & Gnesi, S. (2017, September). Pure: A dataset of public requirements documents. In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 502-505). IEEE.

[10] Alexander, L. C., & Davis, A. M. (1991, January). Criteria for selecting software process models. In 1991 The Fifteenth Annual International Computer Software & Applications Conference (pp. 521-522). IEEE Computer Society.

[11] Dhami, J., Dave, N., Bagwe, O., Joshi, A., & Tawde, P. (2021, December). Deep Learning Approach To Predict Software Development Life Cycle Model. In 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3) (pp. 1-7). IEEE.

[12] Kute, S. S., & Thorat, S. D. (2014). A review on various software development life cycle (SDLC) models. International Journal of Research in Computer and Communication Technology, 3(7), 778-779.

[13] Verma, S. (2014). Analysis of strengths and weakness of sdlc models. International Journal of Advance Research in Computer Science and Management Studies, 2(3).

[14] Salve, S. M., Samreen, S. N., & Khatri-Valmik, N. (2018). A Comparative Study on Software Development Life Cycle Models. International Research Journal of Engineering and Technology (IRJET), 5(2), 696-700.

[15] "Hybrid Model", Java T point, [Online]. Available: https://www.javatpoint.com/hybrid-model

[16] Bashir, N., Bilal, M., Liaqat, M., Marjani, M., Malik, N., & Ali, M. (2021, March). Modeling Class Diagram using NLP in Object-Oriented Designing. In 2021 National Computing Colleges Conference (NCCC) (pp. 1-6). IEEE.

[17] Hamdani, M., Butt, W. H., Anwar, M. W., Ahsan, I., Azam, F., & Ahmed, M. A. (2019). A Novel Framework to Automatically Generate IFML Models From Plain Text Requirements. IEEE Access, 7, 183489-183513.

[18] Abdelnabi, E. A., Maatuk, A. M., Abdelaziz, T. M., & Elakeili, S. M. (2020, December). Generating UML Class Diagram using NLP Techniques and Heuristic Rules. In 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA) (pp. 277-282). IEEE.

[19] Agarwal, P., Singhal, A., & Garg, A. (2017). SDLC Model Selection Tool and Risk Incorporation. International Journal of Computer Applications, 975, 8887.

[20] Shreda, Q. A., & Hanani, A. A. (2021). Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches. IEEE Access.

[21] Murray, A. (2016). The complete software project manager : mastering technology from planning to launch and beyond. Wiley.

[22] Cudney, E. A., & Furterer, S. L. (2012). Design for Six Sigma in product and service in development : applications and case studies. Crc Press.

[23] Chen, S., Webb, G. I., Liu, L., & Ma, X. (2020). A novel selective naïve Bayes algorithm. Knowledge-Based Systems, 192, 105361.

[24] Pandey, A., & Jain, A. (2017). Comparative analysis of KNN algorithm using various normalization techniques. International Journal of Computer Network and Information Security, 9(11), 36.

[25] Nelson, D. (2022, July 21). Gradient Boosting Classifiers in Python with Scikit-Learn. Stack Abuse. https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/

[26] 1.4. Support Vector Machines. (n.d.-b). Scikit-learn. https://scikit-learn.org/stable/modules/svm.html

[27] Support Vector Machine (SVM) Algorithm - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[28] Liu, Y., Wang, Y., & Zhang, J. (2012, September). New machine learning algorithm: Random forest. In International Conference on Information Computing and Applications (pp. 246-252). Springer, Berlin, Heidelberg.

[29] An, T. K., & Kim, M. H. (2010, October). A new diverse AdaBoost classifier. In 2010 International conference on artificial intelligence and computational intelligence (Vol. 1, pp. 359-363). IEEE.

[30] Balakrishnama, S., & Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. Institute for Signal and information Processing, 18(1998), 1-8.

[31] Kumar, A. (2022, October 3). Ridge Classification Concepts & Python Examples. Data Analytics. https://vitalflux.com/ridge-classification-concepts-python-examples/

[32] Singh, A., Prakash, B. S., & Chandrasekaran, K. (2016, April). A comparison of linear discriminant analysis and ridge classifier on Twitter data. In 2016 International Conference on Computing, Communication and Automation (ICCCA) (pp. 133-138). IEEE.

[33] Swain, P. H., & Hauska, H. (1977). The decision tree classifier: Design and potential. IEEE Transactions on Geoscience Electronics, 15(3), 142-147.

[34] Decision Tree Algorithm in Machine Learning - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[35] Welcome to LightGBM's documentation! — LightGBM 3.3.2 documentation. (n.d.). https://lightgbm.readthedocs.io/en/v3.3.2/

[36] GeeksforGeeks. (2021, December 22). LightGBM (Light Gradient Boosting Machine). https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/

[37] Majidi, S. H., Hadayeghparast, S., & Karimipour, H. (2022). FDI attack detection using extra trees algorithm and deep learning algorithm-autoencoder in smart grid. International Journal of Critical Infrastructure Protection, 37, 100508.

[38] sklearn.ensemble.ExtraTreesClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[39] Turing. (2022, July 22). Natural language processing functionality in ai. Natural Language Processing Functionality in AI. https://www.turing.com/kb/natural-language-processing-function-in-ai

[40] Chowdhary, K. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649.

[41] Palmer, D. D. (2000). Tokenisation and sentence segmentation. Handbook of natural language processing, 11-35.

[42] Jivani, A. G. (2011). A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl, 2(6), 1930-1938.

[43] Khyani, D., Siddhartha, B. S., Niveditha, N. M., & Divya, B. M. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Journal of University of Shanghai for Science and Technology.

[44] Covington, M. A. (2001). A fundamental algorithm for dependency parsing. In Proceedings of the 39th annual ACM southeast conference (Vol. 1).

[45] Ian Somerville, Software Engineering ,Addison Wesley,9th ed,2010.

[46] McKeever, C. (2006). The project charter–blueprint for success. CrossTalk: The Journal of Defense Software Engineering, 19(1), 6-9.

[47] Chowdhary, K. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649.

[48] Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), 544-551.

[49] GeeksforGeeks. (2022, August 23). ML | One Hot Encoding to treat Categorical data parameters. https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

[50] sklearn.preprocessing.OneHotEncoder. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[51] Nur Hidayati, S. (2020). Application of Waterfall Model In Development of Work Training Acceptance System.

[52] Wong, T. T., & Yeh, P. Y. (2019). Reliable accuracy estimates from k-fold cross validation. IEEE Transactions on Knowledge and Data Engineering, 32(8), 1586-1594.

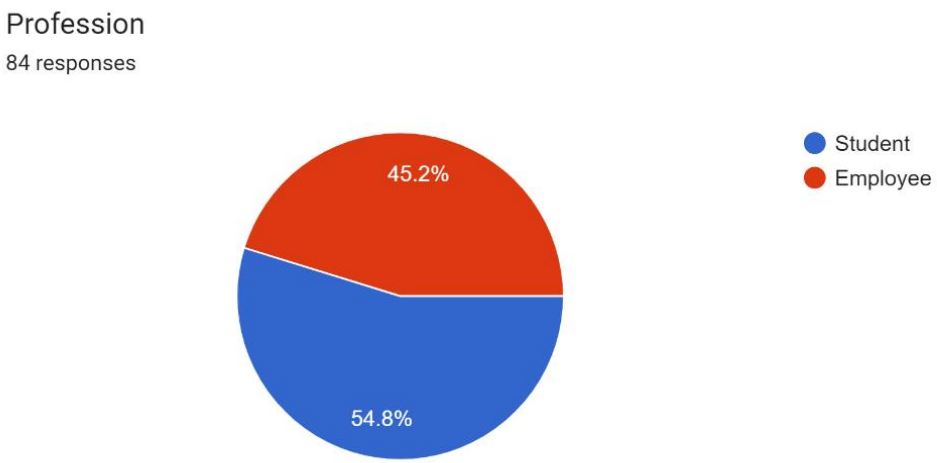
[53] Li, C. N., Shao, Y. H., Yin, W., & Liu, M. Z. (2019). Robust and sparse linear discriminant analysis via an alternating direction method of multipliers. IEEE transactions on neural networks and learning systems, 31(3), 915-926.

[54] Zhang, L., Wang, J., & An, Z. (2021). Vehicle recognition algorithm based on Haar-like features and improved Adaboost classifier. Journal of Ambient Intelligence and Humanized Computing, 1-9.

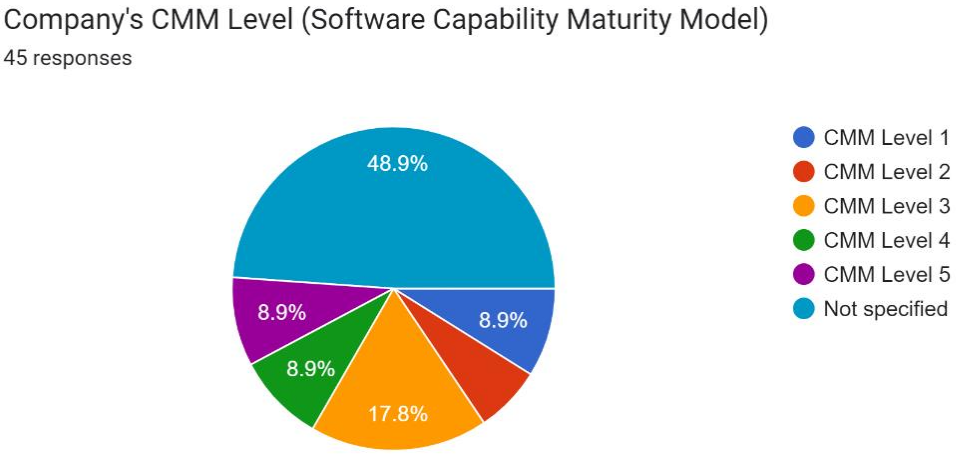
[55] Sinha, A., & Das, P. (2021, September). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. In 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech) (pp. 1-4). IEEE.

[56] Duclervil, S. R., & Liou, J. C. (2019). The study of the effectiveness of the secure software development life-cycle models in it project management. In 16th International Conference on Information Technology-New Generations (ITNG 2019) (pp. 91-96). Springer, Cham.

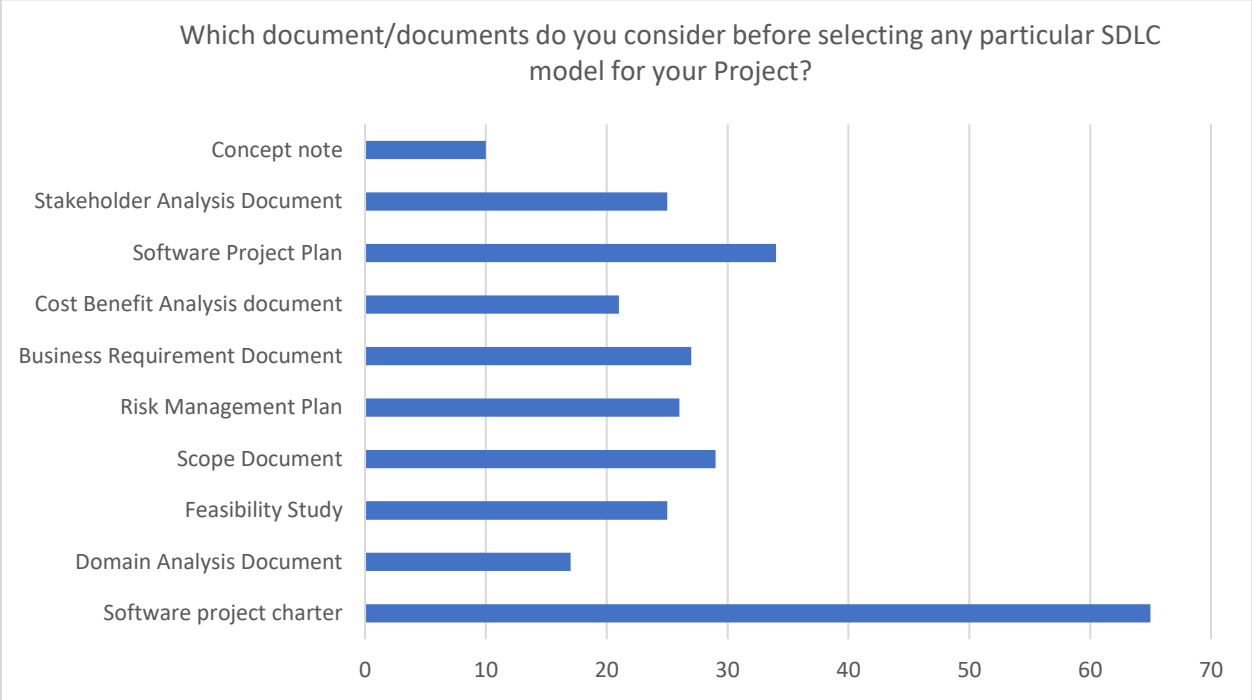
Appendix A



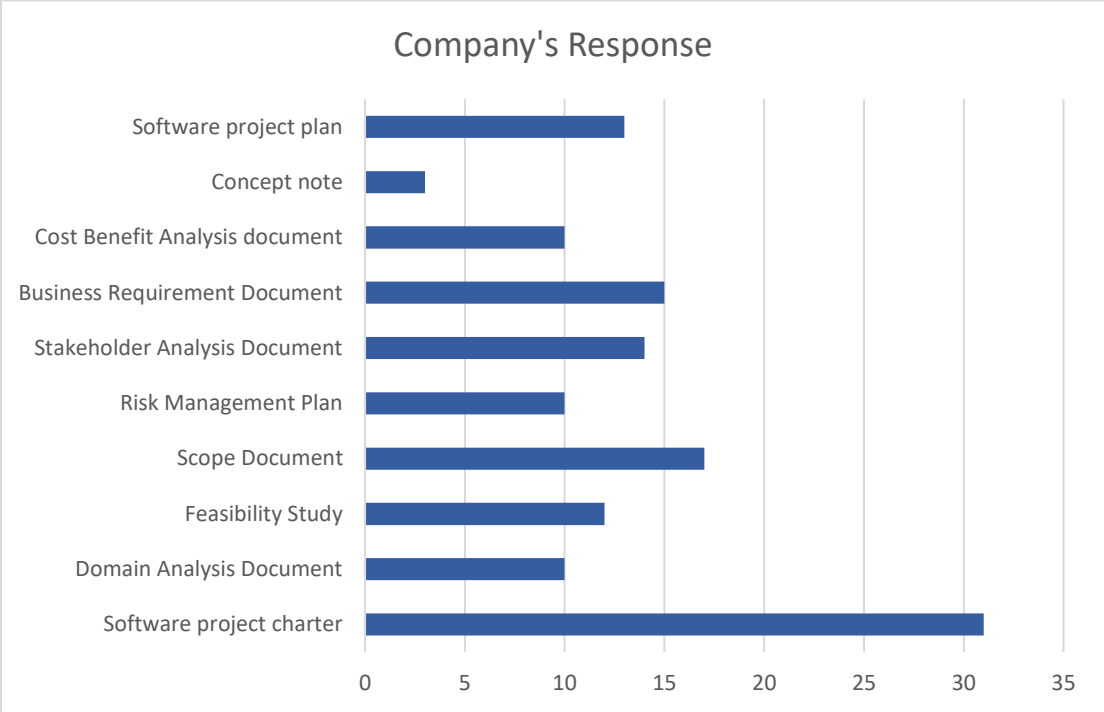
Appendix B


Company's CMM Level (Software Capability Maturity Model)
45 responses
48.9%
8.9%
8.9%
8.9%
17.8%
CMM Level 1
CMM Level 2
CMM Level 3
CMM Level 4
CMM Level 5
Not specified

Appendix C


Which document/documents do you consider before selecting any particular SDLC model for your Project?
Concept note
Stakeholder Analysis Document
Software Project Plan
Cost Benefit Analysis document
Business Requirement Document
Risk Management Plan
Scope Document
Feasibility Study
Domain Analysis Document
Software project charter
0
10
20
30
40
50
60
70

Appendix D


Company's Response
Software project plan
Concept note
Cost Benefit Analysis document
Business Requirement Document
Stakeholder Analysis Document
Risk Management Plan
Scope Document
Feasibility Study
Domain Analysis Document
Software project charter
0
5
10
15
20
25
30
35

Appendix E

NCSAEL

TO WHOM IT MAY CONCERN

It is to certify that **Ms Mahira Gul** Registration no **00000329238** has verified her dataset of SDLC model selection from National Cyber Security Auditing and Evaluation Lab (NCSAEL). The dataset consists of 8 characterisitcs on which the Project SDLC model selection depends. The models that she verified as an output are Waterfall model, Evolutionary model, Incremental model, and Hybrid model. We wish her good luck and bright future in her academic career.

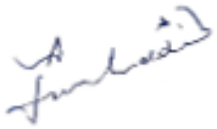
Maliha Safdar
Project Lead
NCSAEL
MCS-NUST

Appendix F

TechEase Solutions Pvt Ltd

TO WHOM IT MAY CONCERN

It is to certify that **Ns Mahira Gul,** Registration no **00000329238,** from **Military college of Signals, NUST,** has verified her dataset of SDLC model selection from TechEase Solution Pvt Ltd. The dataset consists of 8 characteristics on which the Project SDLC model selection depends. The models that she verified as an output are Waterfall model, Evolutionary model, Incremental model, and Hybrid model. We have verified the characteristics and their dependency in the selection of that particular model. We wish her good luck and bright future in her academic career and research work.



Alla ud Din Yousafzai
allauddinyousafxai@gmail.com
CTO - TechEase Solutions
12/06/2022