

People's Democratic Republic Of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry Of Higher Education And Scientific Research
جامعة برج بوعريريج
University Of Bordj Bou Arreridj
Faculty OF Mathematics And Computer Science
IT Department

*Report Of The Final study project
In view of obtaining the Diploma of
IT master
Option: Business Intelligence Engineer*



Theme:

***Analysis Of Teamwork Quality Impact On Software
Development Team Performance***

Realized by :

- ***Helali Khadhir***
- ***Talbi Chahinaz***

Publicly defended on: October th 2020 in front of the jury composed of:

President: Mr:	<i>.A à L'U. El Bachir El Ibrahimi-BBA</i>
Examiner: Mr:	<i>M.A à L'U. El Bachir El Ibrahimi-BBA</i>
Supervisor: Mr:	<i>MAA à L'U. El Bachir El Ibrahimi-BBA</i>

Promotion :2019/2020



ACKNOWLEDGEMENT:

First of all, I would like to thank my supervisor **Dr. Mohamed Amine Beghoura**, my teacher and my role model **Dr. Foudil Belhadj** for the opportunity to conduct my research, also I would like to thank them for their support and assistance.

I want to thank my **mom** and **dad** for caring for me, and their eagerness to study me. Also, **my friends** in university residence, we participated for 5 years and lived the sweet and bitter together

Table of contents:

Acknowledgment	
Table of contents	
Table of figures	
General Introduction	I
Introduction	2
Chapter 1	
1. Introduction	
1.1 Introduction	4
1.2 What is teamwork?.....	4
1.2.1 The effect of communication on teamwork.....	5
1.3 What is Software Engineering.....	5
1.3.1 Definition of Software.....	5
1.3.2 Engineering.....	6
1.3.3 History of software engineering.....	6
1.4 Stages of building the initial system.....	6
1.5 Software Development Life Cycle.....	6
1.5.1 SDLC Activities.....	6
1) Waterfall Model.....	6
2) Iterative Model.....	7
3) Spiral Model.....	7
4) V-Model.....	8
1.6 The difference between programming and software engineering.....	8
1.7 Conclusion.....	9
1.8 Memory plan.....	9
1.8.1 Chapter 1 - Introduction.....	9
1.8.2 Chapter 2- Related work.....	9
1.8.3 Chapter 3- Methodology.....	9
1.8.4 Chapter 4-Impact analysis of teams.....	9
Chapter 2	
2. Related Work	
2.1 Introduction	11
2.2. Related work.....	11
2.3 Contribution.....	13
2.4 Conclusion.....	13
Chapter 3	
3. Methodology	

3.1 Introduction	14
3.2 Git and how it works.....	14
3.3 Git Hub.....	17
3.3.1 Forking a Repo.....	17
3.3.2 Pull Requests	17
3.3.3 Social networking.....	17
3.3.4 Change logs.....	18
3.4 Dataset description.....	18
3.5 Models Study.....	18
3.5.1 Random Forest.....	18
3.5.1.1 Decision trees.....	18
3.5.1.2 Random forest.....	19
3.5.1.2.1 Working of random forest algorithm.....	19
3.5.2 Extreme gradient boosting.....	20
3.5.2.1 Tree boosting algorithm.....	21
3.6 Conclusion.....	21
Chapter 4	
4. Impact analysis of team	
4.1 Introduction	23
4.2 Auxiliary environment to work.....	23
4.3 Implementation of models.....	23
4.4 Evaluation and feedback.....	25
4.4.1 XGBoost model.....	25
4.4.2 Random forest.....	27
4.5 Comparison of the results.....	29
4.5.1 Similarities.....	29
4.5.2 Differences.....	29
4.6 Conclusion.....	30
General Conclusion	
5. Conclusion	32
6. Recommendation	32
7. Future work	32
References	35
Abstract	

Table of figures:

Figure 1: Waterfall model	7
Figure 2: Iterative model	7
Figure 3: Spiral model	8
Figure 4: V_model	8
Figure 5: Storing data as changes to a base version of each file	15
Figure 6: Storing data as snapshots of the project over time	16
Figure 7: Working tree, staging area, and Git directory	16
Figure 8: The structure of decision tree	19
Figure 9: The structure of Random Forests	20
Figure 10: The structure of XGBoost	20
Figure 11: Packages and functions for XGBoost	23
Figure 12: Packages and functions for RF	23
Figure 13: Read data	24
Figure 14: Dropping data	24
Figure 15: Release count attribute	24
Figure 16: Has download attribute	25
Figure 17: The parameters of XGBoost	25
Figure 18: Number of trees classified	26
Figure 19: The score of XGBoost	26
Figure 20: Confusion Matrix of XGBoost	26
Figure 21: The importance features of XGBoost	27
Figure 22: The parameters of Random Forest	27
Figure 23: The Score of Random Forest	28
Figure 24: Confusion Matrix of Random Forest	28
Figure 25: The importance features of Random Forest	28



General Introduction

I. Introduction:

“No wonder we have failed projects, software practitioners cannot tell success From failure.” Robert L Glass.

The study of multi-function teams is still common as teams are used to integrate multifunction expertise to achieve higher performance in innovative projects. A stream of relevant research shows that the difference is associated with only higher performance in highly innovative situations projects. This research depends on the assumption that organizations can freely choose integration mechanisms on a project-by-project basis.

However, this assumption is flawed in cases where Organizations, such as software development laboratories, are committed to a team-based organization designed on a permanent basis. Also software development teams working on projects with varying degrees of innovation.

Today, groups are changing rapidly, as a prerequisite for doing business. However, in this research, we will address the extent to which the quality of teamwork affects the performance of software development, or rather software projects, whereas in the past most studies have called for the importance of teamwork as a basis for software success.

Teamwork stimulates unity in the workplace, promotes friendship and loyalty among team members, creates interrelated relationships and stimulates cooperation in the work environment. Since individuals have different skills, talents and strengths, collaboration motivates each employee to do his/her best and use his/her skills and abilities to achieve goals for the benefit of the team and the company in the end. It also gives an enterprise or company a variety of ideas. This environment allows individuals to share ideas and experiences, which in turn creates a competitive advantage for achieving goals and objectives, and exchanges of views and experiences enhance accountability and help make effective and rapid decisions. Also in the study, which stressed the importance of communication between members of the software team for its success, which considered factors affecting either its success or its loss? While all the studies that have been done before come to our mind as problematic as the wizards affecting do the software development team have.

A wide range of literature suggests importance. As a team, for successful software projects. That is growth. The awareness that good teamwork increases the success of innovative projects raises new questions: How can it be measured ?Why and how is teamwork linked to the success of innovative projects? How strong the relationship between teamwork and different measures of project success is like performance or team satisfaction? This is an article that sets out a comprehensive concept of cooperation in the area of teams, and this is one of the highlights that we will be talking about in the note.

A decorative border in a light orange color, shaped like a scroll. It has rounded corners and a vertical strip on the left side that looks like a scroll's edge. There are three grey circular accents: one at the top left, one at the top right, and one at the bottom right of the scroll's frame.

Chapter 1: Introduction

1.1 Introduction:

“The only truly successful project is the one that delivers what it is supposed to, gets results, and meets stakeholder expectations.” **James P Lewis.**

The aim of this chapter is to introduce how literature defines success Software projects and a detailed definition of teamwork and its relationship to the performance of the software team.

1.2 What is teamwork?

The concept of teamwork depends on the team itself, and in general, it is the process by which team member’s work together collaboratively to achieve a set of goals; That is, the members of the work team will cooperate and use their different individual skills and talents to provide constructive and useful feedback, and teamwork is also known as the work performed. By a group of partners, and each of them performs aspecific individualtask and exerts his best efforts and uses his skills, and this work is to achieve a common goal in which the employee abandoned On personal prominence for the benefit of the company or institution in which he works. Thus, teamwork is based on the division of work and tasks and their coordination among them, while “Cohen (1999) says that teams replace individuals as basic elements in the organization. In this century, the skill of teamwork was taught as an educational concept. Essential in schools, so that students learn appropriate strategies to develop professional skills as part of their educational process” [1].

It is defined in the International Encyclopedia of Social and Behavioral Sciences as: the ability that team members have to work together, communicate effectively, and anticipate each other's requests, In the context of speech, communication is an important element in teamwork Communication is an important part of teamwork. Because it allows the exchange of information, new ideas between people and a focus on effective communication through good listening, which is a way to show respect, then, developing mutual trust within the team environment.

In our research, this we encountered in previous research the importance of teamwork and cycle .team work it was considered an important reason for achieving work goals, and there are many actions that a single person cannot accomplish on time and with the required effectiveness [2]. The following is the importance of teamworkConsolidating social relationships, improving efficiency and productivity, acquiring new skills, diversifying ideas and opinions, improving services and attracting talents [2].

Teamwork represents a skill of success, and it requires focus, clarity, codification, objectivity, avoiding individualism and communication, and aims to develop problems and solve them through problem testing, treatment planning, problem studies and group discussion. Skills for success at work include skills in dealing and communicating with others, [3]establishing good human relationships, and the ability to work as part of a team. This ability became an urgent need to meet the demands of the world of work. Cooperation between individuals (individuals and groups) and working with them has become a necessity for life, whether you use direct or indirect communication skills.... In the context of speech, we can talk about communication and its importance in teamwork.

1.2.1The effect of communication on teamwork:

Communication affects teamwork in positive and negative ways. The quantity and quality of communication within a team and from leadership affects teamwork. The more collaboration your projects require the more assertive and intentional your communication should be. [3] Every member of the team needs to take the initiative to communicate. When a team is not actively communicating, their work is at stake. It is important for everyone to learn how to communicate effectively to improve teamwork.

1.3What is Software Engineering?

Can be defined, as follows is the profession of high-quality software and applications. Software engineering through configuring the program from its stages during problem solving, then designing and writing the program until testing, testing and installing it to implement its own maintenance process, and now we will talk about a historical brief on software engineering.

1.3.1 Definition of software:

Is a description of all the integrated operations that the computer does, such as solving mathematical and statistical problems, in addition to making the necessary correction to the editorial formula and performing the operations requested by the user to the fullest extent. The term software refers to everything that a computer consists of except for the physical components of the computer.

- ✓ Is more than just a program code? A program is an executable code, which serves some computational purpose. Software is considered collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.

- ✓ Software is aintangible thing compared to other products. It a series of thousands or millions of commands that require the computer to perform certain operations such as displaying information, performing calculations, or storing data. This software is the soul of the body in the computer system, and it is constantly expanding and increasing in complexity, requirements and tasks that it performs. As for software engineering, it is a branch of engineering thatis based on a set of foundations and rules aimed at designing and developing programs in abundance and high quality that meet the needs of users. This branch of engineering is distinguished by that it does not need a large capital and therefore the loss is little, unlike the rest of the other branches of engineering. It is not enough to find a good integrated software for one person's work, but rather requires a team of good engineers. It was necessary to find a science concerned with software engineering to lay the foundations and standards that protect this profession from intruders, so that a good program can be distinguished from a good one.

1.3.2 Engineering:

In the other hand, is all about developing products, using well-defined, scientific principles and methods.

1.3.3 History of software engineering:

Software engineering was used as a theoretical concept from time to time in the late 1950s and early 1960s. As for the first official use of this term, it was in a conference held by the Scientific Committee of the North Atlantic Treaty Organization in 1968 on software, and this term has been spreading since then and has met with increasing interest in various aspects. The conference was held to address what is known as the "software crisis", which appeared due to the lack of a methodology of thinking (Software Development Process) when building software, which led to the emergence of many errors during the process of building and maintaining the software, and thus the software has become a lot of time to develop and maintain, and a high financial cost. More than it is estimated, and after enduring the delay in time and exceeding the budget, the software was of poor efficiency in accomplishing the required functions.

1.4 Stages of building the initial system:

In software engineering, building a software system is not just writing code, but rather a production process that has several basic and necessary stages to obtain the product, which is the program at the lowest possible cost and the best possible performance. These stages are called Software Lifecycle, some of which may seem unrelated to programming. There are many visualizations and models in software engineering, that describe the process of producing a program and the steps involved in it. In addition, this cycle is always subject to development, as in addition to the classic courses, the concept of the flexible system appeared (Agile Process), which abandoned the fixed model of the classical system in order to achieve more freedom of movement for the project. The following is a presentation of one of the most famous classic software system life cycles, the waterfall cycle.

1.5 Software Development Life Cycle:

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

1.5.1 SDLC Activities:

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

1) Waterfall Model:

Waterfall model is the simplest model of software development paradigm. All the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

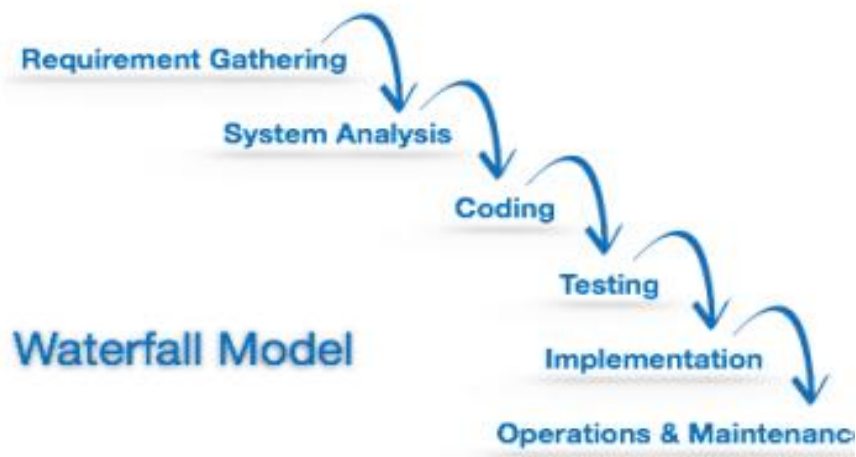


Figure1. Waterfall model

2) Iterative Model:

This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.

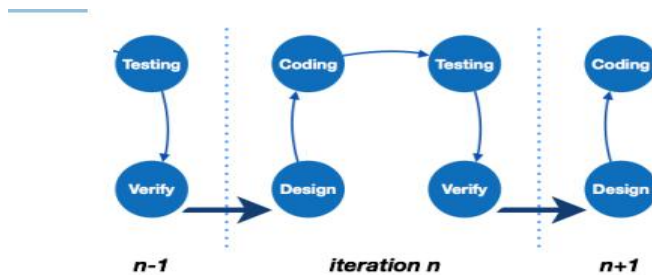


Figure2. Iterative model

3) Spiral Model:

Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combined it with cyclic process (iterative model).

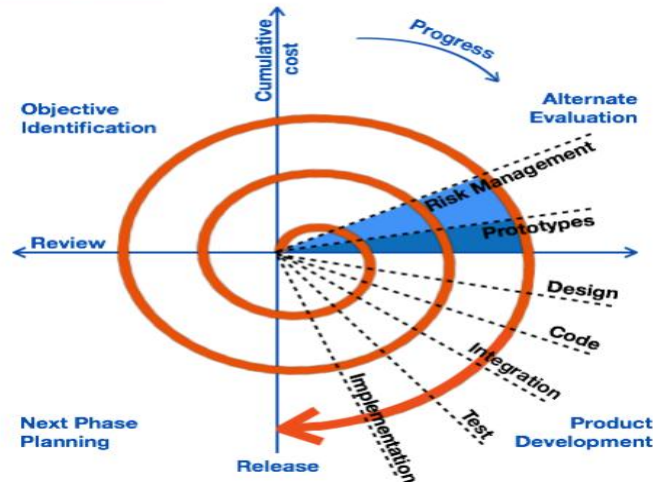


Figure3.Spiral model

Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

4) V – model:

The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is Software Engineering Tutorial 13 found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.

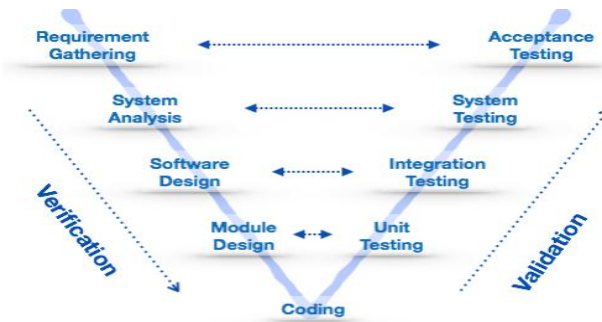


Figure 4.V_model

So these are the software development models and the formats that software engineers use to build a specific program.

1.6The difference between programming and software engineering:

Programming is writing code, which some consider the most important process in building programs. Programming is not concerned with things like the feasibility of the program, the ability of the user to accept it, or even the viability of it. Whereas, software engineering works on building the software system as an integrated project, and studies it from all aspects: programmatic construction, technical support and maintenance, marketing

and sales, development and training on its use, thus it can build large systems for use in the team work system while individual programming is unable to do so. The various fields that are related to software engineering.

1.7 Conclusion:

The purpose of this chapter is to introduce the topic of the master’s thesis. Whereas, this chapter also deals with research objectives, research questions, and in the second chapter we will talk about some previous projects related to the topic of our memo as well as research contribution.

1.8 Memory plan:

1.8.1 Chapter 1: Introduction:

In this chapter, we will deal with the basic concepts, which are teamwork, software engineering, and the relationship between them.

1.8.2 Chapter 2: Related work:

In this chapter, we discussed open source projects that were previously completed and related to the topic of our memo, and we mention some of those studies related to our work.

1.8.3 Chapter 3: Methodology:

In this chapter, we will talk about Git and try to understand how it is work, further, the way that Git see its data and the difference between Git and the other systems. Finally, we see the stream of snapshots method of Git. After that, we will jump into GitHub and we will see their main pages and the main work of it.

We will describe our data, and the different models study.

1.8.4 Chapter 4: Impact analysis of teams:

In this chapter, we will choose the best environment for our work. After that, we will apply our models on our data, and will see the result from different algorithms.

Finally, we will see the comparison of the result.



Chapter 2: Related Work

2.1. Introduction:

After the completion of the presentation of all the basic concepts of our project, we will present in this chapter the findings of previous research and project studies related to the Master's thesis.

Many studies have been conducted in order to fully analyze the impact of teamwork on the performance and quality of software development at the personal and work group levels.

Among the researchers studied, some achieved acceptable results. Here I quote some projects that have inspired me as part of my master's degree to realize my project, which aims to find out the factors affecting the software development team.

2.2. Related work:

We have both Zerina Ibrahim _Md Gapar Md Johar_Nurmi Rafida Abdel Rahman by studying the high quality of practical quality. [4]The project development project was developed in various stages. Teamwork. Continuous resumption of work in the teamwork process. Application development the framework, which is the User Requirements Policy Approach (URS), System Requirements (SRS), Module Planning, Planning Base Data, User Acceptance Test (UAT), and Final Acceptance Test (FAT) for completing phases during development. Our current OS applications are considered on readiness. This research expands the need for findings to empiricism from the commercial, commercial, commercial, and commercial systems. The validation process is teamwork at work in order to work on the way to work. This research found that it greatly supports the skill of teamwork in the OS the new proposed ASDF method is an important part of their present research. With regard to the system development objectives, a verification of the main problem of extending the project schedule is reached demonstrate and accomplish teamwork and reduce significant team cooperation and understanding in system development. Therefore, determine the project development in progress Equalizer in Oder to control and increase the quality of skills in cooperation and understanding of teamwork in system development. The current survey found that TWQ and team performance were most related when team members evaluated these two concepts. Moreover, the relationship between TWQ and team member success -Satisfaction with work and learning - loneliness. One explanation is team members regard TWQ and team members as success as concepts that cannot be distinguished, this is what the authors of the research reached

Another research [7], Junaid Maqsood, Iman Eshraghi, and Syed Sarmad Ali have conducted a research essentially identifying success and unsuccessful projects on GitHub from a sample of 5,000 randomly selected projects in a number of randomly selected languages (Java,PHP, JavaScript, C # / C, HTML). We have selected 1000 projects

For each of these languages through the publicly available GitHub API has optimized their dataset and applied various machine-learning algorithms to achieve their goal. They initially executed several queries against the dataset and found meaningful relationships and correlations between some of the traits brought in that had an extension of effect on the popularity of these projects. Later we can develop an application that will determine the success or failure of an open source project file. the most popular project currently widely used repository and hosting is GitHub. Having Git as a primary component of its architecture, GitHub makes the ideal place to conduct such research. In their work, they randomly selected projects from a number of different languages, which gave them the opportunity to not only

touch the tip of the iceberg for open source projects, but also touch the gray area, by randomly selecting the project and they applied an appropriate algorithm to extract the data. They developed a program that explains the hidden relationship structure from their training data model. As determined whether, a particular project has more chances of success, in the way of development or vice versa.

In this, research [5] authors have examined projects' success by two measures of popularity and developer activity for 283 OSS projects from Source Forge over a period of 3 years between 2000 and 2004. They have restricted their sample to projects written wholly or partially in C++ programming language and designed for Microsoft Windows operating system. They have defined two sets of intrinsic and extrinsic factors involved as cues. Intrinsic cues represent product-related attributes. Extrinsic cues on the other hand are not part of the physical product. Intrinsic cues have a major role in product selection decision-making process, however consumers depend more on extrinsic cues for risk reduction. Researchers made hypotheses about the success of open source projects and tried to evaluate, the ones that hold true. They originate that project licensing has a major impact on the popularity and developer's activity. Commercial exploitation is one of the fear that stops developer to contribute towards project. The research also identified that the projects, which have a larger user base, tends to be more popular, however negative point founds in developers at a certain point, when it increases. Results [7] show the popularity in the number of language translations of a project (mostly in regards with the UI). Another finding is the positive correlation between the correct delegation of responsibilities to developers who are capable of doing it and the higher developer activity as a result. Since major releases are more stable, they have found out that developers prefer major releases over minor ones. Their results show that different factors over time can have different effects on the success of these projects.

In another research [6] authors have studied the impact of App churn on the App success through the analysis of 154 Android Apps that have a total of 1.2k releases. They have found that high App churn leads to lower user ratings and less stable apps will result in lower ratings as well. They have also investigated the link between how frequently API classes and methods were changed by App developers relative to the amount of discussion of these code elements on Stack Overflow.

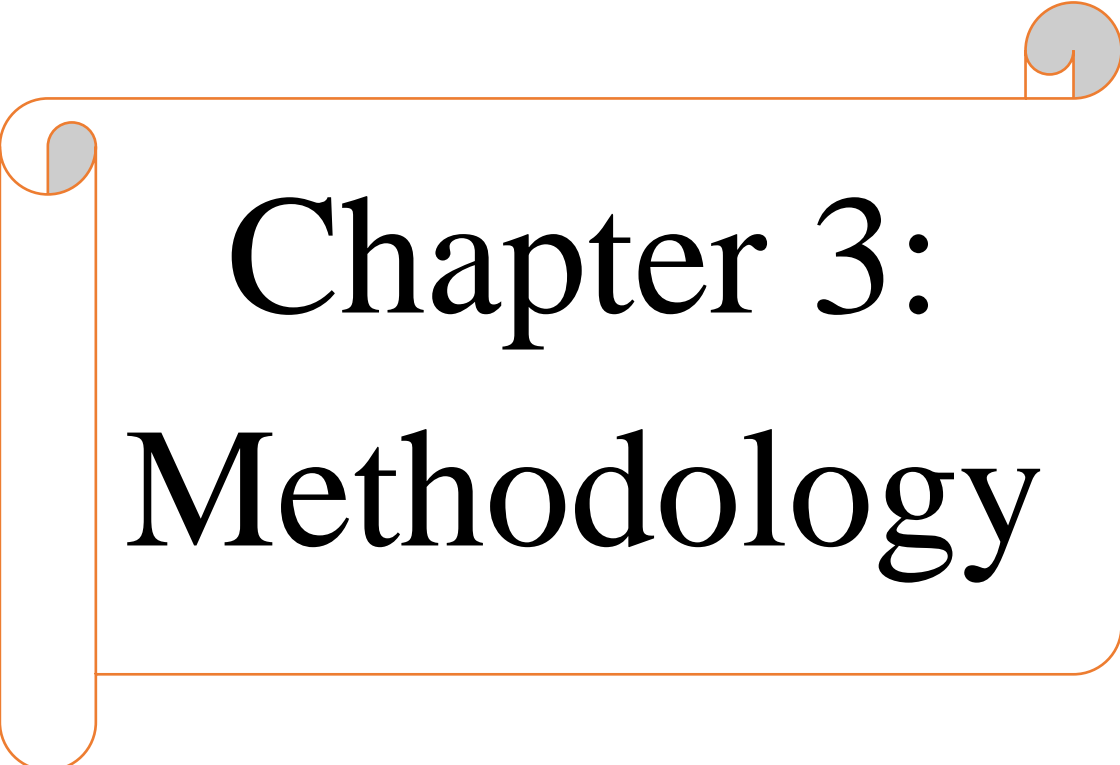
In this paper [8] authors have conducted a research, in order to find the success factors of 5000 most-downloaded Open Source projects at sourceforge.net. They have defined a set of parameters including rank, download, activity, members, translation, operating systems, license, programming languages. They have applied the Association Rule Data-mining technique to find rules determining success factor using Weka Data-mining tool. They have concluded that increasing activity of project, along with the high rank of it, the higher number of participants in the project and the age of the project are all influencing factors in success of open source projects.

2.3. Contribution:

Regardless of previous research on studying and analyzing, the success or failure factors of software project. we have seen in every research conducted by researchers on the latter. The factors affecting its success have not identified. We have chosen in our work a database for this project, and we have implemented an analysis and classification program through an existing database. From them, we extracted some of the factors that often affect the success or loss of the project we have used in our research the method of analysis and classification using Python program we used the random forest algorithm as our first classifier .In addition to using, the Expression Blend included in the Visual Studio package that was used specifically for designing a simple graphical user interface. the goal of our research is to design a predictive algorithm by using the statistics of any project to predict whether it is successful or losing and to provide feedback on the projects.

2.4 Conclusion:

This chapter introduces the existing models, related to the research thesis. And the research contribution we made, and in the next chapter we'll talk about Git and Github, more deeply as we define the relationship between Git and Github and describe our dataset related to teamwork. We will look at some of the methods and algorithms that we use. We depended on it to reach our goal.



Chapter 3: Methodology

3. Methodology:

3.1 Introduction:

In this chapter we will take a step further with Git and Github, we look deeper and try to have detailed information that lead us to understand the Git and how it is working after that we define the relation between Git and Github next we investigate Github and their features.

Our aim was to identify the most features that impact of teamwork' s software projects, so we need a dataset that have teamworks' data of each project. in section 3.3 we will show you our dataset description. Finally we will see some methods and algorithms that lead us to a better understanding.

3.2 Git and how it works:

“Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.” [9]

“Git is a distributed version-control system for tracking changes in source code during software development It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development since 2005, Junio Hamano has been the core maintainer. As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full- fledged repository with complete history and full version-tracking abilities, independent of network access or a central server Git is free and open-source software distributed under GNU General Public License Version 2.”[10]

Systems like (CVS, Subversion, Perforce, Bazaar, and so on) store information as a list of file-based changes. And think of the information they store as a set of files and the changes made to each file over time

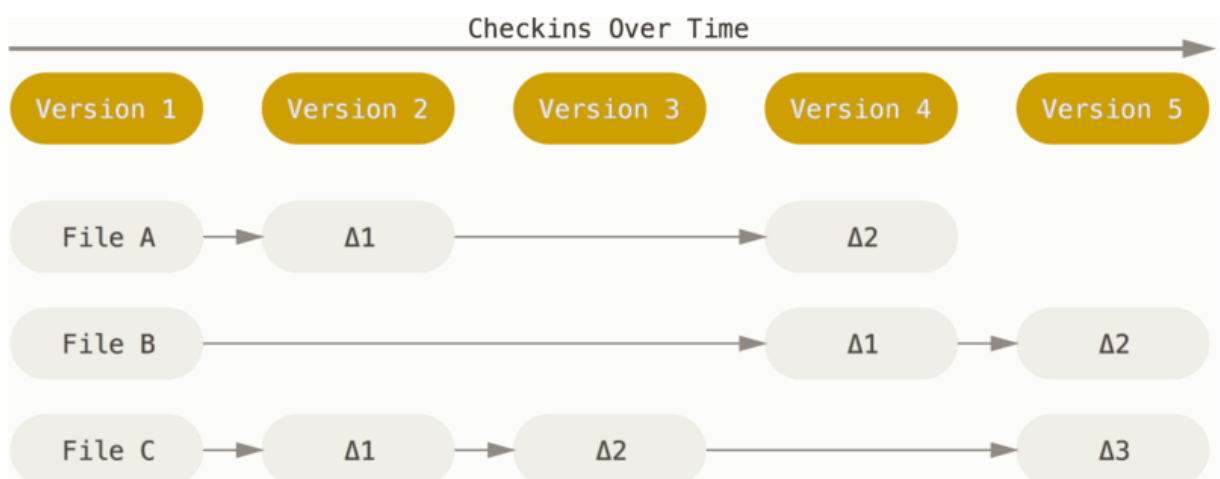


Figure5. Storing data as changes to a base version of each file

Git doesn't think of or store its data this way. Instead, Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a **stream of snapshots**.

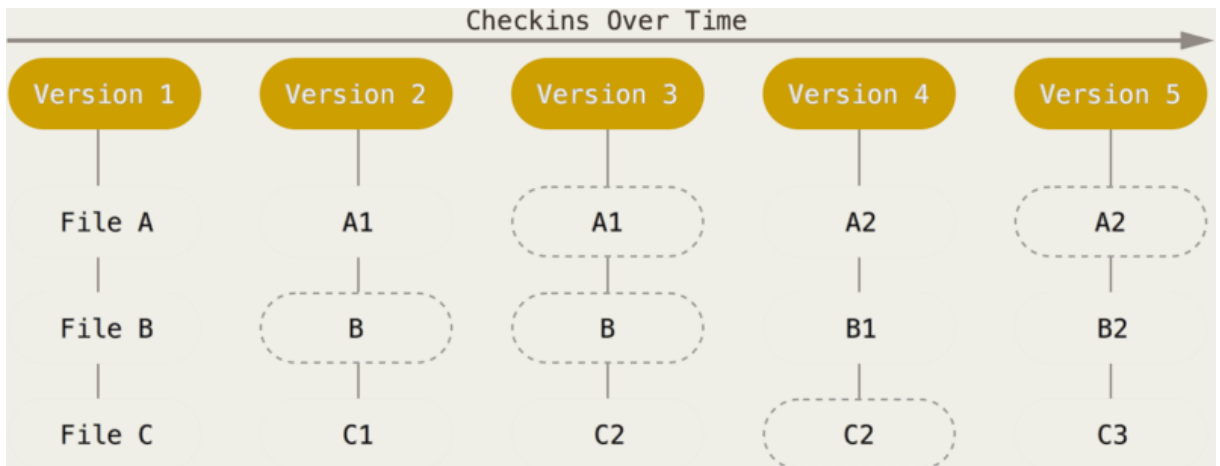


Figure6. Storing data as snapshots of the project over time

Git has three main states that your files can reside in: **modified**, **staged**, and **committed**: This leads us to the three main sections of a Git project: the working tree, the staging area, and the Git directory.

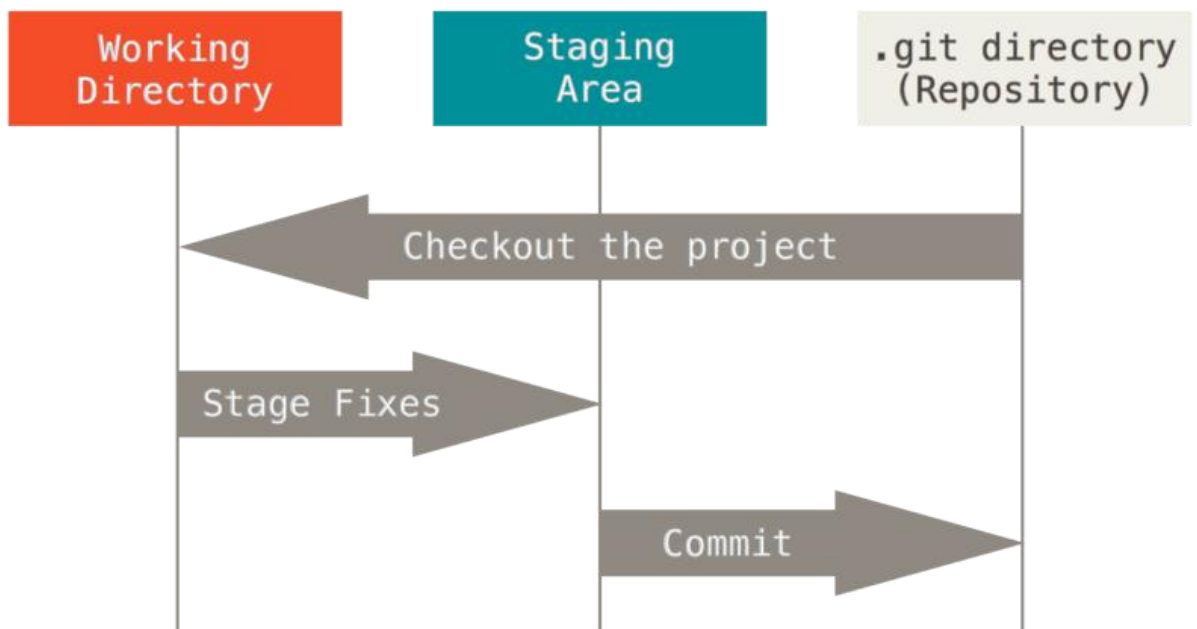


Figure7. Working tree, staging area, and Git directory

3.3 GitHub:

GitHub is a web-based version-control and collaboration platform for software developers. Microsoft, the biggest single contributor to GitHub, initiated an acquisition of GitHub for \$7.5 billion in June, 2018. GitHub, which is delivered through a software-as-a-service (SaaS) business model, was started in 2008 and was founded on Git, an open source code management system created by Linus Torvalds to make software builds faster.

Git is used to store the source code for a project and track the complete history of all changes to that code. It allows developers to collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers. GitHub allows developers to change, adapt and improve software from its public repositories for free, but it charges for private repositories, offering various paid plans. Each public or private repository contains all of a project's files, as well as each file's revision history. Repositories can have multiple collaborators and can be either public or private(**Posted by: Margaret Rouse**).

3.3.1 Forking a Repo:

“Forking” is when you create a new project based off of another project that already exists. This is an amazing feature that vastly encourages the further development of programs and other projects. If you find a project on GitHub that you'd like to contribute to, you can fork the repo, make the changes you'd like, and release the revised project as a new repo. If the original repository that you forked to create your new project gets updated, you can easily add those updates to your current fork.

3.3.2 Pull Requests:

You've forked a repository, made a great revision to the project, and want it to be recognized by the original developers—maybe even included in the official project/repository. You can do so by creating a pull request. The authors of the original repository can see your work, and then choose whether or not to accept it into the official project. Whenever you issue a pull request, GitHub provides a perfect medium for you and the main project's maintainer to communicate.

3.3.3 Social networking:

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Each user on GitHub has their own profile that acts like a resume of sorts, showing your past work and contributions to other projects via pull requests.

Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward. Before the advent of GitHub, developers interested in contributing to a project would usually need to find some means of contacting the authors—probably by email—and then convince them that they can be trusted and their contribution is legit.

3.3.4 Changelogs:

When multiple people collaborate on a project, it's hard to keep track revisions—who changed what, when, and where those files are stored—. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.

3.4 Dataset description:

To create any sort of algorithm or model we were need a good dataset that let us know how the software teams work in their projects. We wanted to get a huge dataset from GitHub GHTorent contained a huge number of GitHub project's metadata [11]. After further researches, when we was searching for our data, we find that almost data does not contain the attributes that might help us take our research a step further, and contained different data about what we want..

We find a dataset [12] it is available and free for researchers and student in their site [13]. This dataset contained five randomly selected languages (C and C#, Java, HTML, JavaScript and PHP) from the popular list projects of GitHub and then for each language has 1000 randomly selected projects from GHTorent big data. We choose 15 different attributes including the Name of the project, Author, Language, URL along with the sum of its contributes, commits, stars, forks, branches, watchers, pull-requests, total issues, open issues, releases number and the Boolean attribute telling if it had any download file specified or not. The collected dataset was then formatted in file CSV.

3.5 Models Study:

When we finished going deeper through the dataset' depths and discovered a lot of things which are amazing notes we can start with it, we are now ready to apply our model.

3.5.1 Random Forest:

3.5.1.1 Decision Trees:

To understand the random forest model, we must first learn about decision trees, the basic building block of a random forest. We all use decision trees in our daily life. Decision tree is one of the most popular machine learning algorithms used all along, Decision trees are used for both classification and regression problems.in this paper we talk about classification problems. [14]

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.[15]

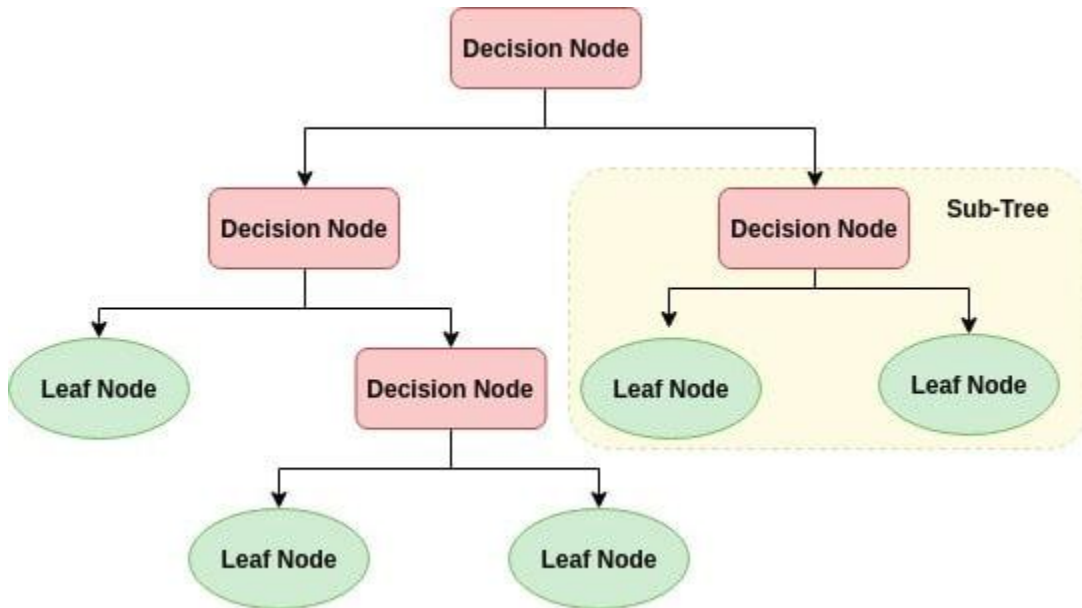


Figure8.The structure of decision tree

3.5.1.2 Random Forest:

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.[16]

3.5.1.2.1 Working of Random Forest Algorithm:

We can understand the working of Random Forest algorithm with the help of following steps:

- **Step 1** – First, start with the selection of random samples from a given dataset.
- **Step 2** – Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** – In this step, voting will be performed for every predicted result.
- **Step 4** – At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working:

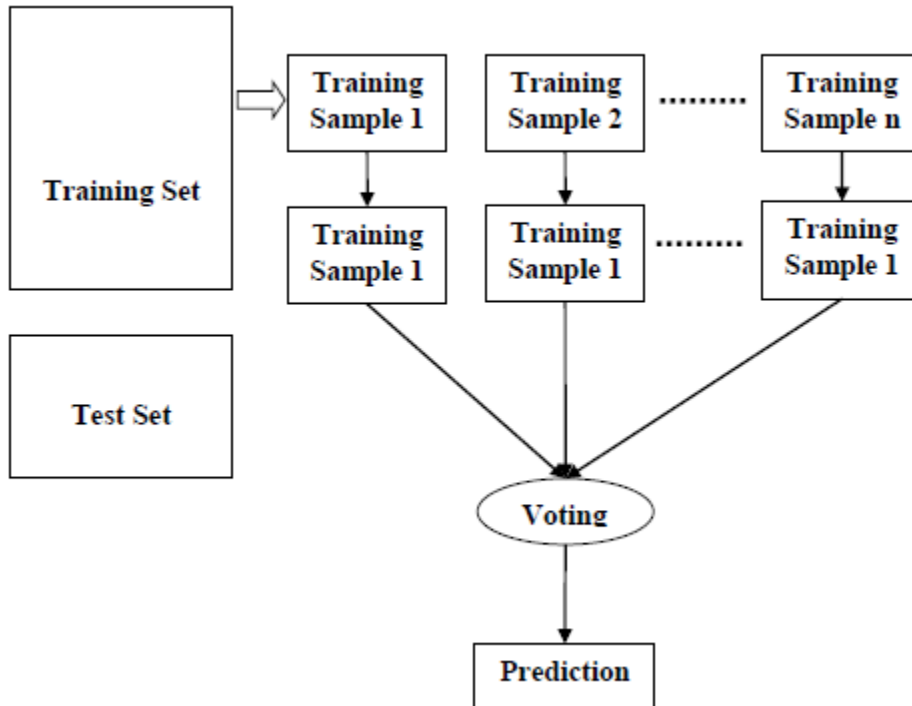


Figure9.The structure of Random Forests

3.5.2 Extreme Gradient Boosting:

XGBoost stands for “Extreme Gradient Boosting”, where the term “Gradient Boosting” originates from the paper *Greedy Function Approximation: A Gradient Boosting Machine*, by Friedman.

Ever since its introduction in 2014, XGBoost has been lauded as the holy grail of machine learning hackathons and competitions. From predicting ad click-through rates to classifying high energy physics events, XGBoost has proved its mettle in terms of performance – and speed.

Gradient boosting is an important tool in the field of supervised learning, providing state-of-the-art performance on classification, regression and ranking tasks. XGBoost is an implementation of a generalised gradient boosting algorithm that has become a tool of choice in machine learning competitions. This is due to its excellent predictive performance, highly optimised multicore and distributed machine implementation and the ability to handle sparse data.

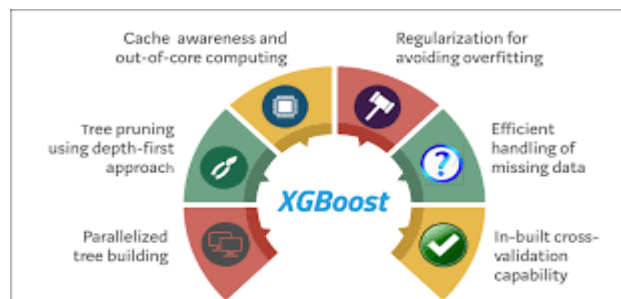


Figure10.The structure of XGBoost

3.5.2.1 Tree boosting algorithm:

XGBoost is a supervised learning algorithm that implements a process called boosting to yield accurate models. Supervised learning refers to the task of inferring a predictive model from a set of labelled training examples. This predictive model can then be applied to new unseen examples. The inputs to the algorithm are pairs of training examples $(x \rightarrow 0, y_0), (x \rightarrow 1, y_1) \dots (x \rightarrow n, y_n)$ where $x \rightarrow$ is a vector of features describing the example and y is its label. Supervised learning can be thought of as learning a function $F(x \rightarrow) = y$ that will correctly label new input instances.

Supervised learning may be used to solve classification or regression problems. In classification problems the label y takes a discrete (categorical) value. For example, we may wish to predict if a manufacturing defect occurs or does not occur based on attributes recorded from the manufacturing process, such as temperature or time, that are represented in $x \rightarrow$. In regression problems the target label y takes a continuous value. This can be used to frame a problem such as predicting temperature or humidity on a given day.

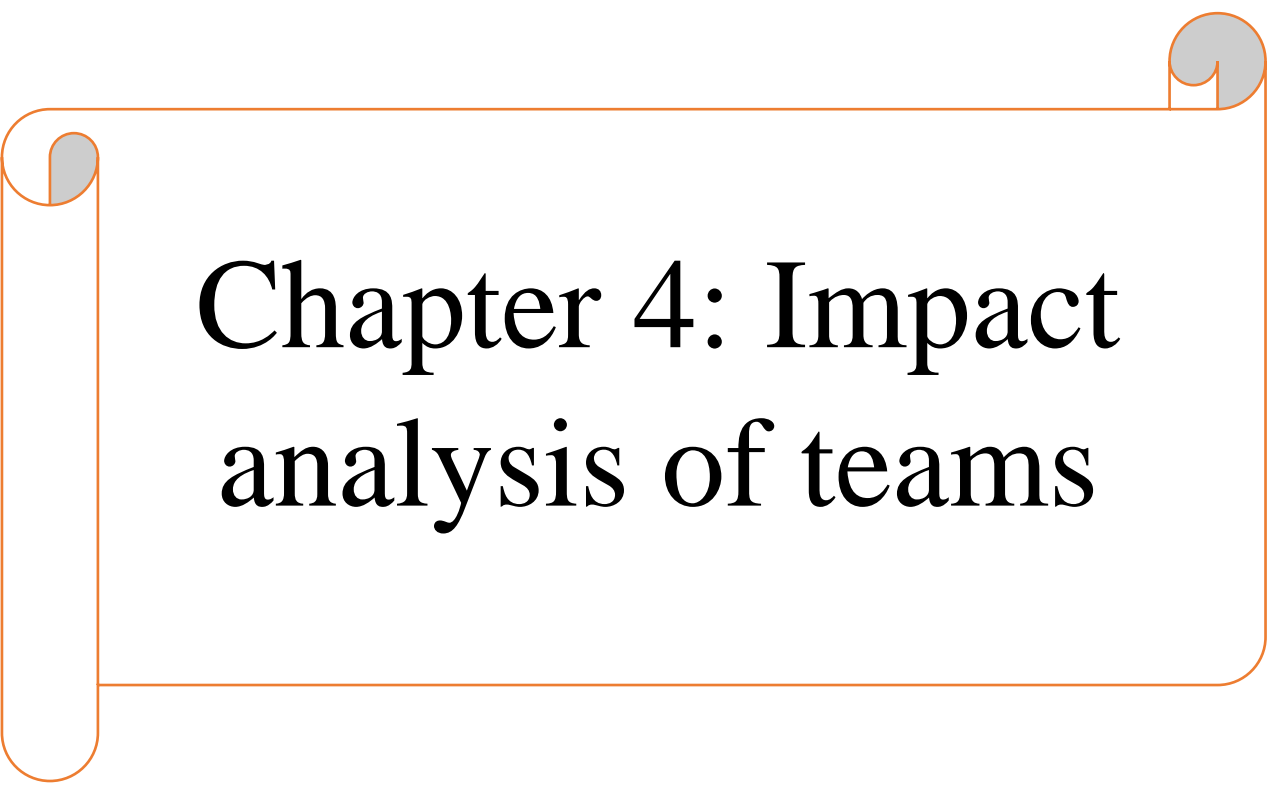
XGBoost is at its core a decision tree boosting algorithm. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model attempting to correct for the deficiencies in the previous model. In tree boosting each new model that is added to the ensemble is a decision tree.

3.6 Conclusion:

This chapter provided different research methods, different strategies of inquiry, and a discussion of how to choose research approach and strategy of inquiry. Based on the discussion, the design of the research in this master thesis was presented.

The research in this master thesis used a classification research approach as strategy of inquiry and identifies the success projects factors. For data collection, all the projects and their information was collected from Github web site, the missing data were verified to get started with the algorithms.

For methods used, two methods were used: Random Forest, XGBoost.

A decorative border in a light orange color, resembling a scroll. It has rounded corners and small circular tabs at the top-left, top-right, and bottom-left corners.

Chapter 4: Impact analysis of teams

4. Impact analysis of teams:

4.1 Introduction:

To achieve our aim, which is: figure out the most important success factors that affected on the open source software success, among GitHub projects we analyzed the projects and try to find the most affected attribute using different tools?

4.2 Auxiliary environment to work:

First move, our data was prepared for classification, we decide that we can use python algorithm for our research, because of his simplex and it support the classification, after that we chose Anaconda environment to be our main program which allow us to choose from different platforms to keep working with.

After we have looking carefully about the best platform for our research we found that Jupyter NoteBook is the best solution for our work aim. It has almost the packages and the function that we need so we choose it for this reason.

4.3 Implementation of models:

Our dataset has 15 attributes including the Author, Language, Project Name and URL, which we don't need it for our classification because they have no effect to the project success so we need to move them out.

After that we check if we have any missing data, lucky we, there is no messing data. So we are ready to apply our algorithms.

For both XGBoost and Random Forest, we need to call a few packages, each function has a role for our work process. As we see in figures below the packages and functions we need it.

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score, roc_auc_score, make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
import pydotplus
```

Figur11. Packages and functions for XGBoost

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

Figur12. Packages and functions for RF

We create a variable to read our data into, for this process we apply pandas packages like we see below:

```
df = pd.read_csv(r"C:\Users\Lenovo\Downloads\DataSet(C).csv")
df.head()
```

	Authors	ProjectName	Language	Contributers	Commits	Stars	Forks	Branches	Watchers	PullRequests	TotalIssues	OpenIssues	HasDownloads	F
0	glock45	iOS-Hierarchy-Viewer	C	4	94	1213	136	1	61	5	25	9	True	
1	Xfennec	progress	C	12	97	2668	150	1	106	28	58	13	True	
2	keendreams	keen	C	1	8	1592	159	4	105	1	3	1	True	
3	toland	qlmarkdown	C	22	94	1591	90	1	69	20	57	24	True	
4	xoreaxeaxeax	movfuscator	C	1	33	1204	63	1	77	1	5	2	True	

Figur13. Read data

After we drop the bad attributes our data is looking good to start:

```
df.drop(['Authors', 'ProjectName', 'Language', 'URL'], axis=1, inplace=True)
df.head()
```

	Contributers	Commits	Stars	Forks	Branches	Watchers	PullRequests	TotalIssues	OpenIssues	HasDownloads	ReleaseCount
0	4	94	1213	136	1	61	5	25	9	True	3
1	12	97	2668	150	1	106	28	58	13	True	0
2	1	8	1592	159	4	105	1	3	1	True	0
3	22	94	1591	90	1	69	20	57	24	True	3
4	1	33	1204	63	1	77	1	5	2	True	0

Figur14. Dropping data

After that, we need to clean our data. We have the Release Count need to be fixed, if any team don't have a release then we fix 0 to their release count else if they have more than 1 release we fix it 1 we consider that 1 value refer to success team because they reach their aim and get a release.

Also, Has Download attribute which has a Boolean values (True, False) we turn it into binary (1, 0) values because this easy for the python and the algorithms to understand.

```
df.loc[(df['ReleaseCount'] >= 1), 'ReleaseCount'] = 1
df.loc[(df['ReleaseCount'] == 0), 'ReleaseCount'] = 0
df.head()
```

	Contributers	Commits	Stars	Forks	Branches	Watchers	PullRequests	TotalIssues	OpenIssues	HasDownloads	ReleaseCount
0	4	94	1213	136	1	61	5	25	9	True	1
1	12	97	2668	150	1	106	28	58	13	True	0
2	1	8	1592	159	4	105	1	3	1	True	0
3	22	94	1591	90	1	69	20	57	24	True	1
4	1	33	1204	63	1	77	1	5	2	True	0

```
df['ReleaseCount'] = pd.to_numeric(df['ReleaseCount'])
```

Figur15. Release count attribute

```
df.loc[(df['HasDownloads'] == True), 'HasDownloads'] = 1
df.loc[(df['HasDownloads'] == False), 'HasDownloads'] = 0
df.head()
```

	Contributors	Commits	Stars	Forks	Branches	Watchers	PullRequests	TotalIssues	OpenIssues	HasDownloads	ReleaseCount
0	4	94	1213	136	1	61	5	25	9	1	1
1	12	97	2668	150	1	106	28	58	13	1	0
2	1	8	1592	159	4	105	1	3	1	1	0
3	22	94	1591	90	1	69	20	57	24	1	1
4	1	33	1204	63	1	77	1	5	2	1	0

```
df['HasDownloads'] = pd.to_numeric(df['HasDownloads'])
```

Figure16. Has download attribute

4.4 Evaluation and feedback:

In this section we will apply our algorithms.

4.4.1 XGBoost model:

First hand, we applied the XGBoost model on our dataset with different parameters until we get a higher score:

```
clf_xgb = xgb.XGBClassifier(Objective='binary:logistic',
                            missing=None,
                            seed=42,
                            max_depth = 4 ,
                            learning_rate = 0.01,
                            gamma = 0.25,
                            reg_lambda = 1.0,
                            scale_pos_weight = 1,
                            subsample = 0.9,
                            colsample_bytree=0.5)

clf_xgb.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds = 10,
            eval_metric='aucpr',
            eval_set=[(X_test , y_test)])
```

Figure17.The parameters of XGBoost

This classification builds a 15 tree:

```
[0] validation_0-aucpr:0.401439
Will train until validation_0-aucpr hasn't improved in 10 rounds.
[1] validation_0-aucpr:0.42236
[2] validation_0-aucpr:0.421917
[3] validation_0-aucpr:0.388337
[4] validation_0-aucpr:0.386172
[5] validation_0-aucpr:0.415721
[6] validation_0-aucpr:0.418897
[7] validation_0-aucpr:0.437358
[8] validation_0-aucpr:0.432622
[9] validation_0-aucpr:0.43388
[10] validation_0-aucpr:0.441262
[11] validation_0-aucpr:0.437321
[12] validation_0-aucpr:0.43663
[13] validation_0-aucpr:0.439146
[14] validation_0-aucpr:0.442624
[15] validation_0-aucpr:0.454235
[16] validation_0-aucpr:0.450516
[17] validation_0-aucpr:0.448473
[18] validation_0-aucpr:0.448236
[19] validation_0-aucpr:0.44897
[20] validation_0-aucpr:0.449612
[21] validation_0-aucpr:0.449578
[22] validation_0-aucpr:0.450699
[23] validation_0-aucpr:0.451792
[24] validation_0-aucpr:0.440372
[25] validation_0-aucpr:0.446059
Stopping. Best iteration:
[15] validation_0-aucpr:0.454235
```

Figure18. Number of trees classified

With a score equal to 74%

```
print("Accuracy= ",metrics.accuracy_score(y_test, prediction_test))
Accuracy= 0.74
```

Figure19.The score of XGBoost

Finally our objective is the answer of the question which attribute has the most affected on the success of the software project?

To answer this question we need to look at the Releases attribute, we consider that any project has more than one release is a successful project. Furthermore, we split our data into two parts the first for training set the second for testing data. Our best accuracy was 74%

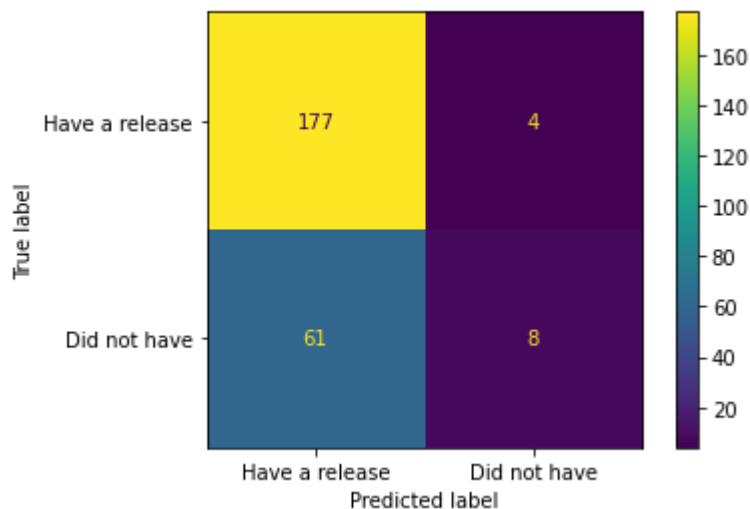


Figure20.Confusion Matrix of XGBoost

Among our testing data 200 projects, the matrix confusion shows that XGBoost classify 177 projects as they have a release and they did have. Also 8 projects didn't have a release and this is true, it consider that 61 projects have releases the fact they didn't have a release and 4 projects didn't have a release which is a not true.

So, we have almost 10 attribute we would like to know which is the most affected attribute ? For that we let the XGBoost tell us

```
feature_list = list(X.columns)
feature_imp = pd.Series(clf_xgb.feature_importances_,
                       index = feature_list).sort_values(ascending=False)
print(feature_imp)
```

TotalIssues	0.250232
PullRequests	0.172247
Contributors	0.125176
Commits	0.107895
OpenIssues	0.082836
Watchers	0.079456
Stars	0.079008
Branches	0.061527
Forks	0.041623
HasDownloads	0.000000
dtype:	float32

Figure21.The importance features of XGBoost

In the figure above we see that both XGBoost model classify the Total Issues as the highest affected attribute with 0.25.

We note that Pull Request, Commits, and Contributors attributes have classified as they have high affect on the success factor,

Otherwise, numbers of open issues, watchers, stars, branches and forks have a low affect on the success factor.

Finally, the download attribute has no affect on the success factor.

4.4.2 Random Forest:

We applied the random forest classifier model on our data as we see the parameters below:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 100,
                             random_state=42,
                             max_depth = 10,
                             min_samples_split = 90,
                             min_samples_leaf= 3,
                             oob_score = True,
                             criterion = "gini",
                             min_weight_fraction_leaf = 0.0,
                             max_features = "auto",
                             max_leaf_nodes = 12,
                             min_impurity_decrease = 0.0,
                             min_impurity_split = None,
                             bootstrap = True,
                             verbose = 0,
                             warm_start = True,
                             class_weight = None,
                             ccp_alpha = 0.0,
                             max_samples = None)

model.fit(X_train, Y_train)
```

Figure22.The parameters of Random Forest

This classification gives us score equal to 75.5%

```
print("Accuracy= ",metrics.accuracy_score(Y_test, prediction_test))
Accuracy= 0.755
```

Figure23.The score of Random Forest

The confusion matrix as we see:

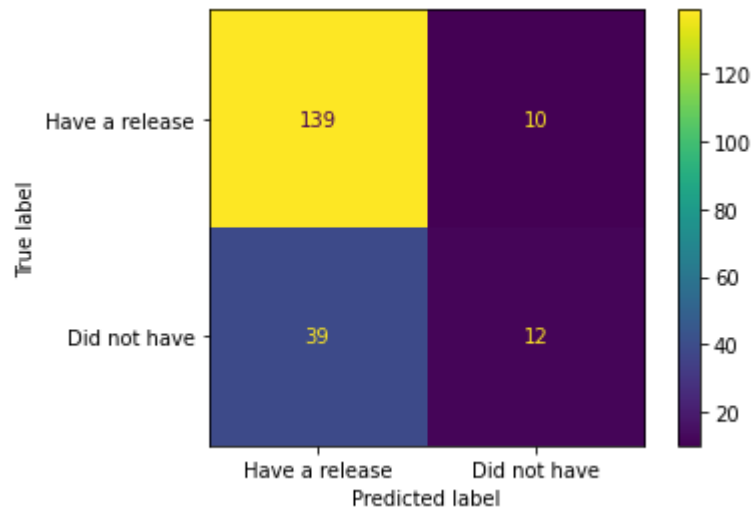


Figure24.Confusion Matrix of Random Forest

The random forest model has classify 139 projects as they have release which is true, also 12 projects they didn't have one, in the other hand 39 projects classified as they have a release but in true data they don't have it, furthermore 10 projects which they have a release classified as they don't have it from our testing data 200 projects.

As usual we need to figure out which is the most imported attribute on the success of the software projects we found that:

```
feature_list = list(X.columns)
feature_inp = pd.Series(model.feature_importances_,
                        index = feature_list).sort_values(ascending=False)
print(feature_inp)
```

```
TotalIssues      0.208289
Commits           0.140878
PullRequests      0.118670
Stars             0.112155
Forks            0.110634
Watchers          0.101589
OpenIssues        0.085636
Contributors      0.079001
Branches          0.041584
HasDownloads      0.001566
dtype: float64
```

Figure25.The importance features of Random Forest

In the figure above we see that both Random forest model classify the Total Issues as the highest affected attribute.

We note that both Pull Request, Commits, Stars, Forks and Watchers attribute classified as they have high affect on the success factor,

Otherwise, numbers of contributors, branches and open issues have a low affect on the success factor.

Finally, the download attribute has no affect on the success factor.

4.5 Comparison of the results:

To get better results, we think that we can make a comparison between our feedbacks, so in this section we try discover the similarities and the differences between the feedbacks.

4.5.1 Similarities:

Both result show that the Total Issues is the most affected attribute and this is a logical conclusion, because when a team works on a project it's very common that their members have issues, in github for example have a page called issues that let any member to define his/her issue and this what we called an open issue the manger of the team read the issue and he must figure out the problem, and he can help (directly, indirectly) the members and this what github called issue closed.

Also, any member of the team can help his team member, it is true that he lose some time on his main work but he win much time for the project and the team progress. This what we called collaboration.

Second place we see that both algorithms consider Pull Request and Commits have a high affected on the success of teams, the frequency of the commits affect on the number of the pull request, if number of commit increase then the number of the pull request also increase, the opposite is true, instead of a team do all of the work and commit it one time it is better of them commit each part of their work, and this cost the increase of the pull request number.

In the third place we note that Open Issues and Branches considered as they have low importance on the project release, like we knew the issues has an important value on the success of the project, github let any user to open an issue if he is a member or not, the manager of the project see the issues and he decides if he must work on it or not, the most of the open issues came from users not from members and this has low affect on the project release. And the number of branches also has different values from a project to another so it has low importance.

Finally, Has Download the project or not has no affect at all because it simple if you need a project you can download it.

4.5.2 Differences:

There are differences in the classification of the algorithms, the Stars, Forks, Contributors and Watchers each model has his word to say.

XGBoost model consider Contributors have a high affect on the success factor .Stars, Forks and Watchers are considered as they have a low affect. But the random forest see that number of stars, forks and watchers has a high affect on the project success also the contributors considered as they have a low important in this model.

4.6 Conclusion:

This chapter presented the results and a discussion of the findings in relation to the research objectives of this study. The first research objective was to identify success attributes for software projects and products. The identified success attributes are:

First place:

- Total issues
- Commits
- Pull request

Second:

- Stars
- Forks
- Branches
- Watchers
- Contributors

The second research objective was to identify success factors for the software projects. The identified success factors are:

- Good Communication
- Collaboration between members
- An experienced leader
- Time response
- Problem attention
- Problem definition



Conclusion

5. Conclusion:

Teamwork is increasingly applied in many organizations in an effort to improve performance, yet empirical evidence demonstrating the relationship between team effectiveness and project success is scarce. Consequently, this study has undertaken an empirical assessment of the linkages between team effectiveness and project success on Github's open software projects.

From a sample of 5000 projects on GitHub, This study has provided an insight into the various factors that affect teamwork and project success. The key conclusions of the study are: time response, collaboration, problem attention, clear processes and roles ensure that projects are done right; and leadership competences correlate directly with project success. Without these factors, it is highly likely that an application of teamwork will be counterproductive.

From the results of the XGBoost and Random forest we found that number of stars of the project is linearly correlated with number of forks and watchers in these projects. Number of Commits is linearly correlated with number of pull request also we found that the frequency of commits and number of pull request affect on the project success. Also we can conclude that the download of the project has no affect at all on the project success.

Finally, it was broadly perceived among the sample that potential impacts of teamwork were significantly greater on the success of the project and that is why complex projects today utilize teams as part of the project management.

6. Recommendation:

Based on the results of the algorithms and the testing of the hypotheses, the authors recommend the following:

- It is recommended that having well defined and realistic goals, roles, issues, responsibilities and appropriate leadership are necessary for successful software projects.
- It is recommended that leader of the team should be suitable and competent enough to assist team members in effective decision making.
- It is advisable that roles, issues and responsibilities should be well defined and assigned to qualified members.
- The objectives and goals should be clearly defined at the onset of a project.
- For the success of the project, the team should handle all its conflicts constructively and respectfully.
- Team members should treat, collaborate and support each other honestly, sincerely and with respect and trust.
- Communication is a very important aspect for effective teamwork that leads to project success.

7. Future work:

When conducting this study, other questions were found interesting for future studies. Therefore, these questions are presented below:

Data collection: our data was too small when we look on GitHub that we collected from, we talk about a huge dataset that it is free to download it from GHTorent, the main cause we didn't fetch our data from this site is too big for our abilities.

Models study: we try two different algorithms on our data and we get a big success on our research, what if we apply more than two algorithms? What kind of the results we will get? And this is a big if.

Accuracy: the reason behind our low accuracy which is equal to 75.5% was our data sorted and we only have tested on 10 attributes what if we search about more data details? We think that our accuracy will increase.

Prediction Interface: the time for our research is not longer enough to create our application that can let any leader of a team pass the information of his/her team on our application and can easily predict if the project will get success or not and this will help the software engineering for the best.

So, after we think of these questions, we will get the full GHTorent data with the max information we can, and use them to create our clean dataset, after that we choose the best algorithms for classification (more than 8) until we get a higher accuracy (between 95% to 99%) we will work finally on our prediction application.

A decorative graphic of a scroll with an orange outline and grey rollers at the top and bottom corners. The word "References" is centered on the scroll.

References

- [1] When teamwork really matters: task innovativeness as a moderator of the teamwork–performance development projects Martin Hoegl a,*, K. Praveen Parboteeah b,1, Hans Georg Gemuenden relationship in software
- [2] The Quality of Teamwork on Methodology in Software Development Workflow Zairina Ibrahim1*, Md Gapar Md Johar2 , Normy Rafida Abdul Rahman3
- [3] The Impact of Teamwork on Work Performance of Employees: A Study of Faculty Members in Dhofar University Shouvik Sanyal1* , and Mohammed Wamique Hisam
- [4] The Quality of Teamwork on Methodology in Software Development Workflow
- [5] [Midha, V., and Palvia, P. 2012. Factors affecting the success of Open Source Software. Journal of Systems and Software 85 (4), (2012), 895-905.]
- [6] [Guerrouj, L., Azad, S., and Rigby, P. C. 2015. The influence of App churns on App success and StackOverflow discussions. In Y.-G. Guéhéneuc, B. Adams & A. Serebrenik (eds.), SANER (p./pp. 321-330), : IEEE. ISBN: 978-1-4799- 8469-5
- [7] Success or Failure Identification for GitHub’s Open Source Projects
- [8] [Istiyanto, J. E., and Wahyu Rahardjo Emanuel, A. 2009. Success Factors of Open Source Software Projects using Datamining Technique. 1st Information and Communication Technology International Seminar, July 2009. ISSN 2085- 692X]
- [9] <https://git-scm.com/>
- [10] <https://en.wikipedia.org/wiki/Git/>
- [11] The GHTorent dataset and tool suite by [Georgios Gousios](#)
- [12] Success or Failure Identification for GitHub’s Open Source Projects by Junaid Maqsood and Iman Eshraghi
- [13] <https://junaidmaqsood.com/success-or-failure-identification-for-githubs-open-source-projects/>
- [14]Chapter 4: Decision Trees Algorithms by [MadhuSanjeevi\(Mady \)](#)Oct 6, 2017
- [15] Decision Tree Classification in Python by [AvinashNavlani](#)December 28th, 2018
- [16]https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm
- [17] Accelerating the XGBoost algorithm using GPU computing by Rory Mitchell, Eibe Frank July 24, 2017

Abstract:

Successful or failed software projects have been discussed in literature for many years. Successful software projects are often defined as meeting business objectives, deliver on time and within budget, and meeting requirements. Different factors that contribute to software project success have been identified in the literature. Some of the most common factors that lead to software project success are: user involvement, management support, realistic requirements, and having good estimations. However, there are different opinions about what a successful software project is.

Since there are different factors that affect on the success of the project, this study investigated in factors that affect on the GitHub projects. In addition, a comparison between XGBoost and Random Forest result classification. The result shows that good communication and collaboration between members, an experienced leader, problem attention and time response have a higher affects on the project success.

نبذة مختصرة:

تمت مناقشة مشاريع البرامج الناجحة أو الفاشلة في الأدبيات لسنوات عديدة. غالبًا ما يتم تعريف مشاريع البرامج الناجحة على أنها تحقيق أهداف العمل، والتسليم في الوقت المحدد وفي حدود الميزانية، وتلبية المتطلبات. تم تحديد العوامل المختلفة التي تساهم في نجاح مشروع البرمجيات في الأدبيات. بعض العوامل الأكثر شيوعًا التي تؤدي إلى نجاح مشروع البرنامج هي: مشاركة المستخدم، ودعم الإدارة، والمتطلبات الواقعية، والحصول على تقديرات جيدة. ومع ذلك، هناك آراء مختلفة حول ماهية مشروع البرمجيات الناجح.

نظرًا لوجود عوامل مختلفة تؤثر على نجاح المشروع، فقد بحثت هذه الدراسة في العوامل التي تؤثر على مشاريع GitHub. بالإضافة إلى ذلك، مقارنة بين XGBoost وتصنيف نتائج الغابة العشوائية. تظهر النتيجة أن التواصل والقيادة الناجحة التعاون والاهتمام بالمشكلة وقت الإجابة لها تأثير كبير على نجاح المشروع.

Résumé :

Les projets de logiciels réussis ou échoués sont évoqués dans la littérature depuis de nombreuses années. Les projets logiciels réussis sont souvent définis comme la réalisation des objectifs commerciaux, la livraison à temps et dans les limites du budget, et la satisfaction des exigences. Différents facteurs contribuant au succès du projet logiciel ont été identifiés dans la littérature. Certains des facteurs les plus courants qui mènent au succès d'un projet logiciel sont: l'implication des utilisateurs, le soutien de la direction, des exigences réalistes et de bonnes estimations. Cependant, il existe différentes opinions sur ce qu'est un projet logiciel réussi.

Puisqu'il existe différents facteurs qui affectent le succès du projet, cette étude a examiné les facteurs qui affectent les projets GitHub. En outre, une comparaison entre la classification des résultats XGBoost et Random Forest. Le résultat montre que la communication, collaboration, l'attention aux problèmes et le temps de réponse ont un impact plus important sur la réussite du projet.