UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA


Laurea Magistrale in Informatica

Curriculum Software Engineering and IT Management

# BeForE: A Bert-based Effort Estimation Approach for Agile User Story

RELATORE

Prof. **Filomena Ferrucci**

CORRELATORE

Prof. **Fabio Palomba**

Dott. **Stefano Lambiase**

CANDIDATO

**Gianluca Spinelli**

Mat. : 0522500981

Anno Accademico 2021/2022

# CONTENTS

# Abstract

*(English)*

Effort refers to the number of units of work required to complete a task, activity or project, and can be expressed in different units of measure: Minutes, Days, Hours, and Story Points in Agile projects. In project management, estimating the effort required to complete a task or activity in a project serves as a foundation to determine the duration of said tasks, activities and therefore the project. Effort estimation is one of the most critical tasks, which if done incorrectly can lead to project failure. In this regard, in recent years, there has been an increase in interest in using Machine Learning/Deep Learning techniques to create models which, given a task, could return a numerical value representing the effort. This study focuses on the Effort estimation of a User Story in Agile projects. The goal of this work is to advance state of art, providing various contributions to the scientific community: we present a new model "BeForE" for Effort estimation based on the Bert neural network, capable of estimating, starting from the description of a User Stories, both in Story Points and Minutes; we experiment the use of other Effort Drivers of a User Story to verify the variation in the performance of the models. The research objectives were summarized in four research questions, which were answered by conducting a quantitative study, through which the BeForE model was built and the various experiments were carried out. The results show the excellent performances obtained by the BeForE model, both for the estimation of the Story Points and the estimation of the Minutes. Furthermore, both in the estimation of the Story Points and in the estimation of the effort in minutes it can be noted that the addition of secondary Features, such as the Priority of a US, passed to the BeForE model built with Bert, does not provide any advantage in the estimation. This result is very interesting as it answers several questions posed by the researchers in the latest papers and opens the door to new experiments.

# Abstract

*(Italiano)*

L'Effort (sforzo) si riferisce al numero di unità di lavoro necessarie per completare un'attività o un progetto e può essere espresso in diverse unità di misura: minuti, giorni, ore e Story Point nei progetti Agile. Nella Project Management, la stima dello sforzo richiesto per completare un compito o un'attività in un progetto, serve come base per determinare la durata di detti compiti, attività e quindi del progetto. La stima dell'Effort è una delle attività più critiche, che se eseguita in modo errato può portare anche al fallimento del progetto. A questo proposito, negli ultimi anni, c'è stato un aumento dell'interesse nell'utilizzo di tecniche di Machine Learning/Deep Learning per creare modelli che, dato un task/issue, potessero produrre un valore numerico rappresentativo dello sforzo. Questo studio si concentra sulla stima dell'Effort di una User Story nei progetti Agile. L'obiettivo di questo lavoro è quello di avanzare lo stato dell'arte, fornendo diversi contributi alla comunità scientifica: viene presentato un nuovo modello "BeForE" per la stima dell'Effort basato sulla rete neurale Bert, in grado di stimare, partendo dalla descrizione di una User Story, sia in Story Point che in Minuti; sperimentiamo l'utilizzo di altri Effort Drivers di una User Story per verificare la variazione delle prestazioni dei modelli. Gli obiettivi di ricerca sono stati racchiusi in quattro domande di ricerca, alle quali si è risposto conducento uno studio quantitativo, attraverso il quale è stato costruito il modello BeForE e sono state effettuate le varie sperimentazioni. I risultati mostrano le ottime performance ottenute dal modello BeForE, sia per la stima degli Story Point che per la stima dei Minuti. Inoltre, sia nella stima degli Story Point che nella stima dello sforzo in Minuti si può notare che l'aggiunta di Features secondarie, come ad esempio la Priorità, passata al modello BeForE costruito con Bert, non fornisce alcun vantaggio nella stima. Questo risultato è molto interessante in quanto risponde a diverse domande poste dai ricercatori negli ultimi lavori e apre le porte a nuovi esperimenti.

# CHAPTER 1

# INTRODUCTION

'Good estimation helps product owners optimize for efficiency and impact.'–an Atlassian manager.

Nowadays, software is a component present in the life of each of us. For this reason, its development and engineering are essential activities capable of influencing various aspects, first of all, human life. Therefore, it is important to note that up to 69% of software projects fail. Knowing better the factors that influence the outcome of a project is mandatory to improve the percentage of successful projects and increase their probabilities. The leading cause of project failure is time or resource estimation errors, resulting in incorrect cost estimates.

## 1.1 Motivations and Objectives

Estimating software development effort is the basis for budget management, project planning, and resource management. Effort Estimation is the process of forecasting how much effort is required to develop or maintain a software application. This practice is critical as poor estimates and consequent planning can lead to disastrous consequences, often not remedied. The estimate that is made must not be too pessimistic, because you could lose business opportunities, nor too optimistic, as it could lead to significant losses. Estimating the effort is an operation that can be performed both at the beginning of a project and during, for example, to estimate the effort required for a maintenance intervention. Many industrial disasters have occurred as a result of incorrect estimates, but despite this, many companies are still unable to propose realistic estimates.

Summarizing, it can be said that Software Engineering and project management are two

of the most critical factors influencing the project outcome. In recent years we have heard more and more about the use of Machine Learning in Software Engineering. Software developers and architects have begun to experiment with the use of this technique in many use cases and many different ways. Among these, several researchers have experimented with the use of this technique to create models which, starting from an input given by the user, could return a numerical value in terms of effort.

In recent years various Effort estimation models have been proposed, which however have focused on the use of Story Points as a unit of measurement and only on the description of the Issue.

Starting from this, the objective of our study was born, namely to experiment first of all with new estimation models, not only in Story Points but also in Minutes, which is obviously a much more practical and above all clear-cut unit of measurement compared to Story Points, and then experimenting with the use of new features in the estimation and comparing their performance with models that use only the description of the issues.

In addition, the context of the study is Agile development, Project Management, Effort Estimation and the use of machine learning models. In particular, the study focuses on User Stories, an Agile practice, used above all in Scrum, to "capture" user needs by expressing in a general, non-detailed way, characteristics, functions and requirements for the product to be created. Therefore, with this study we want to advance the state of the art, not only by providing new models for estimating effort in Story Points and Minutes of development, using one of the most powerful technologies in this area, Bert, but also answering the main question "Can I get better estimates if I use other information than the description of the User Story itself?".

The research approach consists of a quantitative study, which consists of constructing an MLP neural network that uses the output of the Bert model to perform a regression. To evaluate and compare the various models, MeanAbsoulteError, MeanSquaredError and RootMeanSquaredError were chosen as metrics. Furthermore, to obtain models that can estimate the US effort of new projects, and not necessarily US maintenance; the User Stories were sorted in chronological order before each training, to train the models on the old US. As a first step, a Data Analysis operation was carried out on the TAWOS database, a large database of about half a million Issues, to extrapolate details, understand its structure, how it is populated and study its possible uses. As a second step, baseline models were built that estimate effort in two ways: Story Points and Minutes. Subsequently, the respective models were redeveloped using additional features. Whereas the TAWOS database contains several fields representing the effort in minutes for each User Story, three different Jira fields were used as target variables of the model for building the minute estimation

models, i.e. 'In Progress Minutes', 'Resolution Time Minutes' and 'Total Commitment Minutes'.

## 1.2 Results

In this research thesis, we extend the state of the art by providing the BeForE model for Story Point and Minutes Effort estimation and answering the main question of our study.

The quantitative approach shows that by using the Bert language model to extrapolate contextual information from a User Story, it is possible to obtain excellent performances in estimating both Story Points and Minutes needed for development which, with a suitable comparison, could overcome the state of the art constituted by the GPT2SP model. To verify this, it would be enough to train a replica of the GPT2SP model on the data used to train our BeForE model and compare the performances. This would also allow us to observe which of the two most famous language models, GPT-2 and Bert, performs better in the execution of this NLP regression task. A further important result is the negative outcome of the two research questions RQ2 and RQ4, whose experiments show that the addition of the secondary features, Priority and Project_ID, do not improve the model's performance but rather worsen it. This allows us to advance the state of the art since, in the replies made by V. Tawosi et al., the research team suggests building models that are not only based on semantic similarity but also other characteristics, in particular, extracted from the TAWOS database. Our analyzes carried out on the dataset's Features show that effectively of all the fields that make up the Issue table, none has a sufficiently high correlation (measured with the Pearson coefficient) to make the estimate more accurate.

This is not to say that the assumption made by the researchers is wrong, but certainly, it is necessary to operate Feature Engineering to obtain better predictive attributes. In particular, we believe that additional features that should be used in conjunction with textual information should indicate the complexity of the task and perhaps, also the productivity and experience of the team. Two identical User Stories but belonging to different projects and contexts, such as: "As an administrator, I want access to my control panel" will appear to the model as identical, when in reality they could have very different complexities because, for example, in the first case it is a system in which the administrator has few responsibilities, while in the second case it is a system in which the administrator has many tasks and functions.

Therefore, providing an indicator, even categorical, on the complexity of the User Story, in terms of the number of software components involved, for example, could certainly prove to be an excellent strategy.

Estimating in Story Point remains a difficult task today as it is a metric whose use and interpretation varies from team to team and for this reason the prediction performed by a model

(using only the description), in the case of projects that have just started, will not be never perfectly aligned with the team's esteem policy, as he had no way to learn from the team's past experiences. The situation is different if instead, we are talking about User Stories or Issues that represent maintenance operations. In this case, the model, training on the historical data of the single team, can learn their policy and estimate a value close enough to what the team would estimate for an Issue carried out on the same project. Another observation we can make on the experiments carried out is that in the case of the estimation in minutes, the Feature Augmentation operation led to a drastic decrease in the number of entries in the dataset, therefore it is necessary to collect other data to repeat the training with multiple User Stories.

## 1.3 Document Structure

We have proposed all the topics mentioned above deeply in the seven chapters of which this thesis work is composed. Following is a summary of the chapters:

- **Chapter 2: Background and Related Work**, that illustrates the currently available works and discoveries of literature in the field of human aspects and global software development;
- **Chapter 3: Research Methodology**, that illustrates the research questions and the research strategies used to achieve the main objective;
- **Chapter 4: Analysis of The Results**, that illustrates the achieved results for each research question and the conclusions regarding the main objective;
- **Chapter 5: Observations and Lessons Learned**, that illustrates observations on the research and the lessons learned for the research community;
- **Chapter 6: Threats to Validity**, that illustrates the threats to the validity of the study and the way we have mitigated them;
- **Chapter 7: Conclusions and Future Work**, that illustrates a summary of the research and future prospects.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

This chapter illustrates the state of the art of the key topics and the work in the literature related to our study.

## 2.1 Effort Estimation

Estimating software development effort is the basis for budget management, project planning, and resource management. Effort Estimation is the process of forecasting how much effort is required to develop or maintain a software application. This practice is critical as poor estimates and consequent planning can lead to disastrous consequences, often not remedied. The estimate that is made must not be too pessimistic, because you could lose business opportunities, nor too optimistic, as it could lead to significant losses. [1]. Estimating the effort is an operation that can be performed both at the beginning of a project and during, for example to estimate the effort required for a maintenance intervention. There are many industrial disasters that have occurred as a result of incorrect estimates, but despite this, many companies are still unable to propose realistic estimates. This problem translates into stressed working times, a worsening of the quality of work and, in case of late delivery, lack of customer satisfaction. [2]. A Project manager is often challenged to align mainly six project constraints - Scope, Time, Cost, Quality, Resources, and Risk in order to accurately estimate the project. The most frequently asked questions that a project manager must consider at the beginning of the project are the following:

- How much work is to be estimated? (Scope)
- How to estimate the project? (Techniques)

- How much time it will require to complete the project? (Schedule)
- Who will be doing the project? (Resources)
- What is the budget required to deliver the project? (Cost)
- Any intermediary dependencies that may delay or impact the project? (Risks)

The Project Estimation Approach that is widely used is Decomposition Technique. Decomposition Techniques take divide and conquer approach. Size, Effort and Cost estimation are performed in a stepwise manner by breaking down a Project into major Functions or related Software Engineering Activities. The main steps we see below:

- Understand the scope of the software to be built.
- Generate an estimate of the software size.
- Generate an estimate of the effort and cost.

There are various techniques for estimating effort. Following Boehm's classification, we can represent methods in three different categories: expert judgment, algorithmic estimation, and analogy based estimation.

**Algorithmic Methods:** In this approach, however, we have an algorithm that processes and uses the input data to obtain the predicted effort. Also in this case, we need information on the systems developed in the past, such as costs, which will be fed to the algorithm, together with some project measures, to obtain the model to be used for the new project; the effort will be estimated from the model. We have two categories of algorithmic approach:

1. **Cost models:** Cost models: provide estimates of the project cost (commitment, effort) on the basis of a primary characteristic (usually the size, called cost factor) and a series of secondary characteristics (cost drivers).
2. **Constraint models:** they provide estimates of the project cost (commitment, effort) based on multiple characteristics, therefore not only based on size. Cost drivers can be evaluated (depends on the model chosen).[3]

**Expert Judgment:** it is one of the most used methods of estimating effort. Experts are people who have participated in the development of the system in the long run and who know the application domain (this last requirement is not trivial). The advantage of consulting experts lies in the fact that it is a relatively cheap technique and, above all, accurate when the experts already have experience with similar application contexts; the disadvantages, on the other

hand, consist in the lack of experience in that precise context on the part of these experts, in order to arrive at forecasts very far from what the real value of the effort will be. The execution of estimates based only on experience and without using data, could introduce bias factors, or subjective factors that vary from expert to expert and that lead to an estimate that is not strictly objective.

**Analogy Based Estimation:** The analogy allows you to select certain projects from the past with certain characteristics similar to those of the system to be developed; the cost of the system to be developed is identified by comparing the project with similar ones, and then considering one in particular (perhaps the one that comes closest in terms of similarity). Analogy is an accurate technique if similar projects are available, otherwise it turns out to be impossible to practice; moreover, a repository of previous projects is required (which costs a lot) and it is difficult to determine when an old system looks like the one to be developed (a similarity function must be identified, but it is not trivial). The analogy is, therefore, a type of estimate that can be used when you have similar products, in terms of size and utility. The estimate is obtained by evaluating the expected amount of work and the development time: coefficients (modifiers) are applied to the costs of the system, which take into account the particular differences in characteristics of the previous systems compared to the current one in development.

To date, more and more companies are modifying their project development models, integrating Agile within these practices. Adopting agile approaches such as Scum and Extreme Programming involves a phase of estimating the different effort, which sees the use of techniques different from the traditional ones. One of the most popular techniques is Planning Poker: this technique sees the participation of all the developers and the manager (Scrum Master) in estimating the various features to be implemented (in jargon User Story) in a specific period of time (in jargon Sprint) [4, 5].

The **Planning Poker** technique consists of using the Fibonacci sequence to assign a Story Point value to a User Story or an object. The Fibonacci sequence is a mathematical series of numbers introduced in the 13th century and used to explain some formative aspects of nature, such as the branching of trees. The series is generated by adding the two previous numbers to get the next value in the sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on. For agile estimation purposes, some of the numbers have been changed, resulting in the following series: 1, 2, 3, 5, 8, 13, 20, 40, and 100.

These numbers are represented in a series of playing cards. Team members play "Planning Poker" to provide an estimate in the form of a point value for each item[6]. Here are the steps:

- Each team member gets a set of cards.

- The business owner (who does NOT get to estimate) presents the item to be estimated.
- The item is discussed.
- Each team member privately selects a card representing his/her estimate.
- When everyone is ready, all selected cards are revealed at the same time.
- If all team members selected the same card, then that point value is the estimate.
- If the cards are not the same, the team discusses the estimate with emphasis placed on the outlying values:
  - The member who selected the lowest value explains why he/she selected the value.
  - The member who selected the highest value explains why he/she selected the value.
- Select again until estimates converge.
- Should lengthy or “in-the-weeds” conversations result, team members may use a two-minute timer to timebox the discussion, selecting again each time the timer runs out, until conversion.
- Repeat for each item.

It is important to point out that the points do not have the same meaning between the various teams; for example, the "two" of one team is not the same as the "two" of another team. Therefore, team speed, derived from Story Points, should not be used to compare productivity between teams [7].

## 2.2 Machine Learning for Effort Estimation

*“The field of study that makes computers capable of learning without being explicitly programmed.”*

Thus Arthur Lee Samuel, pioneer of Artificial Intelligence, 1959 coined the term "Machine Learning" [8] [9]. Machine Learning allows computers to learn from experience, so that they can perform specific tasks, partly "human" and very complex. In recent years there has been a considerable increase in interest in Machine Learning, prompting companies and researchers to experiment with the use of this technique in every possible field. Contrary to what one might think, the use of Machine Learning for effort estimation has been a well-known topic for some time. There are several studies, already in the 90s, which see the use of Machine Learning, such as the 1995 study by Krishnamoorthy Srinivasan and Douglas Fisher *”Machine Learning Approaches to Estim Software Development Effort”*[10], in which two Machine Learning models are tested for effort estimation task.

To date, there are several Machine Learning and Deep Learning models proposed in various research, however, a stable point has not yet been reached.

The Machine Learning approaches used to date are different: Bayesian Network, Decision Tree, Support Vector Machine, Genetic Algorithm, Artificial Neural Network, Logistic Regression, Random Forest and more [11].

Models that use neural networks to predict effort through textual information have also been proposed[12][13]. Morakot Choetkiertikul et al proposed a deep learning *"DeepSE"* approach to estimate effort in Story Points. This model is based on the construction of a language model and uses Long Short-Term Memory (LSTM) with Recurrent Highway Network (RHWN) to learn information from the description of a user story and predict story points [12]. In response to this model, whose performances have not been confirmed in the replication performed by F. Sarro, V.Tawosi et al [14], a new model has been proposed that uses the well-known pre-trained language model GPT-2 [15]. Michael Fu et al proposed a Transformer-based architecture *"GPT2SP"* that leverages the GPT-2 language model to extract semantic information, such as context, from the description of a generic Issue. Also in this case the replicas carried out by F. Sarro, V. Tawosi et al did not confirm the results [16]. Both proposed models use purely textual information, without considering other additional numerical information.

### 2.2.1 Bert Language Model

With the introduction of BERT, Google has changed the way search engines try to use queries typed by internet users. The battle for the top positions in SERP Search Engine Results Pages, has been played for years on the awareness of how search engine algorithms work, primarily Google, and how they decide which ones they are. web content that is more consistent with the queries typed by users. In October 2019 Google introduced a significant update in this field: it is the BERT (Bidirectional Encoder Representations from Transformers) algorithm [17]. What distinguishes this algorithm from the previous ones is the ability to analyze the words typed bidirectionally.

Bert's other 'similar' understand the terms by observing only what is on their left or only what is on their right. On the other hand, the new algorithm tries to interpret a word by completely contextualizing it based on everything that precedes and follows it.

Thanks to this modus operandi, BERT can understand much better users' search intentions and provide more relevant results, increasingly refining its work based on the millions of data that arrive every second. BERT is an open source Machine Learning framework for NLP, designed to help computers understand the meaning of ambiguous language in the text by using surrounding text to establish context. The BERT framework has been pre-trained using Wikipedia text and can

be fine-tuned with a question and answer datasets to meet a wide range of tasks. This model is based on a Transformer, a deep learning model in which every output element is connected to every input element. The weightings between them are dynamically calculated based on their connection (In NLP, this process is called attention [18]).

The transformer uses a network of encoders and decoders but, as Bert is a pre-training model, only a multi-level network of encoders is used (12 standard encoders), to learn a representation of the input text. Bert acquires each word having its representation, more appropriately called embedding. Each level performs a calculation of the attention on the embedding of the terms of the previous level, and then creates a new intermediate embedding, with the same dimensions as the embeddings produced by the previous levels. In the figure, extracted from the official article, we can see that E1 is the representation of the considered token, T1 is the final output and Tom are the intermediate representations of the same token. For example in a 12-level Bert model, a token will have 12 intermediate representations.

Bert's cutting-edge performance is based on two key points: first, new pre-training strategies called Masked Language Model (MLM) and Next Sentence Prediction (NSP), secondly, the use of a large amount of data and computing power for the training phase.

**Masked Language Model:** The Bert architecture analyzes sentences with some randomly masked words (hence the name MLM) and tries to correctly insert the word "hidden".

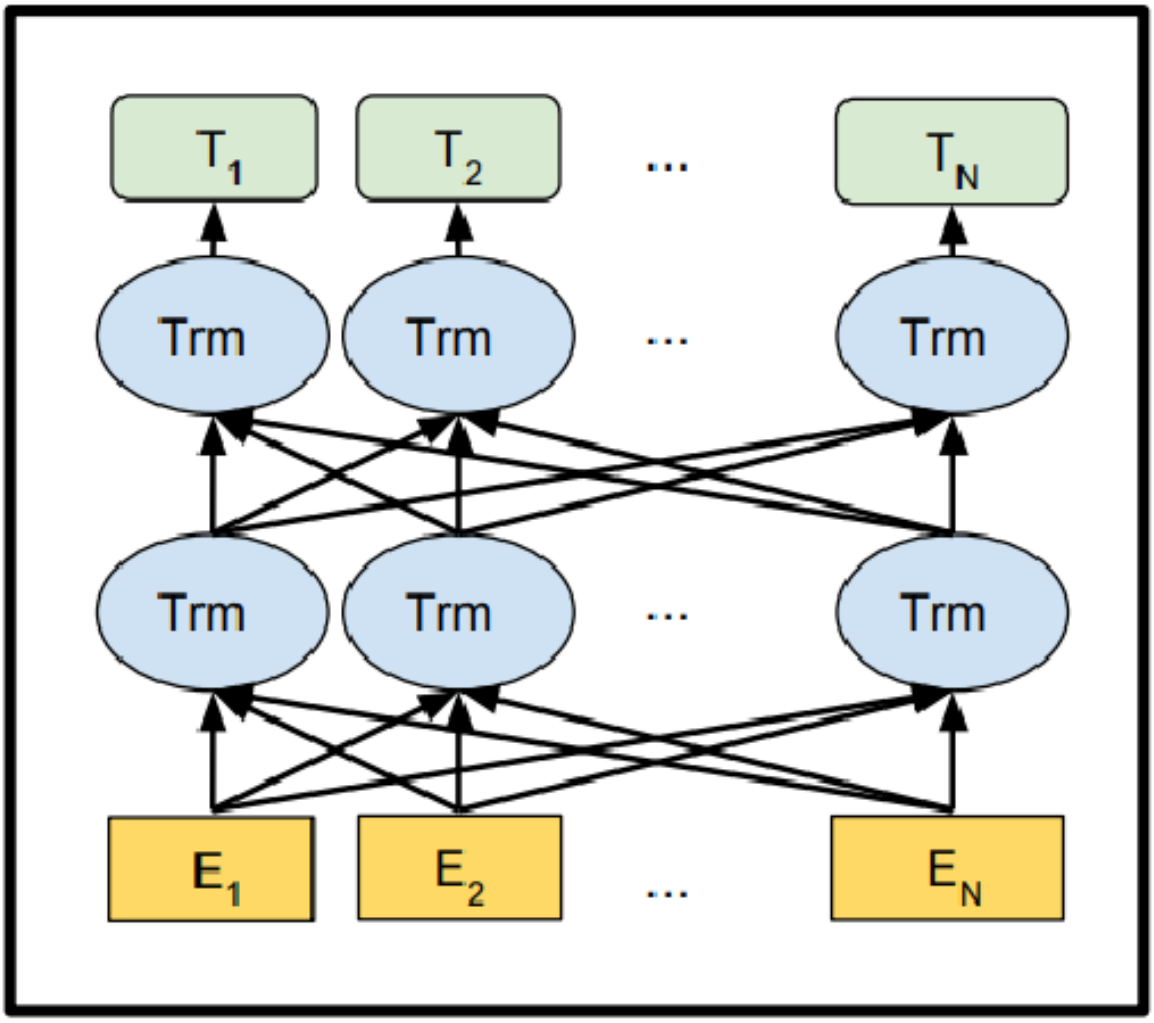
**Next Sentence Prediction:** One of Bert's main innovations is being able to predict what a user will say after a certain sentence. The sentence is analyzed and then a confidence level is determined to predict whether a second sentence hypothesis "fits" logically as the next sentence.

The understanding of the context by an algorithm derives from its ability to observe all the words of a sentence at the same time and therefore to understand how every single word affects all the others.

Bert, after selected a target word, can take into consideration at the same time all the other words present in that same sentence, therefore in a bidirectional way. This procedure allows the model to better understand the context and the semantic meaning of the sentences.

## 2.3 Correlation between Story Point and Development Time

Story Point estimation through Planning Poker is an effort estimation method widely used in Agile contexts today. Given its nature, Story Points cannot be standardized and therefore we cannot




consider two identical estimates at the time these were made by different teams. In the application of Planning Poker and the use of Story Points, by the development team, a kind of policy is created, through which it is defined what is meant by 1 Story Point. So we would like to believe that we cannot consider estimates of different projects and different teams in the same way. V. Tawosi's [19] paper analyzes this aspect by studying the correlation between Story Point and Development Time. The results show us what we had guessed, which is that there is no statistically significant correlation between Story Point and Development Time. In addition, this work evaluated which data recorded by Jira in minutes is the most representative of the development effort. The results show that the "In Progress" data is the most representative data, representing the task's time elapsed "In Progress" state. In addition, this work evaluated which data recorded by Jira in minutes is the most representative of the development effort. The results show that the "In Progress" data is the

most representative data, representing the task's time elapsed "In Progress" state.

Considering these points illustrated above, our work aims to test first of all a model that estimates the effort in minutes of a User Story starting from textual information only and then experimenting if augmenting textual features with numerical features allows us to have better performance in estimating.

## 2.4 Related Work

This work was born from an analysis of some papers regarding Agile Effort Estimation. The first paper analyzed was "A Versatile Dataset of Agile Open Source Software Projects" by Vali Tawosi et al [20]. This paper presents a dataset, published as a relational database, of Issues extracted from the Jira Management Tool. These issues belong to open-source industrial projects, such as Spring, Apache etc. The TAWOS database contains about 480,000 issues of various types such as Bug, Story, Improvement, New Feature, Task etc., this allows it to be used in various contexts. Therefore, analyzing this dataset the idea was born to try to identify new predictive attributes that could perform the regression task more precisely.

**DeepSE:** Morakot Choetkiertikul et al. in 2019 proposed DeepSE [12], a model based on LSTM and RNN for the effort in Story Point. Their goal was to build a predictive model that, taking the title and description of any issue as input, would return an estimated value in Story Point. DeepSE combines title and description into a single feature on which the model is trained. To train their model, Morakot et al. carry out a pre-training of the lowest level of the architecture (LSTM) and subsequently, using Theano they train the Recurrent Highway Network. Six research questions are presented in their paper: With RQ1 they verify the feasibility of their proposed model for estimating Story Points. Questions RQ2 and RQ3 they highlight the benefits of using Recurrent Highway Networks and Long Short-Term Memory Networks for regressor training. With the RQ4 question, they evaluate the use of the DeepSE model for cross-project estimation, i.e. the estimation of new Issues on new projects. Question RQ5 compared the performance of the baseline model, without any normalization applied to the target variable, with a model in which the Story Points have been normalized. Finally, with the last question, RQ6, they compare the performance of their model with the other methods proposed in the literature.

The results, not confirmed by the replication performed by F. Sarro et al. [14], Respond positively to all research questions and show the validity and excellent performance of the model.

**GPT2SP:** The second work analyzed is "GPT2SP: A Transformer-Based Agile Story Point Estimation Approach"[13]. GPT2SP is a model proposed by Michael Fu et al. which estimates the effort in Story Point, of a general issue, using Title and Description. This model uses the well-known pre-trained language model GPT-2, which, starting from the texts, produces subword-tokenized issues. Unlike the DeepSE model which uses a word-level tokenizer, GPT2SP preserves the most common words and splits only the rare words. The model proposed by Michael Fu et al., as well as the DeepSE model analyzes only textual features, thus relying only on the semantic similarity of the Issues. Their main goal is to provide a new modern estimation model, which surpasses DeepSE's performance. In addition to this, the researchers have developed a web interface where it is possible to test the model and have an explanation of the result. As the last step, they perform a survey to assess whether a tool estimates the effort, which also explains, could be more credible and used by Software Engineers. In this work, the researchers divide the training into Within-project and Cross-Project. In Within-project training, the GPT2SP model is trained on the part of the Issue of a specific project and tested on the remaining issues of the same. In cross-project training, on the other hand, the GPT2SP model is trained on the issues of one project and tested on the issues of another project. This approach is because an estimation model could be used in two ways: to estimate the effort of a task concerning a project already in progress, such as a maintenance intervention, or to estimate the action of a User Story at the beginning of a project, therefore without historical data.

The first difference between our work and the models described above is that, considering that human estimation of Story Points may not be a perfect indicator of development effort, training a model on this dependent variable could lead to inaccurate estimates. For the above reasons it was decided to replace the dependent variable with "In Progress Minutes" as it is the most representative of the development time, [19].

Providing a detailed estimate of the development effort could be a very useful contribution for developers and Project Managers / Scrum Masters.

Since this variable is effective and objective, as it is stored by Jira automatically, it does not present subjective factors, unlike the Story Points. Faced with this choice, we concentrated on Cross-project estimation only, to provide a valuable model for estimating the development effort of a User Story in minutes, even in the early stages of the project.

In addition to these differences, our work differs from the two models for the following reasons:

**User Story:** A first important difference is in data selection. In our work we only use User Story Issues, so we want to create and experiment with a novel model that estimates the User Story

development effort, even at the beginning of the project. This filter halves the number of possible data to use but is necessary to perform our task.

**Bert Model:** The third difference is in the model used to process the textual information. In our work, we use Bert, an open-source language model developed by Google, which allows users to optimize it according to their needs. With BERT, users can train their question response patterns in about 30 minutes on a single Cloud TPU and hours using a single GPU.

**Textual and Numeric Features:** The last substantial difference is in the choice to use other numerical features to train the model, thus experiencing a possible increase in performance. This choice arose from the nature of the TAWOS database which, for each Issue, is accompanied by various numerical and categorical information.

# CHAPTER 3

# RESEARCH METHODOLOGY

This chapter illustrates the research questions and the research strategies used to achieve the main objective. The primary objective of our research is to propose a method of estimating effort through the use of Machine Learning and to verify if the addition of other numerical characteristics, extracted from the TAWOS database, allows a better estimation. We also want to provide an analysis of the TAWOS database to understand better how it is structured and how it could be used for Agile Project Management. The perspective is that of both Project Managers / Scrum Masters and developers as the process of estimating effort in Agile projects is faced by all team members.

## 3.1 Objectives and Research Questions

**Research Objective:**

Propose a new Machine Learning model for effort estimation and experiment if the addition of secondary numerical features leads to an increase in performance.

To achieve this we have formulated four research questions, which have as their purpose the analysis of the features in the prediction of the effort, both in Story Point, as an initial step, and in minutes, as explained in the Related Work.

> **$RQ_1$**. Is the BeForE model suitable for estimating Story Points of a User Story in Agile Projects?

This research question was formulated to have a starting point. It aims to provide a model that estimates the Story Points of a User Story using only textual information. To date, Story Points,

although subject to human error, are the effort metric most used by Agile team members and for this reason, we believe it is useful to provide an estimation model to support them. Our main goal is not to propose the definitive estimation model, but rather to experiment with a new approach, which could follow with other in-depth studies on the subject.

> **$RQ_2$**. Does the addition of secondary features allow the BeForE model to obtain better Story Point estimations?

This research question was formulated to test whether adding additional features could improve the estimates of the proposed model. The objective of this second research question is to understand if, considering the information that a Jira task can give us, we can derive characteristics related to the effort (in this case expressed in Story Point) to try to "break down" the error human presence in the estimate of the Story Points. To answer this question, a prior analysis of the TAWOS database will be required, which we will illustrate in the Research Approach.

> **$RQ_3$**. Is the BeForE model suitable for estimating Effort Development in minutes of a User Story in Agile Projects?

This research question arose from the analysis of the work "On the Relationship Between Story Points and Development Effort in Agile Open-Source Software" done by V. Tawosi et al. [19], Who analyzes the correlation between development time and Story Points. Analyzing the results proposed in the paper, the idea was born to distinguish ourselves from all the works presented so far in the literature, providing a model that estimates the effort but in minutes, as we think it may be more useful for subjective factors which, on the contrary, they can influence the estimate in Story Point.

> **$RQ_4$**. Does the addition of secondary features allow the BeForE model to obtain better Effort Development estimations?

Similarly, as for RQ2, this research question was formulated to verify if using secondary characteristics extracted from the TAWOS database, there is a performance improvement in the estimation of the development effort in minutes.

To answer these questions, we conducted a preliminary statistical analysis of the characteristics of an Issue. This analysis aims to explore the TAWOS database, analyze its structure and obtain information about the correlation and representativeness of the fields of the "Issue" table.

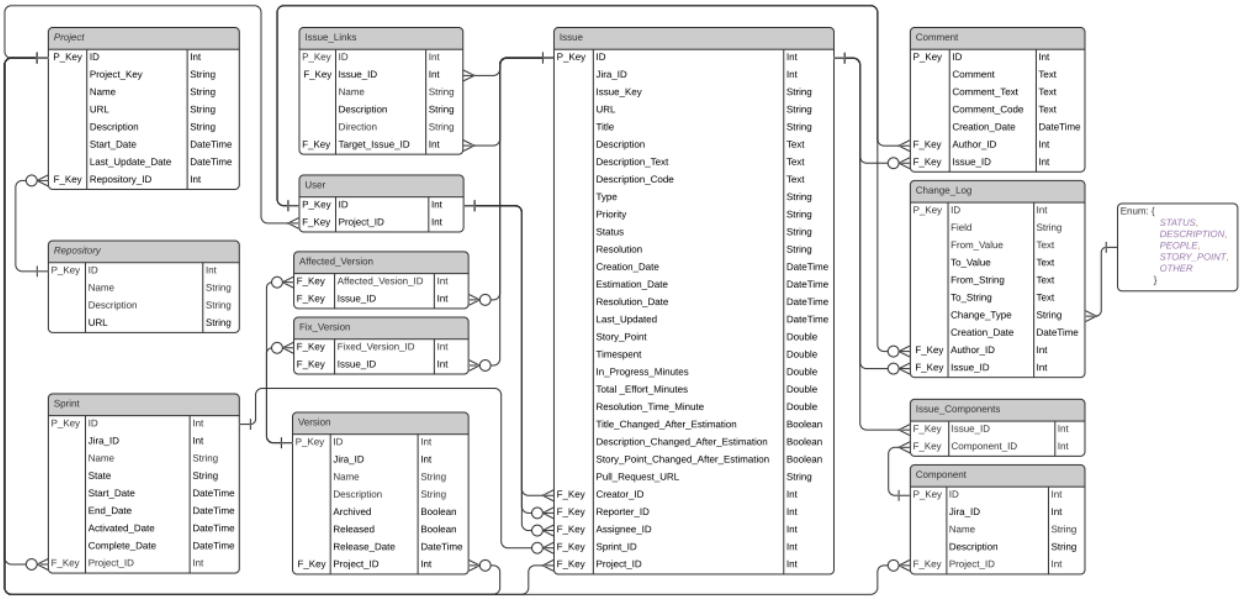



**Figure 3.1:** Overview of TAWOS Database.

## 3.2 Context and Data Gathering

The context of the study is Agile development, Project Management, Effort Estimation and the use of Machine Learning models. In particular, the study focuses on User Stories, an Agile practice, used above all in Scrum, to "capture" user needs by expressing in a general, non-detailed way, characteristics, functions and requirements for the product to be created.

As illustrated in Figure 3.2, the research process began with a Data Gathering phase. We needed to collect a large enough number of User Stories to run our experiments. Therefore, we started from the database made available by V. Tawosi et al. [20] on Github (https://github.com/SOLAR-group/TAWOS). The dataset contains information on the issues of 39 open-source projects selected from 12 public Jira repositories. In total there are 458,232 issues of various types. In the Data Gathering phase TAWOS database was loaded on MySQL. This operation was due to the fact that the researchers released the database in relational form, to allow for better querying. This allowed us to view and query the various tables to carry out a more detailed analysis of the Issues that populate the database.

In figure 3.1 we can observe a representation of the Entity-Relationship Diagram of the TAWOS database. The database used contains 13 distinct tables: 'affected version', 'change log', 'comment', 'component', 'fix version', 'issue', 'issue component', 'issue link', 'project', 'repository', 'sprint', 'user' and 'version'.

**affected version:** This table, containing the columns: Issue ID and Affected Version ID, allows you to trace, given an Issue, the version modified by it. We note that the number of tuples in this

table is 264004, less than the total number of Issues which is 458232.

**change log:** This table contains 9253419 change logs of the various fields of an Issue. It is particularly important as it allows you to analyze any changes made to Jira items, such as Rank, Priority, Story Point estimation, Issue type and more. For each log the field that has been modified, the initial value, the new value, the type of change, the date, the author and the Issue to which it refers are indicated.

**comment:** This table contains 1518327 rows collecting all the comments made in the various issues. There is also a "Comment Code" field which contains code comments.

**component:** This table contains the list of all the components of the various projects. In particular it contains the Jira identifier, the name of the component, the description and the related project.

**issue component:** This table serves as a link, to view each Issue which component/s it refers to.

**project:** This table contains the list of all the projects present in the database, with the following fields: Name, URL, Description, Start Date, Last Update Date, Repository ID and SP Field ID.

**repository:** This table contains the list of all the repositories with name, Description and URL.

**user:** This table contains the list of all the users who have taken part in the development of the projects. Unfortunately it doesn't contain much information but only the user id and the project id. It would have been interesting if in addition to this information it had been possible to obtain the division into groups, with information on the size of the teams.

**sprint:** This table collects all the sprints performed. For each of them there is the Name, the State, Start ate, End Date, Activated Date, Complete Date and the project id.

For our work we mainly focused on the issue table, the most important. This table contains various information relating to each issue, below we show an explanatory table with all the most important fields.

**Table 3.1:** Overview on Issue fields.

| Field | Description |
|---|---|
| Title | Issue title |
| Description | Issue Description |
| Description Code | Contains the descriptive code of the issue. |
| Type | Represents the issue type, it can have one of 26 different types ranging from User Stories to Documentation Tasks. |
| Priority | Represents the Issue Priority specified on Jira. |
| Status | Represents the state the issue is in. |
| Resolution | This field represents how the issue ended. It, therefore, indicates whether the Issue was closed successfully (it was resolved/concluded/implemented etc.) according to the type of Issue, or not. |
| Creation Date | Indicates the issue creation date. |
| Estimation Date | Indicates the date the Story Point estimation was made. |
| Resolution Date | Indicates the date on which the Issue was closed, thus setting the Resolution field. |
| Story Point | Indicates the estimation made in Story Point. The present value is the most recent one, therefore it is noted that it may have been modified even more times until the end of the issue. |
| Timespent | Indicates the total time spent working on the Issue. V. Tawos et al. in their paper they did not provide information on the nature of this field. It should be noted that out of the entire database, only 10243 Issues have a non-zero Timespent value. |
| In Progress Minutes | This field indicates the time spent by an issue in the "In Progress" status. |
| Total Effort Minutes | This field represents the effort in minutes spent to implement an issue. Unlike the In Progress field, this includes time spent on implementation, testing, review, discussions, etc. |
| Resolution Time Minutes | This field represents the time that elapsed between when an issue was created and it was resolved. This feature also takes into account the time between issue creation and the first time it was placed in In Progress status [19]. |
| Title Changed After Estimation | This binary field indicates whether the title was updated after the estimation was made. |
| Story Point Changed After Estimation | This field indicates if, after making the estimation in Story Point, it has been updated. |
| Description Changed After Estimation | This binary field indicates whether the description was updated after the estimation was made. All information relating to updates to these fields are saved in the change log table. |

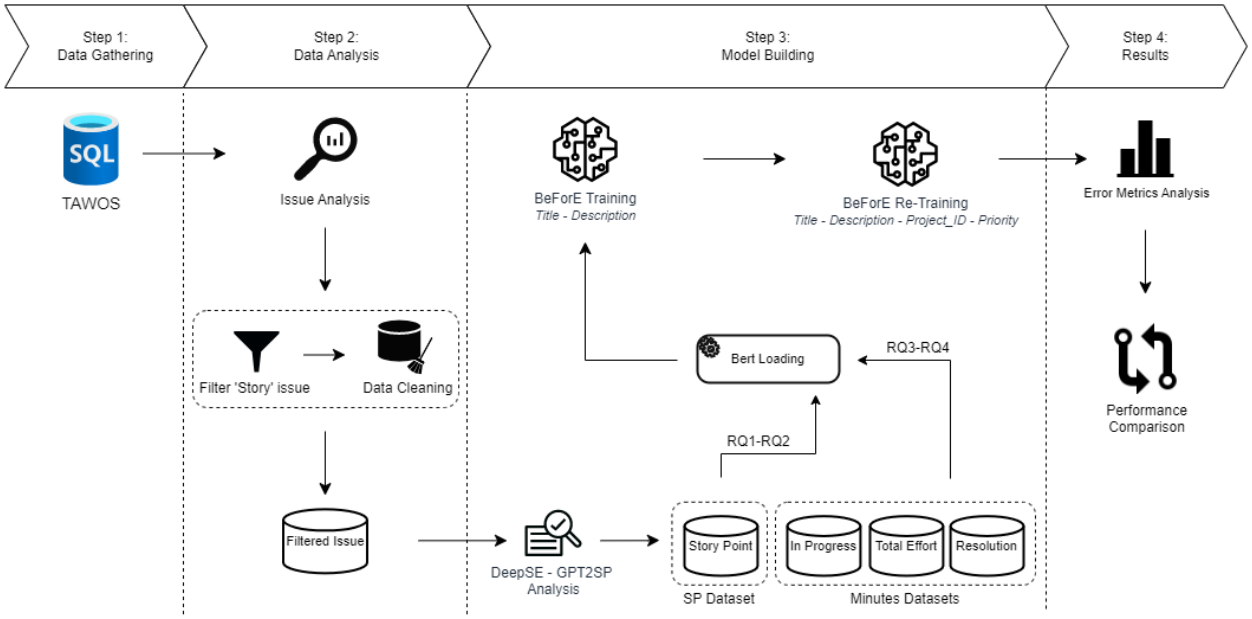



**Figure 3.2:** Overview of our quantitative methodology.

## 3.3 Research Approach

In Figure 3.2 we summarize the entire research process for our quantitative study. To answer the four research questions posed, we first need to carry out a careful analysis of the preliminary data and then select only the data necessary for the construction of the models.

To answer the first research question, it is necessary to build a starting model that can accurately estimate the effort of a User Story in Story Points, using only the title and description as features. This model serves to be able to answer the second research question, verifying whether using other secondary features contained in the TAWOS database, adequately selected, it is possible to obtain better performances. To answer the third, it is necessary to build three models which, starting from the title and description, can estimate the effort of a User Story in minutes, in particular, the three target variables used for the construction of the models are: "In Progress Minutes", "Total Effort Minutes" and "Resolution Time Minutes". To answer the fourth question it is necessary to add other features, suitably selected, and rerun the training to evaluate a possible increase in performance. Our quantitative research approach consists of 4 steps: The first step consists of Data Gathering, as illustrated in the previous section, and allows us to acquire a sufficiently large data source for our work. The second step consists in a data analysis. This step turns out to be of particular importance as it allows us to explore the database and capture important information, so as to select only the Issues that fit our context and objective. The third step consists of various parts: a preliminary analysis of the models proposed in the literature and a phase of construction of new estimation models with the various target variables. Finally, step four consists in the analysis of

the metrics related to the models, comparison of the performances obtained from them and final considerations on the results of the experiments.

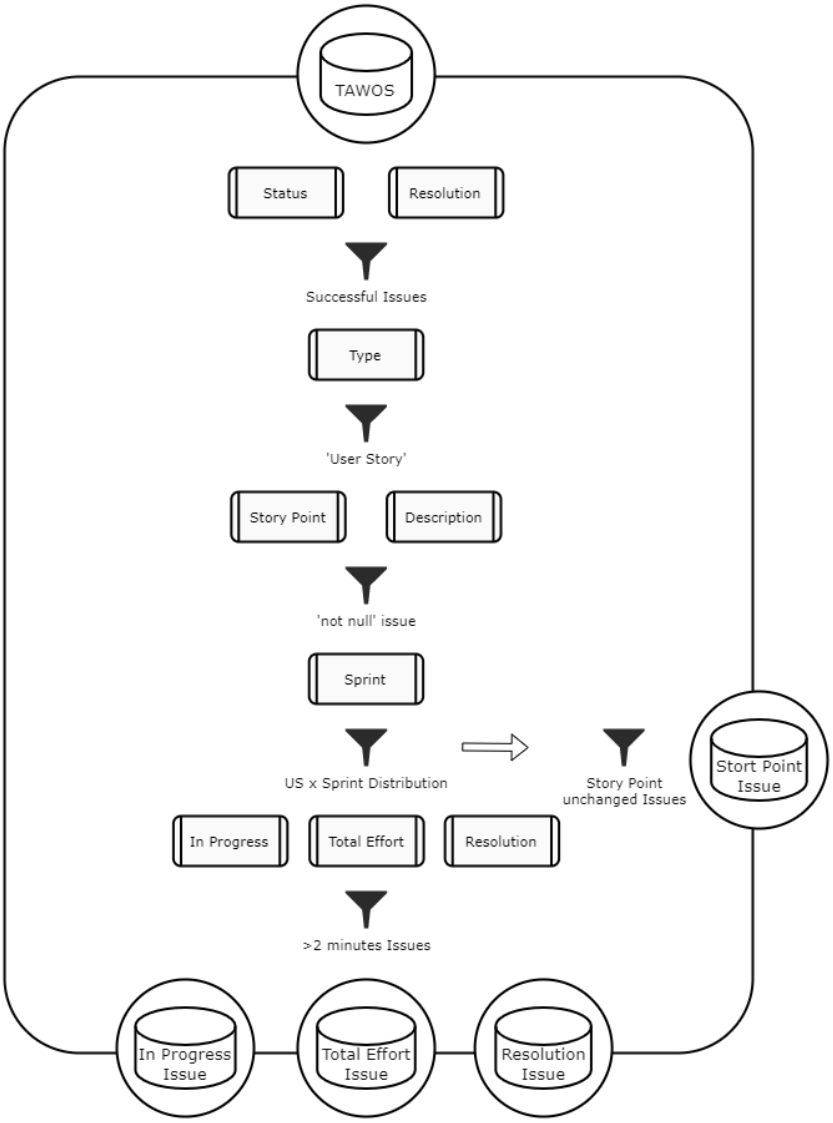



**Figure 3.3:** Step 2: Overview of SQL Analysis Process.

### 3.3.1 Data Analysis

In the data analysis phase, we execute a series of SQL queries to extract information on the database and select the data which we will then use to build the models in the subsequent phases. This phase begins with the analysis of the Issue table. We can observe in image 3.3 all the steps of this analysis phase. In this subsection we illustrate all the steps taken to perform the analysis with the related SQL queries executed. The first fields to analyze are 'Status' and 'Resolution'. Since the database is very large, it is important as a first step to understand which status and types of resolution an Issue can have. Through a select query, using the "Group By" clause, we can identify 75 possible value types of the "Status" field. The most common status we notice is 'Closed' with

**Table 3.2:** All different types of **'Status'**.

| Status | Number of Issues |
|---|---|
| Closed | 334368 |
| Done | 27238 |
| Resolved | 11548 |
| Complete | 5151 |
| Waiting for Release | 75 |
| Ready for Work | 70 |

334368 Issue, followed by 'Gathering Interest', 'Done', 'Open' and 'Resolved'. All other states have an issue number of less than 10,000. Analyzing all the various states we can identify the success states shown in the table 3.2. These states should represent a situation where the issue is in a final state, so it won't undergo any further changes. However, for our work, analyzing only the 'Status' field does not allow us to understand if an Issue was completed successfully and therefore, in the case of a User Story, if it has been implemented.

For this reason, the 'Resolution' field must also be analyzed. The "Resolution" field is an important feature in Jira. It specifies the reason an issue is closed and removes the need of having multiple statuses with the purpose of stating why the issue is closed. Through a select query, using the "Group By" clause, we can identify 54 possible value types of the "Resolution" field. The most common status we notice is 'Fixed' with 160857 Issues, followed by 'Done' with 43256 Issues and 'Won't Fix' with 40816. Given the results of this query, shown in table 3.3, we can make several important considerations: first of all, we note the presence of 77498 Issues with a null 'Resolution' value, this means that for almost 1/5 of the Issues present in the Database the reason for the closure was not specified.

Furthermore, the most common Resolution value is 'Fixed' gives us a first clue as to the fact that most of the Issues are of the Bug and not the Story type and for this reason, a more in-depth analysis of the 'Type' field results is indispensable. We can therefore identify the following as successful reasons for closing an Issue: Fixed, Done, Answered, Complete, Handled by Support, Completed, Resolved Locally, Community Answered, Implemented, Deployed and Resolved; for a total of 224505 successfully resolved issues.

Considering that the actual successful completion of an issue is represented by the value of

the 'Resolution' field, we can consider it as the only issue selection criterion, without considering the 'Status' field.

**Table 3.3:** All different types of **'Resolution'**.

| Resolution | Number of Issues |
|---|---|
| Fixed | 160857 |
| 'null' | 77498 |
| Done | 43256 |
| Won't Fix | 40816 |
| Duplicate | 33166 |
| Won't Do | 15709 |
| Answered | 9292 |
| Complete | 7149 |
| Handled by Support | 1493 |
| Completed | 1378 |
| Resolved Locally | 707 |
| Community Answered | 116 |
| Implemented | 111 |
| Deployed | 102 |
| Resolved | 44 |

We proceed with the analysis phase with the 'Type' field. Through a select query, using the "Group By" clause, we can identify 24 possible value types of the 'Type' field. As imagined when analyzing the Resolution field, the most common problem type within the dataset is 'Bug' with 215570 issues, followed by 'Suggestion' with 96370 Issues and 'Improvement' with 42691 Issues. These characteristics confirm that the TAWOS database is mainly populated by Issues which probably represent evolutionary and corrective maintenance interventions rather than User Stories. However, we note that the fourth most common type of issue is 'Story' with 31394 Issue. So we can say that we have a total of 31394 potentially usable User Story Issues (obviously a different phase of data analysis and data cleaning will be required on User Stories only). As we can see in table 3.4 different types of Issues could approach a User Story, such as 'New Feature' and 'Improvement'.

Therefore, analysing these issues individually is necessary to understand if they can be considered User Stories.

**Table 3.4:** All different types of **'Type'**.

| Type | Number of Issues |
|---|---|
| Bug | 215570 |
| Suggestion | 96370 |
| Improvement | 42691 |
| Story | 31394 |
| Task | 28338 |
| Sub-Task | 14396 |
| New Feature | 14239 |
| Epic | 4157 |
| Enhancement Request | 3239 |
| Support Request | 2368 |
| Build Failure | 1715 |
| Question | 1396 |
| Technical task | 987 |
| Documentation | 699 |
| Test Task | 336 |
| Problem Ticket | 113 |
| Wish | 88 |
| Milestone | 44 |
| Technical Debt | 30 |
| Investigation | 22 |
| Incident | 18 |
| Test | 17 |
| Release | 4 |
| Publc Security Vulnerability | 1 |

It is important to consider the nature of this database as it could be very useful for future work such as maintenance effort prediction tasks or other Software Maintenance tasks.

Let's continue our work with the analysis of the 'Story Point' and 'Description' fields of the 'Story' type Issues only. Through the query below we analyze the 'Story' type Issues that contain a null Story Point value and do not have a description.

**Query for Story Point's and Description's null values:**

```
SELECT Count(*) FROM tawos2.issue WHERE Type = 'Story'
AND (Story Point IS NULL OR Story Point <= 0 OR
Description IS NULL OR Description='');
```

The result of this query shows that there are 12962 Issues of type Story that contain null values in the Story Point or Description fields; therefore these Issues need to be removed from the data to be used for our work. A further analysis to do is on the 'Story Point Changed After estimation' field. This binary field (0 - 1) indicates whether, since the estimation, the Story Point value has changed over time. It is logical to select those that have not changed to the Story Point value among all the issues, as the change in this value can be considered a sign of incorrect estimation. We then run a query to collect the Issues of type Story, which have a description and a Story Point value other than zero and which has not been changed (Story Point Changed After Estimation = 0).

**Query for User Story with Story Point not changed after estimation:**

```
SELECT * FROM tawos2.issue WHERE Type = 'Story'
AND ((Story Point IS NOT NULL AND Story Point > 0
AND Story Point Changed After Estimation = 0))
AND (Description IS NOT NULL AND Description != '');
```

The result of this query is 15137. We note how the increase in the selection criteria, unfortunately necessary, leads to a drastic decrease in the number of usable Issues; basics remember that we started from about half a million total Issues!

We continue our analysis phase, analyzing the distribution of Issues in the various Sprints. Therefore we create two tables: In the first, obtained with the query shown below, we obtain the total number of valid User Stories and Sprints per project. We can see that there are projects like number 2 that are not on the list. This is because there are entire projects that have no Issues of type Story, but rather have Issues of type Task and Improvement.

**Query for the Sprints/User Stories distribution among the various projects:**

```
Select tempTable.ID as 'Project', tempTable.Name, Count(*) AS 'Total US', tempTable.Total Sprint
from ( select tawos2.project.ID, tawos2.project.Name, count(*) as 'Total Sprint' from tawos2.project
inner join tawos2.sprint on tawos2.sprint.Project ID = tawos2.project.ID
group by tawos2.project.ID) tempTable
inner join tawos2.issue
on tempTable.ID = tawos2.issue.Project ID
where tawos2.issue.Type = 'Story' and ((Story Point IS NOT NULL AND Story Point > 0 AND Story Point Changed After Estimation = 0))
AND (Description IS NOT NULL AND Description != '')
group by tempTable.Name;
```

So in total, we have 19 valuable projects. Of these, we can see in table 3.5 that there are projects such as 'Alloy framework' and 'Lyrasis Dura Cloud' which have a non-optimal distribution as we notice that, in the first case, they have a low number of User Stories and a very high number of Sprints and in the second case, a low number of Sprints but a high number of User Stories. In the case of the 'Alloy framework' project, this contains 1519 Issues but of the 233 User Stories, only 9 have an estimated Story Point value. So we understand another important piece of information: that is that not all the development teams have carried out an In Story Point effort estimation operation. With the second join table we analyze the distribution of User Stories in the various sprints. We are interested in knowing what are the average US values per Sprint in the various projects. To obtain this analysis we execute a first join query which, for each Sprint of each project, collects the number of User Stories. Obviously what interests us is to have an average value to observe the distribution of User Stories in the various Sprints. So, by running the below query we can get the join table shown in figure 3.6.

**Query for the US x Sprint distribution among the various projects:**

```
select tempTable.Project ID, AVG(tempTable.Total US Sprint) as 'AVG US xSprint'
from (select tawos2.sprint.Project ID, tawos2.sprint.ID as 'Sprint ID', count(*) as 'Total US Sprint'
from tawos2.sprint join tawos2.issue
on tawos2.sprint.ID = tawos2.issue.Sprint ID where tawos2.issue.Type = 'Story' AND
((Story Point IS NOT NULL AND Story Point > 0 AND Story Point Changed After Estimation =
0)) AND (Description IS NOT NULL AND Description != '')
group by tawos2.sprint.ID) tempTable
group by tempTable.Project ID;
```

It turns out that there are some projects that have Project and Sprint mapped with keys in their tables, but the Issue table tuples for these projects have 'null' in the Sprint ID field so they are not linked. An example is project ID 8 which contains 531 User Stories but only one has a value in the Sprint ID field. This leads to an alteration of the mean value. Ultimately we can see a total range that goes from 1.1 to 16.2 User Stories per Sprint with a total average, across all projects, of 3.47 US x Sprint.

**Table 3.5:** User Story/Sprint distribution.

| Project | Number of Stories | Number of Sprints |
|---|---|---|
| Spring XD | 1840 | 66 |
| Sonatype Nexus | 86 | 143 |
| Apache Mesos | 20 | 227 |
| Apache Usergrid | 152 | 38 |
| Apache MXNet | 17 | 41 |
| Alloy Framework | 9 | 118 |
| Aptana Studio | 130 | 12 |
| Command-Line Interface | 23 | 98 |
| Appcelerator Daemon | 18 | 62 |
| Titanium Mobile Platform | 111 | 217 |
| Hyperledger Indy Node | 70 | 76 |
| Hyperledger Fabric | 240 | 142 |
| Lsstcorp Data management | 10900 | 396 |
| Lyrasis Dura Cloud | 279 | 7 |
| MongoDB Compass | 27 | 91 |
| Mule APIkit | 61 | 124 |
| Mule | 43 | 311 |
| Appcelerator Studio | 559 | 163 |
| The Titanium SDK | 327 | 301 |

**Table 3.6:** AVG of USxSprint per project.

| Project ID | User Stories x Sprint |
|---|---|
| 1 | 16.24 |
| 3 | 1.48 |
| 4 | 1.23 |
| 5 | 3.31 |
| 6 | 1.75 |
| 7 | 1.50 |
| 9 | 1.10 |
| 10 | 1.57 |
| 11 | 1.69 |
| 12 | 2.16 |
| 13 | 3.16 |
| 24 | 2.43 |
| 25 | 1.86 |
| 28 | 12.63 |
| 29 | 5.33 |
| 30 | 1.37 |
| 35 | 2.33 |
| 36 | 1.44 |

The last step that remains for us to complete for this phase of analysis is the construction of the various datasets that we will use to train our models and carry out the experiments. In particular, using all the information collected so far, on the various fields, we build the first dataset for Story Point estimation. The tuples contained in this dataset will be selected according to the following selection criteria listed:

1. **Type** = 'Story'
2. **Resolution** = ['Completed', 'Complete', 'Fixed', 'Done', 'Resolved', 'Implemented', 'Answered', 'Handled by Support', 'Resolved Locally', 'Deployed', 'Community Answered']
3. **Story Point Changed After Estimation** = 0
4. **Story Point** > 0
5. **Description** not empty.

for a total of 12864 issues.

To answer RQ3 and RQ4 we need to create three different datasets, one for the dependent variable 'In Progress Minutes', 'Total Effort Minutes' and 'Resolution Time Minutes'. The reason for the creation of these three datasets is the fact that we want to know with which dependent variable among these, the model can generalize better. Running queries to analyze how many issues have a value for each of these variables, we can see that, considering the selection criteria listed above, only 304 Issues have a Timespent value > 2. For this reason, we will not consider the Timespent variable as a dependent variable.

For the dependent variable "In Progress Minutes," we run a select query, including the constraints listed above. To avoid considering too simple tasks, we select Issues that have a value of 'In Progress Minutes' > 2. Moreover, considering that we want to build Cross-Project estimation models, therefore trained in such a way as to be able to estimate the effort of a User Story of a new project, we want to use projects that have several US greater than 150. For this reason, first of all, we analyze the number of US, which comply with the constraints, per project and then we select only the projects with more than 150 User Stories. We get a total of 11142 User Stories.

For the 'Total Effort Minutes' variable, we perform the same selection operation as for the Issues, obtaining a dataset of 13866 User Stories.

For the 'Resolution Time Minutes' variable, we perform the same selection operation as for the Issues, obtaining a dataset of 19221 User Stories.

### 3.3.2 Literature Models Analysis

We now proceed with the analysis of the latest Machine Learning models proposed in the literature, constituting the state of the art. So far, two models have been proposed that use Deep Learning for effort estimation in Story Points: DeepSE and GPT2SP. DeepSE is a model proposed by Morakot Choetkiertikul et al. in 2019, which uses LSTM and RNN networks to perform a regression task. In this case, the estimation is considered a regression task, which is not apparent since in other previous works the effort estimation in Story Point was treated as a Classification task [21], where each class belongs to an element of the Fibonacci sequence in the range 1 - 40. Due to the nature of the estimation task, we believe that the value of the effort estimated by a model should not be limited to a set of classes, but rather that a model capable of making precise estimates should be built. Both the DeepSE and GPT2SP models use Natural Language Processing, a set of artificial intelligence algorithms capable of analyzing, representing and therefore understanding natural language, to extract the context from the title and description of an Issue. This technique is certainly valid and, as the results show, allows for good performance; however, as suggested in the replies made by V. Tawosi et al. [14, 16], it is necessary to analyze the use of other components available directly from Jira.

Therefore, in the next steps, we proceed with the implementation of the various models that will allow us to answer the posed research questions, to provide important results in the literature and new foundations for future work.

### 3.3.3 RQ1: Story Point Baseline Model Building

*Is the BeForE model suitable for estimating Story Points of a User Story in Agile Projects?*

The objective of this research question is to propose a new effort estimation model in Story Point that uses only textual information. To answer this question, let's build a model using Bert, a pre-trained language model from Google. To build and train the model we will use Colab Pro, an enhanced version of Colab, which allows us to access more performing GPU resources. To train our model we use Tensorflow, a Python library for building neural networks.

#### Data Pre-processing

The first step to answer this question, after creating the environment in Colab Pro, is the Data Pre-processing phase. This phase is critical and can be decisive for obtaining a good model. First, we load the dataset extrapolated in the data analysis phase and eliminate the columns we will not use, i.e. all except the Title, Description, Status, Project_ID, Resolution and the Story Point target variable. Since the filtering of the User Stories is done, I don't take into account the value of the

Status field, we perform a quick analysis of this through the value_counts() method of pandas. We notice that two states 'Won't Fix' and 'Invalid' represent a failure state of the User Story. For the initial assumptions, we eliminate the issues with these two statuses, reducing the size of the dataset by 877 User Stories. As the next step, we carry out the analysis of the Project_ID field through the value_counts() method. We need this step to ensure that the model trains on projects that have a relatively large number of User Stories. For this reason, we eliminate the projects that have several User Stories lower than 150, i.e. the following projects: 5, 8, 11, 3, 25, 36, 30, 9, 10, 4, 7 and 6.

We perform sanity checks to check that there are no null values in the selected fields. A sanity check or sanity test is a basic test to quickly evaluate whether a claim or the result of a calculation can be true. It is a simple check to see if the produced material is rational (that the material's creator was thinking rationally, applying sanity). The point of a sanity test is to rule out certain classes of obviously false results, not to catch every possible error. A rule-of-thumb or back-of-the-envelope calculation may be checked to perform the test. Let's remove the quotation marks at the beginning and at the end from the Title and Description fields to have only the part of the text to be used for our template. We then select the three columns that will make up our training and testing dataset: Title, Description and Story Point. For training this model we will reserve 20% of the data for testing and 80% for training.

**Bert Loading**

The next step is to load the pre-trained model Bert. As shown in the state of the art, Bert uses a subword tokenizer (WordPiece), so the maximum length corresponds to 512 subword tokens. We have to set a Token Sequence Length for Bert by analyzing our sentences, and therefore the sequences that we will input to Bert. To set this length, it is, necessary to explore the Title's length distribution and Description as well. Through a histogram, we visualize the distribution of the lengths of the two fields with the average value and the maximum value. We see that for the title we have an average length of 7 words, with a maximum of 38; for the description, we have an average length of 59 words, with a maximum of 1175. We specify that the value of the Token Sequence Length that we set for the construction of the first model will be the same for all the other models, likewise, the architecture, to allow comparing the performance of models in a way that these do not depend on the parameters of the model.

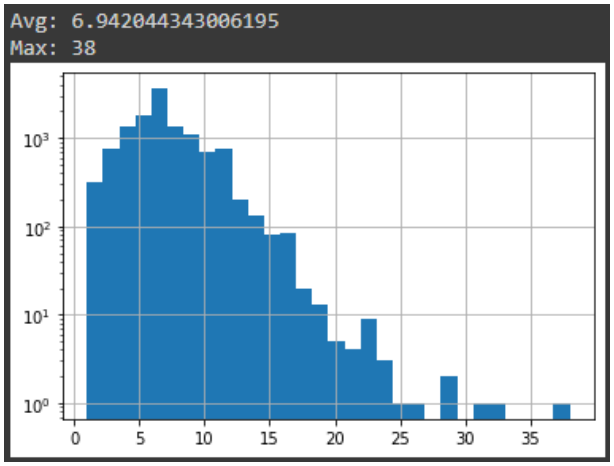



**Figure 3.4:** Title's length distribution.

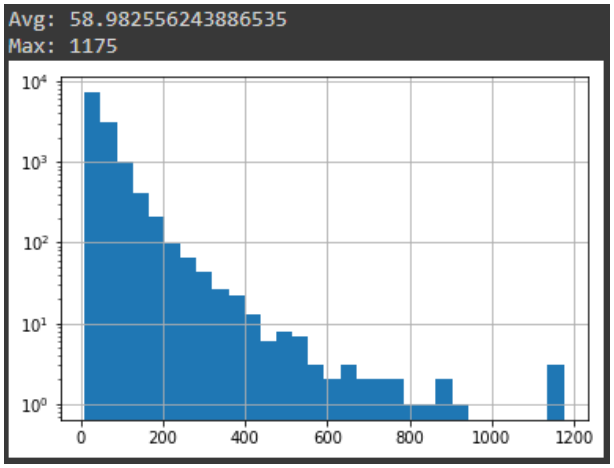



**Figure 3.5:** Description's length distribution.

To set a Token Sequence Length value, we add the 95th percentile values of Title and Description and add a small margin of difference we obtain a value of 256, which is positioned in the middle of 512. Setting the length of the sequence too high makes the tokenization more complex, obviously lengthening both the time to perform the fine-tuning of Bert and to train the final regressor. We load the pre-trained model 'bert-base-uncased', which represents the standard Bert base model working on sentences written in the English language. This basic model has several training parameters equal to 110 million, unlike the large version which has 340 million. Now we initialize the Input Token Identification variable. This argument indicates to the model token indices that numerical representations of tokens build the sequences. Initialize the Attention Mask variable. This argument indicates to the model which tokens should be attended to, and which should not. Let's now fine-tune the Bert model. This phase consists of the generation of a Bert-understandable dataset. We initialize the training target variable by creating a NumPy array with all the training user stories' target values. In this phase, for this model, we do not carry out

any normalisation/standardisation of the target variable, as this range is not particularly large. Furthermore, we want to analyze some error metrics in Story Point. Once the Bert-understandable dataset has been generated, we separate 20% of the training dataset, which we will use as a validation set during training, and proceed with constructing the neural network.

**Neural Network Architecture and Training**

As a first step for the construction of our neural network, we instantiate three Keras Input tensors through layers.Input(). A Keras tensor is a tensor object from the underlying backend (Theano or TensorFlow), which we augment with certain attributes that allow us to build a Keras model simply by knowing the inputs and outputs of the model.

For example, if a, b and c are Keras tensors, it becomes possible to do:

```
model = Model(input=[a, b], output=c)
```

These three tensors will form the level of Bert embeddings which will then go into input to the network's hidden layers. In particular, we have 3 SimpleRNN levels: Recurrent neural networks (RNN) are a class of neural networks that is powerful for modelling sequence data such as time series or natural language. Schematically, an RNN layer uses a for loop to iterate over the timesteps of a sequence, while maintaining an internal state that encodes information about the timesteps it has seen so far. Each RNN level is connected to the previous level, so the first SimpleRNN will receive the berd_embds level as input; the second level SimpleRNN will receive the first intermediate level as input and the third level SimpleRNN will receive the second intermediate level as input. Finally, we find a Dense level. A dense layer also referred to as a fully connected layer is a layer that is used in the final stages of the neural network. This layer helps in changing the dimensionality of the output from the preceding layer so that the model can easily define the relationship between the values of the data in which the model is working. The dense layer does the following operation on the input and returns the output.

```
output = activation(dot(input, kernel) + bias)
```

where:

1. **input** represent the input data
2. **kernel** represent the weight data
3. **dot** represent NumPy dot product of all input and its corresponding weights
4. **bias** represent a biased value used in machine learning to optimize the model

5. **activation** represent the activation function.

For our task, the ReLU function was chosen as the Dense level activation function. It has recently become popular due to its relatively simple calculation. This helps speed up neural networks and appears to perform empirically well, making it a good starting choice for the activation function. The ReLU function is a simple function, which can also be thought of as a piecewise function with all inputs less than 0 mapped to 0 and all inputs greater than or equal to 0 mapped to themselves Finally, we have a Dense level of output which will return the true value of the regression.

This architecture, as mentioned above, will be used to answer all research questions and will not change. Before proceeding with the training of the network it is necessary to set the following Hyperparameters: Learning rate of the optimizer (Adam in the case of Bert as indicated in the official paper), the number of Epochs, the loss function that the model will use to train and finally the metrics to evaluate the performance.

As several epochs, we set the maximum among the values suggested by the developers of the Bert model; as learning rate, we set 5e-5, as loss function we use MeanAbsoluteError and as metrics we use both MeanAbsoluteError, MeanSquaredError and RootMeanSquaredError. We chose MAE as the loss function because it is essentially insensitive to outliers (provided there aren't too many). This is because it is the median of all absolute values of the residuals and the median is unaffected by the values at the tails. Then, this loss function can be used to perform robust regression. Also, the setting of the Hyperparameters, will not be modified in the various pieces of training. Now we can run the training and evaluate its performance to answer RQ1.

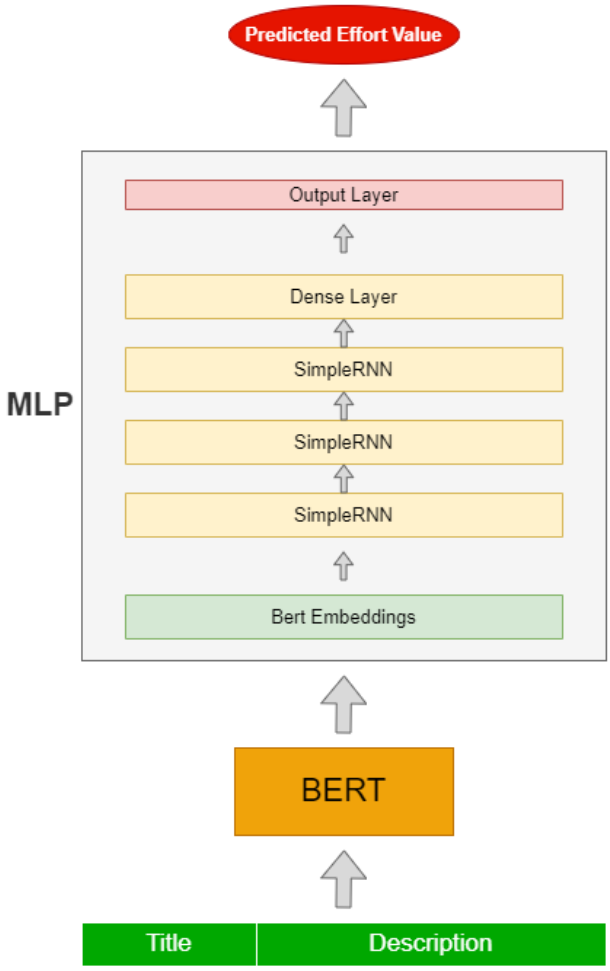



**Figure 3.6:** Overview of model's architecture.

### 3.3.4 RQ2: Feature Augmentation on Story Point

*Does the addition of secondary features allow the BeForE model to obtain better Story Point estimations?*

The objective of this research question is to test whether the addition of secondary features, extracted from Jira, improves the performance of the Story Point estimation model. To answer this question, let's build a model using Bert, a pre-trained language model from Google. To build and train the model we will use Colab Pro, an enhanced version of Colab, which allows us to access more performing GPU resources. To train our model we use Tensorflow, a Python library for building neural networks. To answer this question we will carry out all the steps performed for RQ1, with the only difference that after the Data Pre-processing phase we will carry out a feature augmentation operation.

#### Feature Analysis

We begin by carrying out a feature analysis. Our goal is to select the features that may be most

relevant to experience the performance increase. In this regard, let's proceed with the creation of a very useful matrix, namely the corr_matrix. This matrix, which can be generated using the Pandas library, has a dimension of n x n (where n is the number of features), and in each cell, it calculates the Pearson correlation coefficient between the two features.

So in cell (i,j), we will have the Pearson correlation coefficient between the variable i and the variable j. Pearson's correlation coefficient is interpreted as follows: if the value is close to +1 it means that there is a positive correlation between the variables, ie as one increases the other increases; if the value is close to -1 then it means that there is a negative correlation between the variables, ie as one variable decreases, the other decreases. However, if the values are close to 0, then there is no correlation between the variables.

We then generate the correlation matrix and analyze the results of the Story Point column. In this column, we see all the correlation coefficients between each variable and the Story Point variable. As we note, there are no values very close to +1 and -1. The highest values are Priority and Project_ID, therefore we carry out a feature augmentation operation including these two features in our model. In addition to the numerical correlation between these features, it must be considered that the Bert model, working on context and semantic similarity, could use these new features to extract new semantic information, such as the estimation policy adopted by a team.

**Table 3.7:** Pearson's Correlation Matrix on Story Point.

| Feature | Story Point |
|---|---|
| Priority | 0.0007 |
| Story Point | 1.0000 |
| Timespent | -0.0001 |
| In Progress Minutes | 0.0002 |
| Total Effort Minutes | -0.0001 |
| Resolution Time Minutes | 0.0006 |
| Title Changed After Estimation | -0.0006 |
| Story Point Changed After Estimation | -0.0002 |
| Creator ID | -0.0028 |
| Reporter ID | -0.0028 |
| Assignee ID | -0.0014 |
| Project ID | -0.0031 |
| Sprint ID | -0.0003 |

**Feature Augmentation**

Let's proceed with the feature augmentation operation. To integrate the new features into the model, we will insert these as additional sentences to the description. In particular, we will add a sentence to specify the priority of the User Story and a sentence to specify the project to which it belongs. For example, for a 'High' priority and a Project_ID equal to 3, the description will be increased as follows:

```
"This User Story belongs to project 3 and has High priority. ..."
```

**Re-training**

In this step, we perform all the steps for building and training the model performed in RQ1. Therefore, we supply the augmented dataset to the bert model and subsequently, with the same hyperparameters of the MLP we train our network.

### 3.3.5 RQ3: Minutes Effort Model Building

*Is the BeForE model suitable for estimating Effort Development in minutes of a User Story in Agile Projects?*

The goal of this research question is to propose a new model for estimating effort in minutes using only textual information. To answer this question, we create three models, one for each target variable 'In Progress Minutes', 'Total Effort Minutes' and 'Resolution Time Minutes' and compare them to understand which of the three variables the model generalizes best. We consider the first two research questions as a baseline since according to the paper by V. Tawosi et al. [19] on the correlation between SP and development time, and the replicas of the DeepSE and GPT2SP models, we want to experiment with a new type of model, which does not estimate in Story Points, but in minutes, and also, in this case, verify if the addition of other features leads to better performance.

For this reason, we respond to RQ3 by carrying out three trainings respectively with the three datasets extrapolated in the Data Analysis phase. The architecture of the model that we will use to answer this research question is the same one used to answer RQ1 and RQ2. The only difference is in the handling of the target variable. In this case, we are estimating in minutes and as we can see the three variables can have a value ranging from 2 minutes to over 1.2 million minutes. For this reason, it is necessary to standardize the target variable to obtain better performances.

**Target Variable Standardization**

As a first step, to reduce the size of the values, we convert minutes to days, dividing each value by 1440. We then apply a scaling operation. Deep learning neural network models learn a mapping from input variables to an output variable. Therefore, the scale and distribution of data extracted from the domain can be different for each variable.

Input variables can have different units (for example, feet, kilometres, and hours) which, in turn, can mean that the variables have different scales. A target variable with a large spread of values, in turn, can lead to high error gradient values that cause the weight values to change drastically, making the learning process unstable. There are two types of data scaling you might consider: normalization and standardization. Normalization is a scaling of the data from the original range so that all values are between 0 and 1, also called min-max scaling. A value is normalized as follows:

```
y = (x - minimum) / (maximum - minimum)
```

This might be useful in some cases where all parameters need to have the same positive scale. However, the dataset outliers are lost. For this reason, we consider standardization. This technique scales the data to have a mean of 0 and a standard deviation of 1. Then we perform standardization of the NumPy array through the np.linalg.norm() function thus obtaining the effort values in scaled days.

Reducing the scale of the target variable will, in turn, reduce the size of the gradient used to update the weights and result in a more stable model and training process.

**Models Training**

In this phase we perform three trainings retracing the same steps performed to answer the research question RQ1, then we set the parameters of bert and our network and we perform the training with the same loss function and the same evaluation metrics on the three dependent variables. In this case, the three models will have an overall dataset different from each other, due to the different target variable and the consequent filtering performed in the projects as described in the analysis phase.

The results we would obtain will be resized so to obtain the various error values in minutes it will be sufficient to simply carry out the reverse operation of the standardization and multiply the result by 1440 to transform the value in days back into minutes.

### 3.3.6 RQ4: Feature Augmentation on Minutes Effort

*Does the addition of secondary features allow the BeForE model to obtain better Effort Development estimations?*

The objective of this research question is to test whether the addition of secondary features, extracted from Jira, improves the performance of the Minutes Effort estimation model. To answer this question, let's build a model using Bert, a pre-trained language model from Google. To build and train the model we will use Colab Pro, an enhanced version of Colab, which allows us to access more performing GPU resources. To train our model we use Tensorflow, a Python library for building neural networks. To answer this question we will carry out all the steps performed for RQ3, with the only difference that after the Data Pre-processing phase we will carry out a feature augmentation operation.

**Feature Analysis**

We begin by carrying out a feature analysis. Our goal is to select the features that may be most relevant to experience the performance increase. In this regard, let's proceed with the creation of a very useful matrix, namely the corr_matrix. This matrix, which can be generated using the Pandas library, has a dimension of n x n (where n is the number of features), and in each cell, it calculates the Pearson correlation coefficient between the two features.

Pearson's correlation coefficient is interpreted as follows: if the value is close to +1 it means that there is a positive correlation between the variables, ie as one increases the other increases; if the value is close to -1 then it means that there is a negative correlation between the variables, ie as one variable decreases, the other decreases. However, if the values are close to 0, then there is no correlation between the variables.

In this case, we build three correlation matrices, one for each target variable 'In Progress Minutes', 'Total Effort Minutes' and 'Resolution Time Minutes' so that we can analyze the Pearson correlation coefficients between each independent variable and the target variable. So in cell (i,j), we will have the Pearson correlation coefficient between the variable i and the variable j. From the results, also in this case as for RQ2, the variables with high correlation are Priority and Project_ID.

**Feature Augmentation**

Let's proceed with the feature augmentation operation. To integrate the new features into the model, we will insert these as additional sentences to the description. In particular, we will add a sentence to specify the priority of the User Story and a sentence to specify the project to which it belongs. For example, for a 'High' priority and a Project_ID equal to 3, the description will be

increased as follows:

> "This User Story belongs to project **3** and has **High** priority. ..."

At the end of this feature augmentation phase, we notice a drop in the number of User Stories. In particular, we go from an average of 13,000 User Stories per dataset for RQ3 to an average of 2,300 User Stories per dataset used for this research question. The reason for reducing the number of items in the datasets is that not all User Stories contain a not null value in the Priority and Project_ID fields.

This reduction could affect the overall performance of the model, even if in general Bert can learn even with a not-too-large dataset.

**Re-training**

In this step, we perform all the steps for building and training the model performed in RQ3, for our three models. Therefore, we supply the augmented dataset to the bert model and subsequently, with the same hyperparameters of the MLP we train our network.

# CHAPTER 4

# ANALYSIS OF THE RESULTS

This chapter illustrates the achieved results for each research question and the conclusions regarding the main objective.

## 4.1 $RQ_1$ - BeForE Story Point model

> **$RQ_1$**. Is the BeForE model suitable for estimating Story Points of a User Story in Agile Projects?

As we said in section 3.1, the purpose of research question RQ1 is to propose a new Story Point estimation model that uses only the title and description of a User Story. To answer this question we have created an MLP neural network, using the pre-trained language model Bert. In table 4.1 we can observe the results of the various averages of the metrics obtained at the end of the training. Recall that they are Story Point values. Based on the results obtained from the model we can say that it is certainly a good performing model and that, observing the training curves 4.1, it was able to generalize well. We can see, by comparing ourselves with the other models present in the literature, that the average values obtained from our BeForE model are even better than these. However, we cannot officially say that our model is better than DeepSE and GPT2SP as it was trained on a different dataset considering our study context which focuses on User Stories only.

As we can see we have a MAE value of 1.95, therefore an average absolute error of less than 2 story points, while we note a MSE value equal to 18.45 and RMSE equal to 4.30.

The reason why the MSE and RMSE values are higher than the MAE value is because the mean absolute error is a linear score, which means that all individual errors are weighted equally

on the mean; while, RMSE and MSE are higher as by squaring the differences in predictions, they are more sensitive to outliers. Therefore, the more widespread the outlier values are in the dataset, the more they affect the value of MSE and RMSE. In general, in most cases there is a value of RMSE > MAE, however it is important that the difference between these values is not very large. The greater the difference between them, the greater the variance of individual errors in the sample. In our case the RMSE value is about twice the MAE value, so not too big. In an optimal, but very rare case, MAE=RMSE if all errors are of the same magnitude.

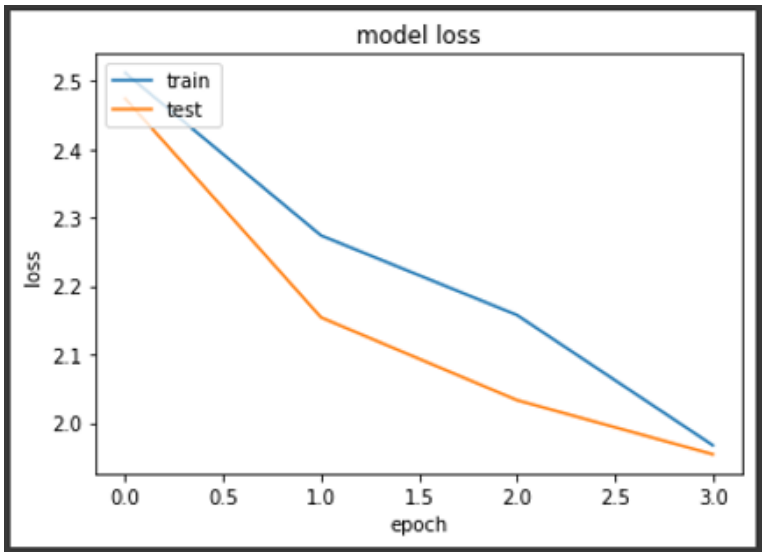



**Figure 4.1:** BeForE Story Point model's Loss Curve.

**Table 4.1:** RQ1 Model's metrics

| MAE | MSE | RMSE |
|---|---|---|
| 1.95 | 18.45 | 4.30 |

## 4.2 $RQ_2$ - Feature Augmentation on BeForE Story Point model

**$RQ_2$.** Does the addition of secondary features allow the BeForE model to obtain better Story Point estimations?

As we said in section 3.1, the purpose of research question RQ2 is to verify whether the use of other secondary features leads to an increase in model performance. To answer this question we created an MLP neural network, using the pre-trained linguistic model Bert and augmenting

the textual description of the User Story with the Priority and Project_ID features, selected after analyzing the correlation coefficients In the 4.2 table we can observe the results of the various averages of the metrics obtained at the end of the training. Recall that they are Story Point values. Analyzing the results we can see that there is no increase in performance compared to the baseline model. We have a MAE value of 2.60, an MSE value of 22.03 and a RMSE value of 4.69, all values higher than the respective metrics of the RQ1 model. These results tell us that the model failed to generalize better, using new features, but even more it got worse. A first reason could certainly be due to the fact that, as seen in the corr_matrix, no feature is sufficiently correlated to the target variable, in our case Story Point. For this reason, providing a model with additional information to extrapolate the context (in our case Priority and Project_ID) which are not representative, does not allow for better estimates. These results are very surprising because it was thought that, being the Story Points a subjective metric between the various teams, the Bert model could learn a sort of 'policy' to make better estimates.

**Table 4.2:** RQ2 Model's metrics

| MAE | MSE | RMSE |
|---|---|---|
| 2.60 | 22.03 | 4.69 |

In figure **??** we can see the performance of the two models compared. In addition to noting the curves of the RQ2 model positioned higher than those of the RQ1 model, we also note a smaller loss decrease during the 4 periods of the settlement. So the model had difficulty generalizing with the new features compared to the first training. However, we note that at the end of the first epoch the two models reached a loss value of 2.7 and 2.5 respectively, therefore not exaggeratedly distant, but during the following epochs, we see an evident difficulty of the RQ2 model in trying to improve the estimate, to then stop at the threshold of 2.6.

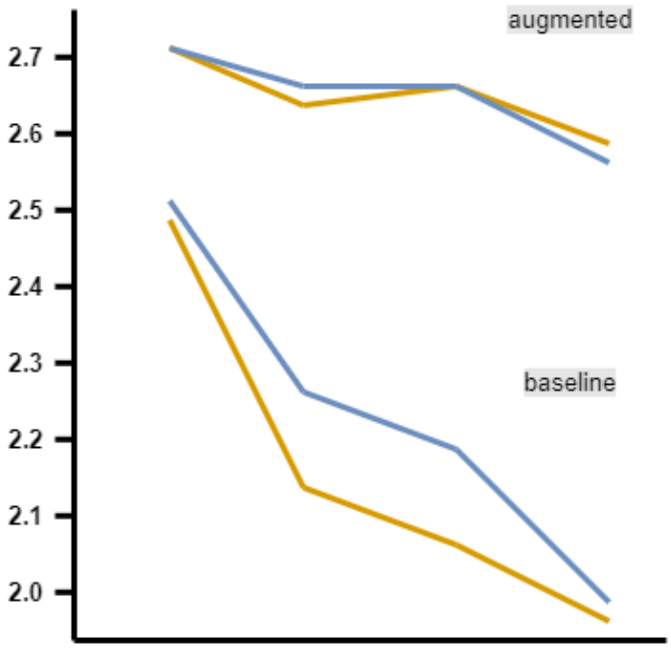



**Figure 4.2:** comparison between the RQ1 and RQ2 models.

## 4.3 $RQ_3$ - BeForE Minutes Effort model

> **$RQ_3$.** Is the BeForE model suitable for estimating Effort Development in minutes of a User Story in Agile Projects?

As we said in section 3.1, research question RQ3 aims to propose a new type of effort estimation model which predicts development effort in minutes. To answer this question we created three models, one for each target variable 'In Progress Minutes', 'Total Effort Minutes' and 'Resolution Time Minutes' and trained them using only the Title and Description features. The models built all reference the **??** architecture and then use the Bert language model and an MLP to perform the regression. In table **??** we can see the results of the three models. The model trained with the target variable In Progress performed best while the model that estimates in 'Resolution Time Minutes' performed 'worst'. Recall that the 'In Progress Minutes' variable represents the time that an Issue spends in the In Progress state. This should be, by the study by V. Tawosi et al. the most representative variable of the development effort of an Issue, therefore the fact that the model has managed to generalize better than the others is certainly a positive aspect. The reason why the model that estimates the In Progress variable is the best is probably due to a reduced number of outliers.

We recall that all three target variables, before performing the workouts, were first converted into days and then standardized. This operation, as explained in paragraph 3.3.5, was carried out

to reduce the height difference used upgrade the weights and get a more stable training pattern and process. So if you want to see errors in minutes, you just need to do the following steps:

```
original_days_value = MAE/MSE/RMSE * (np.linalg.norm(MAE/MSE/RMSE))
minutes_value = original_days_value * 1440
```

**Table 4.3:** RQ3 Model's metrics

| Target Variable | MAE | MSE | RMSE |
|---|---|---|---|
| In Progress Minutes | 0.00400 | 0.00013 | 0.01150 |
| Total Effort Minutes | 0.00520 | 0.00097 | 0.09939 |
| Resolution Time Minutes | 0.00817 | 0.00099 | 0.10000 |

## 4.4 $RQ_4$ - Feature Augmentation on BeForE Minutes Effort model

**$RQ_4$**. Does the addition of secondary features allow the BeForE model to obtain better Effort Development estimations?

As we said in section 3.1, the purpose of research question RQ4 is to verify whether the use of other secondary characteristics leads to an increase in model performance. To answer this question we created an MLP neural network, using the pre-trained linguistic model Bert and integrating the textual description of the User Story with the Priority and Project_ID features, selected after analyzing the correlation coefficients In the **??** table, we can observe the results of the various averages of the metrics obtained at the end of the training.

Analyzing the results we can see that there is a deterioration in performance compared to the baseline models. These results tell us that the model failed to generalize better, using new features, but even more, it got worse. A first reason could certainly be because, as seen in the corr_matrix, no feature is sufficiently correlated to the target variable, in our case Story Point. For this reason, providing a model with additional information to extrapolate context (in our case Priority and Project_ID) that are not representative, does not allow for better estimates.

We can see from the results that also, in this case, the model trained on the variable 'In Progress Minutes' turns out to be the most performing, while we can see a difference between the

performance of the 'Total Effort' model and 'Resolution Time' reduced compared to the situation in the RQ3.

**Table 4.4:** RQ4 Model's metrics

| Target Variable | MAE | MSE | RMSE |
|---|---|---|---|
| In Progress Minutes | 0.00870 | 0.00066 | 0.02384 |
| Total Effort Minutes | 0.00915 | 0.00385 | 0.19633 |
| Resolution Time Minutes | 0.00975 | 0.00402 | 0.18369 |

We recall that all three target variables, before performing the workouts, were first converted into days and then standardized. This operation, as explained in paragraph 3.3.5, was carried out to reduce the height difference used upgrade the weights and get a more stable training pattern and process. So if you want to see errors in minutes, you just need to do the following steps:

```
original_days_value = MAE/MSE/RMSE * (np.linalg.norm(MAE/MSE/RMSE))
minutes_value = original_days_value * 1440
```

# CHAPTER 5

# OBSERVATIONS AND LESSONS LEARNED

This chapter illustrates observations on the research and the lessons learned for the research community.

## 5.1 Models and Experiments Observations

We recall that the objective of our study, specified with the four research questions, was to propose a new estimation model, both of Story Points and Development Minutes, to then experiment with whether the addition of secondary features, extracted from the Tawos database, could improve performance. To answer these research questions, a quantitative study was conducted which involved the creation of a total of 8 different models, all built with the same architecture based on the Bert language model and RNN networks. These models have all been trained on chronologically ordered User Stories, to train on historical data and test on more recent data. In particular, the entire set of data used for each training was divided into 60% training, 20% validation, and 20% testing.

The results achieved in this study are very interesting. First of all, we offer a new estimation approach in the literature which, with a suitable comparison, could overcome the state of the art constituted by the GPT2SP model. To verify this, it would be enough to train a replica of the GPT2SP model on the data used to train our BeForE model and compare the performances. This would also allow us to observe which of the two most famous language models, GPT-2 and

Bert, performs better in the execution of this NLP regression task. A further important result is the negative outcome of the two research questions RQ2 and RQ4, whose experiments show that the addition of the secondary features, Priority and Project_ID, do not improve the model's performance but rather worsen it. This allows us to advance the state of the art since, in the replies made by V. Tawosi et al., the research team suggests building models that are not only based on semantic similarity but also other characteristics, in particular, extracted from the TAWOS database. Our analyzes carried out on the dataset's Features show that effectively of all the fields that make up the Issue table, none has a sufficiently high correlation (measured with the Pearson coefficient) to make the estimate more accurate.

This is not to say that the assumption made by the researchers is wrong, but certainly, it is necessary to operate Feature Engineering to obtain better predictive attributes. In particular, we believe that additional features that should be used in conjunction with textual information should indicate the complexity of the task and perhaps, also the productivity and experience of the team. Two identical User Stories but belonging to different projects and contexts, such as: "As an administrator, I want access to my control panel" will appear to the model as identical, when in reality they could have very different complexities because, for example, in the first case it is a system in which the administrator has few responsibilities, while in the second case it is a system in which the administrator has many tasks and functions.

Therefore, providing an indicator, even categorical, on the complexity of the User Story, in terms of the number of software components involved, for example, could certainly prove to be an excellent strategy.

Estimating in Story Point remains a difficult task today as it is a metric whose use and interpretation varies from team to team and for this reason the prediction performed by a model (using only the description), in the case of projects that have just started, will not be never perfectly aligned with the team's esteem policy, as he had no way to learn from the team's past experiences. The situation is different if instead, we are talking about User Stories or Issues that represent maintenance operations. In this case, the model, training on the historical data of the single team, can learn their policy and estimate a value close enough to what the team would estimate for an Issue carried out on the same project. Another observation we can make on the experiments carried out is that in the case of the estimation in minutes, the Feature Augmentation operation led to a drastic decrease in the number of entries in the dataset, therefore it is necessary to collect other data to repeat the training with multiple User Stories.

## 5.2 Lessons Learned

In this section, we report some possible lessons learned that could be useful to IT workers and the research community. Specifically, we observed that:

1. **Improving Story Point Estimation:** Considering that analyzing the dataset, it often happened that following the Story Point estimate, this was changed as it was considered incorrect. This tells us that team members, Scrum Masters and Project Managers should improve the way they use Story Points to make estimations and maybe also use other techniques, which do not use Story Points, for example, an estimation model in minutes, like our BeForE.

2. **Jira Issue Fields:** Another lesson we learned from this study is that the fields accompanying a Jira Issue don't provide meaningful information to make a better prediction. Priority, Project_ID, Rank etc. do not directly influence the complexity of a User Story, and therefore the effort. A team might decide to set a very complex Issue to a low priority or a low priority for a less complex One.

3. **Using language models like Bert:** Leveraging pre-trained language models like Bert as the basis for building an MLP can be considered a great approach considering their power and the performance that can be achieved. While creating the models for the RQ2 and RQ4 research questions, we integrated the two features Priority and Project_ID in the description of the User Story, extending this with two more sentences containing the two pieces of new information. This method allows you to exploit the potential of pre-trained models such as Bert or GPT-2/GPT-3. However, one could also consider including non-text features as single fields of the training dataset, therefore not added to the description, to see if the built MLP network can learn better.

# CHAPTER 6

# THREATS TO VALIDITY

This chapter illustrates the threats to the validity of the study and the way we mitigated them.

## 6.1 Threats to Construct Validity

Threats to the validity of the construction related to the quality of the data used to build the models. Previous studies have highlighted concerns that dataset quality may affect the accuracy of machine learning models. In particular, these regression models can be inaccurate if the values of the target variable exhibit noise. These rumours, in most cases, have to do with the heuristic method used to generate the truth data.

The data used in our study is tagged with data from the Jira management tool, thus not subject to any heuristics. Especially in the case of research questions RQ3 and RQ4, the three variables used as targets are information that Jira automatically stores in their respective issue fields. The only aspect that should be verified is whether the team's developers could carry out the task state shifts promptly, without wasting time. However, these are important open-source projects, such as Spring and Apache projects, so we assume that the state changes occurred promptly without wasting time. Thus, the quality of the User Story datasets may not pose a significant risk to our experimental results.

## 6.2 Threats to Internal Validity

Threats to internal validity concern the setting of BeForE Hyperparameters. Our BeForE model has several Hyperparameters, ranging from the number of intermediate levels, with the respective number of neurons per level, to the parameters of the pre-trained Bert model. Some studies highlight concerns that different hyperparameter configurations can lead to different results. After training has been completed, the inner workings of a neural network are deterministic, not stochastic. However, the training algorithm is non-deterministic, which means that the parameter values obtained after one training process are very likely to be different from what you will get after another training process, even when the training data is the same. This is actually why training a complex neural network is a bit of an art and can often involve quite a lot of error as some training trips lead to poor results or fail to converge. Instead, as regards the setting of the hyperparameters of the Bert language model, this does not constitute a threat to internal validity since, as described in the original paper, the creators of the model have specified only 3 parameters that need to be set, namely: Batch Size, Learning-rate and number of epochs. The objective of our study is not to find the best configuration of the parameters, but rather to verify the increase in performance following the addition of features. About the technique used for adding the two features in the description, this is not said to be the best technique but, as mentioned in the comments, one could try to consider the additional features individually in the training dataset. Thus, the accuracy of our models, measured with the MAE, MSE and RMSE metrics served as the lower bound of our proposed approach, which can be further improved through optimization of the hyperparameters and the architecture of the network itself. To mitigate this threat, we release all hyperparameter settings in the replication package to facilitate future replication studies.

## 6.3 Threats to External Validity

Threats to external validity relate to the generalizability of the accuracy of our BeForE approach. The results of the evaluation of our model are based, in the first two research questions, on two datasets of approximately 13,000 User Stories respectively. Instead, the results of RQ3 are based on 3 datasets (one per target variable), respectively 11142, 13866 and 19221. Finally, the results of RQ4 are based on three datasets of approximately 2300 Issues. Therefore, the results may not be generalized concerning other datasets and other open-source projects and for this reason, other datasets can be considered for use in future work.

# CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

This chapter illustrates a summary of the research and future prospects.

## 7.1 Conclusions

This thesis reports a quantitative study to advance the use of Machine Learning techniques for effort estimation in Agile projects. Our main intentions can be summarized in the following sentences:

- We want to provide a new effort estimation model in Story Point using an innovative language model like Bert.
- We want to provide a new model for estimating the effort in minutes, representing the actual development time, using an innovative language model such as Bert.
- We want to report the consequence of using other secondary features, in addition to the title and description, in predicting effort in the two ways described above.

The research approach consists of a quantitative study, which consists of constructing an MLP neural network that uses the output of the Bert model to perform a regression. To evaluate and compare the various models, MeanAbsoulteError, MeanSquaredError and RootMeanSquaredError were chosen as metrics. Furthermore, to obtain models that can estimate the US effort of new projects, and not necessarily maintenance US; the User Stories were sorted in chronological order before each training, to train the models on the old US.

The quantitative approach shows that by using the Bert language model to extrapolate contextual information from a User Story, it is possible to obtain excellent performances in estimating both Story Points and Minutes needed for development. It is interesting to note that the second part of the study, i.e. the two research questions RQ2 and RQ4, show a deterioration in the performance of the models, following the addition of secondary features (such as Priority and Project_ID). Both in the estimation of the Story Points and the estimation of the effort in minutes, it can be seen that the increase in information passed to the Bert model and then to the neural network of the BeForE model does not provide advantages in the estimation, probably due to the low statistical correlation, measured with the Pearson coefficient, between the respective added features and the target variables. Undoubtedly, further replications are needed first, to generalize the results, using new data and other industrial projects. Furthermore, further research is required to discover new features representative of the complexity of a User Story, as it is believed, based on the study carried out, that specific information on complexity, several software components involved and team metrics such as productivity and experience could be of immediate help to make estimates more accurate and close to reality.

## 7.2 Future Work

In this section, we describe our future research agenda.

**Focus group.** A first step to improve our vision could be to create a focus group with Project Managers/Scrum Masters and developers of various experiences, to extrapolate information on their needs and their judgment towards the models made.

**New Data.** Another important future work concerns the search for new data more suitable for our task. As seen in the Data Analysis phase, the TAWOS database cannot be considered as a dataset purely for the Agile development of User Stories, due to the Issues that populate it and many missing data. It is therefore necessary to find new specific industrial data.

**New Cost Drivers.** Further work is certainly the search for new cost drivers, to be added to the textual information of a User Story. Provided that the previous proposed work is successful, the number of software components involved per Issue could be used as an indicator of complexity. Within the TAWOS database, the number of software components per User Story can be extrapolated but, as a result of the Data Analysis phase, it is noted that around 90% of the User Stories refer to a single software component, making this information significant.

**GPT2SP and DeepSE comparison.** A further step to improve our study certainly consists in the comparison of the BeForE model with the DeepSE models and particularly GPT2SP. This comparison is important to understand if our model, given the same data, can generate better estimates than the main Deep Learning models proposed in the literature, to also analyze the difference between the two language models GPT2SP and Bert.

# ACKNOWLEDGEMENTS

*English*

This work marks the end of these 5 long years of university, which have been full of emotions. I thank Prof. Filomena Ferrucci, my supervisor during this thesis work. During the master's degree, she has always proved to be a competent and prepared figure, as well as willing to help. Thanks to Fabio Palomba, one of my co-supervisors during this work. In him, I found a reliable, cordial and competent person capable of supporting the collaborators around him, as every good manager should be able to do.

Even though much of this master's degree was lived remotely due to the pandemic, there was no shortage of happy moments during the various exams. In particular, I warmly thank Vincenzo, Gaetano and Davide for the wonderful experiences they had together and the endless calls on Teams until late in the evening. Special thanks to Mattia for being willing to help me during my thesis work and for the wonderful experience of Semantic Web. I thank Davide, with whom I shared my latest adventure on this path, the Data Science exam. It was a mystical experience, a mix of nervous exhaustion from too much exercise, joy and fear.

Even if these two years have been quite challenging, the people close to me have always been ready to support me in the various moments.

I thank my family and my non-academic friends, who have supported me and who I hope will continue to do so. I will do everything in my power to deserve the help that is offered to me in the days ahead.

I thank my girlfriend Roberta, without her to give me the energy and energy to face every exam and choice, this journey would certainly have been much more difficult. We practically

played a game in two! You have never discouraged me, on the contrary, you have always tried to motivate me to give my best. I think that no words can ever express my appreciation, and for this reason, I will try to demonstrate it by committing myself both in work and in everyday life.

I conclude with a warm thank you to all my loved ones, promising them and me to commit myself to making this university path bear fruit.

## *Italiano*

Questo lavoro segna la fine di questi 5 lunghi anni di università, che sono stati ricchi di emozioni. Ringrazio la Prof.ssa Filomena Ferrucci, mia relatrice durante questo lavoro di tesi. Durante il master si è sempre dimostrata una figura competente e preparata, oltre che disponibile ad aiutare. Grazie a Fabio Palomba, uno dei miei correlatori durante questo lavoro. In lui ho trovato una persona affidabile, cordiale e competente capace di supportare i collaboratori che lo circondano, come dovrebbe saper fare ogni buon manager.

Anche se gran parte di questa magistrale è stata vissuta a distanza a causa della pandemia, non sono mancati i momenti felici durante i vari esami. In particolare, ringrazio calorosamente Vincenzo, Gaetano e Davide per le belle esperienze fatte insieme e le interminabili chiamate su Teams fino a tarda sera. Un ringraziamento particolare a Mattia per essersi mostrato disponibile nell'aiutarmi durante il lavoro di tesi e per la bella esperienza di Web Semantico. Ringrazio Davide, con il quale ho condiviso la mia ultima avventura di questo percorso, l'esame di Data Science. E' stata un'esperienza mistica, un mix di esaurimento nervoso per i troppi esercizi, gioia e paura.

Anche se questi due anni sono stati piuttosto impegnativi, le persone a me vicine sono sempre state pronte a supportarmi nei vari momenti. Ringrazio la mia famiglia e i miei amici extra-accademici, che mi hanno sostenuto e che spero continueranno a farlo. Farò tutto ciò che è in mio potere per meritare l'aiuto che mi viene offerto nei giorni a venire.

Ringrazio la mia ragazza Roberta, senza di lei a darmi l'energia e la carica per affrontare ogni esame e scelta, questo cammino sarebbe stato sicuramente molto più difficile. Praticamente abbiamo giocato una partita in due! Non mi hai mai scoraggiato, anzi, hai sempre cercato di motivarmi nel dare il massimo. Penso che nessuna parola possa mai esprimere la mia graditudine, per tal motivo, cercherò di dimostrarla impegnandomi sia nel lavoro che nel quotidiano.

Concludo con un caloroso ringraziamento a tutti i miei cari, promettendo loro e a me di impegnarmi così da far fruttare questo percorso universitario.

# REFERENCES


[1] Adam Trendowicz and Ross Münch Jürgenand Jeffery. "State of the Practice in Software Effort Estimation: A Survey and Literature Review". In: *Software Engineering Techniques*. Ed. by Zbigniew Huzar, Radek Koci, Bartosz Meyer Bertrandand Walter, and Jaroslav Zendulka. Springer Berlin Heidelberg, 2011.

[2] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. "Selecting Best Practices for Effort Estimation". In: *IEEE Transactions on Software Engineering* (2006).

[3] Barry Boehm, Ricardo Valerdi, J Lane, and AW Brown. "COCOMO suite methodology and evolution". In: *CrossTalk* (2005).

[4] David Cohen, Mikael Lindvall, and Patricia Costa. "An introduction to agile methods." In: *Adv. Comput.* 62.03 (2004), pp. 1–66.

[5] Martin Fowler, Jim Highsmith, et al. "The agile manifesto". In: *Software development* 9.8 (2001), pp. 28–35.

[6] Viljan Mahnič and Tomaž Hovelja. "On using planning poker for estimating user stories". In: *Journal of Systems and Software* 85.9 (2012), pp. 2086–2095.

[7] Sliger M. "Agile estimation techniques. Paper presented at PMI® Global Congress 2012–North America," in: (2001).

[8] Arthur L Samuel. "Machine learning". In: *The Technology Review* (1959).

[9] John McCarthy and Edward A Feigenbaum. "In memoriam: Arthur samuel: Pioneer in machine learning". In: *AI Magazine* 11.3 (1990), pp. 10–10.

[10] K. Srinivasan and D. Fisher. "Machine learning approaches to estimating software development effort". In: *IEEE Transactions on Software Engineering* 21.2 (1995). DOI: 10.1109/32.345828.

[11] Mohit Arora, Abhishek Sharma, Sapna Katoch, Mehul Malviya, and Shivali Chopra. "A State of the Art Regressor Model's comparison for Effort Estimation of Agile software". In: *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. IEEE. 2021, pp. 211–215.

[12] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. "A Deep Learning Model for Estimating Story Points". In: *IEEE Transactions on Software Engineering* 45.7 (2019), pp. 637–656. DOI: 10.1109/TSE.2018.2792473.

[13] Michael Fu and Chakkrit Tantithamthavorn. "GPT2SP: A Transformer-Based Agile Story Point Estimation Approach". In: *IEEE Transactions on Software Engineering* (2022). DOI: 10.1109/TSE.2022.3158252.

[14] Vali Tawosi, Rebecca Moussa, and Federica Sarro. "Deep Learning for Agile Effort Estimation Have We Solved the Problem Yet?" In: *arXiv preprint arXiv:2201.05401* (2022).

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. "Improving language understanding by generative pre-training". In: (2018).

[16] Vali Tawosi, Rebecca Moussa, and Federica Sarro. "Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Second Replication Study (GPT2SP Replication Report)". In: *arXiv preprint arXiv:2209.00437* (2022).

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[19] Vali Tawosi, Rebecca Moussa, and Federica Sarro. "On the Relationship Between Story Points and Development Effort in Agile Open-Source Software". In: *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2022, pp. 183–194.

[20] Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, and Federica Sarro. "A Versatile Dataset of Agile Open Source Software Projects". In: *arXiv preprint arXiv:2202.00979* (2022).

[21] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. "Estimating story points from issue reports". In: *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. 2016, pp. 1–10.