Exploring the Integration of Machine Learning Models in Programming Languages on GitHub: Impact on Compatibility to Address Them

faten slama (faten.slama@ensi-uma.tn)  
Manouba University

Imen Ismail  
Manouba University

Iassaad Iatrach  
Manouba University

Research Article

Keywords: GitHub projects (GP), file extension (FE), Programming language (PL), machine learning (ML), Compatibility Issues (CI).

Posted Date: February 20th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2591510/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.
Exploring the Integration of Machine Learning Models in Programming Languages on GitHub: Impact on Compatibility to Address Them

Faten slama*, Imen Ismail† and lassaad latrach††

*Corresponding author(s). E-mail(s): faten.slama@ensi-uma.tn; Contributing authors: imen4smail@yahoo.fr; lassaad.latrach@ensi-uma.tn;
††These authors contributed equally to this work.

Abstract
The complex field of accounting and how information technologies have impacted the way it’s managed. It explains how many accounting tasks, such as data entry, billing, and expense management, can be automated using programming languages, and how visualization tools can help present the data clearly. The use of GitHub projects for accounting purposes can lead to improved efficiency and better financial decision-making. However, compatibility issues may arise on GitHub due to differences in programming languages, frameworks, or operating systems. These issues can be resolved by updating code or making changes, and by consulting the issues page, other repositories, or the language/library documentation. Other programming difficulties, such as performance problems or conflicts with other libraries, can also occur. A file extension approach (EF) for defining GitHub project domains in programming languages was presented, which involved sorting file extensions, computing percentage of each extension, and determining the dominant language. The study also investigated the connection between this EF approach and compatibility issues. The technique was used to recover over 200 deposits from GitHub, though the dataset content needs to be organized first. In this paper, we extend our previous work and describe in detail new algorithms and strategies to improve the efficiency and effectiveness
of compatibility testing. To validate our approach, we have performed experiments and simulations on two widely used middle libraries from the field of high-performance computing. For this paper, we restrict the compatibility testing to the build process (i.e. compilation and deployment) of a component with other components on which it depends, but we will extend the work to functional and performance testing in future work.

**Keywords:** GitHub projects (GP), file extension (FE), Programming language (PL), machine learning (ML), Compatibility Issues (CI).

## 1 Introduction

Accounting is a complex subject that involves managing the finances of a business or organization. Information technologies have greatly transformed the way businesses manage their accounting by allowing for automation of many tasks. GitHub projects related to accounting often utilize programming languages to automate accounting tasks such as data entry, billing, and expense management. These projects may also use visualization tools to present data in a clear and concise manner. GitHub projects related to accounting can help businesses improve their efficiency and make more informed financial decisions.

In addition, Compatibility issues on GitHub can occur when code written in one programming language or framework is incompatible with another language or framework. These issues can arise when different versions of a programming language or library are used, or when code written for one operating system is run on a different operating system. To resolve compatibility issues, developers may need to update their code to use compatible versions of programming languages or libraries, or they may need to make changes to the code to ensure that it works correctly on different operating systems. Additionally, checking the issues page of the repository in question, searching for similar issues in other repositories, and consulting the documentation for the language or library in question can also help to resolve compatibility issues.

There are numerous types of programming language difficulties on GitHub. Others may be due to conflicts with other libraries or frameworks, while some may be the result of grammatical errors or programming issues. Users might also experience performance problems or Compatibility issues occur when a project’s code is incompatible with specific platforms or operating systems or problems with the version of their programming language. Finding suitable solutions to these issues is crucial. Some options include looking for answers on developer forums or enlisting the assistance of more seasoned developers.

In this work, a file extension approach (EF) for defining GitHub project domains in programming language was presented. The file extensions of every project folder will be sorted, and percentages for each extension will be computed. The project in question will then be removed with the PL flag set to null, and its dominant language will be determined by the language with the
highest proportion (does not contain any part of the software). This study also led us to search for a connection between the FE approach and Compatibility Issues (CI), which arise when a project’s code is incompatible with particular platforms or operating systems. Using this technique, we recovered over 200 deposits from GitHub. Before starting, the dataset content needs to be organised.

Important companies like Azure, Google, Aws, and Microsoft use GitHub as the finest storage solution for open source projects. Everyday, a large number of projects are registered in GitHub with the ultimate goal of making them feasible for further usage and easing project cloning. The complexity of the programming languages employed in this open source platform can be seen and explored on GitHub. Several million programming languages, including Python, Java, C++, etc., are included in the latter. In essence, the original concept of GitHub is to classify existing projects according to their programming languages (PL). For DevOps engineers, this approach can make it simple to recycle source code, clone projects, detect, and install apps. On the software development site GitHub, users can use Git to store, share, and collaborate on software projects. The source code of a project can be changed, and users of the Git version control system can track those changes.

GitHub supports a wide range of programming languages, including C, C++, Python, Java, JavaScript, PHP, Ruby, C sharp, Swift, Go, Shell, TypeScript, and many more. Users can establish repositories to store their code in addition to using collaborative tools to work on team projects. Developers can use GitHub to host open-source projects, as well as to look for worthwhile projects to join. A web-based platform for hosting and working on software projects is called GitHub. Users can collaborate on projects, log changes to their code, and share their work with others. The ability to manage versions, support for teamwork and communication, and a massive user base are just a few of GitHub’s important features.

GitHub enables businesses to utilise them all—in a one location. Researchers now have the chance to explore a wide range of software engineering issues on a big dataset thanks to the accessible tools and datasets existing on the GitHub ecosystem. Although GitHub indicates the language of each project, after compiling the results, we see that it is standard practise to employ many programming languages in closely related open source projects (OSP). I began to wonder how GitHub determines the language to use for each project as a result of this.

Depending on the kind of project and the interests of the engineers working on it, many programming languages may be employed on GitHub. On GitHub, languages including JavaScript, Python, Java, and C++ are frequently used. To identify projects created in a particular programming language, users can search for repositories on GitHub by language. GitHub offers users tools to manage their projects and work with others in addition to hosting code. This covers tools for team collaboration, project management, and issue tracking.
GitHub ranking is generated by Linguist, a library used on the site to detect languages used in public and private repositories, excluding forks. However, it should be noted that the ranking is relative in that if a language does not exist on GitHub, it will never be part of the ranking, regardless of its popularity outside the collaboration site. At a high level, GitHub is a website and cloud service that helps developers store and manage their code, as well as track and monitor changes to it. To understand exactly what GitHub is, you need to know two related principles:

### Version control

Version control helps developers track and manage changes to the code in a software project. As a software project grows, version control becomes essential. Take WordPress...

At this point, WordPress is a fairly large project. If a core developer wanted to work on a specific part of the WordPress code base, it would not be safe or efficient to have them modify the “official” source code directly. Instead, version control allows developers to work safely through branches and merges. With branching, a developer duplicates a portion of the source code (called the repository). The developer can then safely make changes to that part of the code without affecting the rest of the project. Then, once the developer has successfully made his or her portion of the code work, he or she can merge that code with the main source code to make it official. All these changes are then tracked and can be undone if necessary. Git is a specific open-source version control system created by Linus Torvalds in 2005. Specifically, Git is a distributed version control system, which means that the entire code base and history is available on each developer’s computer, allowing for easy branching and merging.

GitHub’s collaborative design also includes social networking features such as wikis, feeds, and followers. Companies and individuals host their codes on the platform for open source and closed projects. In addition to being cloud storage for code, users can also browse and download public repositories and make contributions such as feature requests and code reviews. The service also offers a self-managed version of its services and features for businesses and organizations called GitHub Enterprise. You can install it on company hardware or on a cloud service. As expected, GitHub integrates well with third-party applications and custom tools to create your projects and automate your workflow. It also has an integration of workflow tools. GitHub applications, which have granular APIs and built-in webhooks, give developers more control over what they build. They are accessible from a dedicated market, where you can search for everything your project needs in multiple categories. Even more interesting, the codes of all these tools that make it possible to quickly create an application are on GitHub, available to all. But developers find multiple defucult to install a github project application for multiple reasons (Show fig.2). The project title does not always provide the program type index to be used. In the single project, you can find multiple folders where files just for
In 2022, the number of first-time contributors to open source projects is on the rise. Thus, the developer takes the time to select the code files to use in each project. For companies and our case study. There are thousands of clone projects in GitHub so it is necessary to develop an intelligent method to classify projects by this programming language. Clustering all project have the similary programming language.

This image shows a set of files with different extensions (".java", ".py", ".mp3", etc.). A classification algorithm is used to group the files into different categories based on their extension. For example, the ".java" files are grouped into the "Java" category, the ".py" files are grouped into the "Python" category, and the ".mp3" files are grouped into the "audio" category.

The algorithm can be trained on a dataset of labeled files to learn to predict the category of a file based on its extension. Once trained, it can be
used to classify new files based on their extension. This image is a simplified representation of classifying files by their extension, and the details of the implementation may vary depending on the data and the algorithm used. If you are interested in implementing this approach, I would recommend consulting a machine learning expert or doing further research on the topic.

This new framework has been successfully applied to a real database containing more than a hundred projects with various extensions. The results obtained after that, we use FE of each PL, then extension-based segmentation to find the most used language. Finally, we use techniques to analyze GitHub projects. In this article, we will further detail this new method and we will describe and explain the obtained results. When you have a set of unlabeled data, it’s very likely that you’ll be using some kind of unsupervised learning algorithm. So we choose K-means clustering because it is the most commonly used clustering algorithm. It’s a centroid-based algorithm and the simplest unsupervised learning algorithm. This algorithm tries to minimize the variance of data points within a cluster. It’s also how most people are introduced to unsupervised machine learning. K-means is best used on smaller data sets because it iterates over all of the data points. That means it’ll take more time to classify data points if there are a large amount of them in the data set. Since this is how we implement the method EF to k-means clusters and facilitate the use of the application GitHub.

There are several ways to categorize GitHub projects by programming language. One of the most popular methods is to use GitHub’s search functionality to look for projects using keywords associated with various programming languages. For instance, you can search for Python-based projects by using keywords like “Python,” “Django,” or “Flask.” Additionally, there are tools like GitHub Trends that allow you to view the trends of the most well-liked projects according to programming languages. There are also third-party services like GitHub Language Statistics, which allows you to view the language statistics for all of GitHub’s open-source projects. There are visualisation tools available that will allow you to see the trends of programming languages used on GitHub. Additionally, data analysis tools exist that can be used to extract information about GitHub projects based on the programming language they use. Github project can be efficiently imitated through the site’s fork process or through a Git clone-push sequence and improve the quality of GitHub project samples that are utilised to conduct empirical software engineering studies. This work surveys the recent attempts, both from the machine learning and operations research communities, at leveraging machine learning to solve combinatorial optimization problems. Given the hard nature of these problems, state-of-the-art algorithms rely on handcrafted heuristics for making decisions which are otherwise too expensive to compute or mathematically not well-defined. In recent years, the development of automated tools that classify or extract information in GitHub has been still in the passage of development, primarily engrossed on the submitting,
review and evaluation process. To recommend experts for the development of AI and machine learning to classify the content of GitHub. Although different classification in GitHub, for example in [4] M. Golzadeh propose an automated classification model to detect bots, Bots are frequently utilized in Github projects to automate iterative activities that are part of the relegated software development process. As a case for study [14] J. Manuel Pérez-Verdejo propose TF-IDF algorithm to identify quality-attributes-related knowledge on such user feedback. There are numerous different issues that could arise when managing a programming project on GitHub. Some such examples include:

Conflicts during branch fusion occur when many people make changes to the same file at the same time, and the changes become incompatible after the branches are combined. Compatibility issues occur when a project’s code is incompatible with specific platforms or operating systems. Performance issues occur when a project’s code takes too long to run or consumes too many system resources. Security issues arise when a project’s source code contains flaws that malicious individuals could use against it. Problems with compatibility can occur when a project’s code has been developed and tested on a certain platform or operating system but does not work correctly on other platforms or operating systems. These issues could be brought about by variations in the system libraries, software versions, or hardware settings. There are several methods for fixing compatibility issues: testing the code across a wide range of platforms and operating systems to find compatibility issues. Making the code compatible with the targeted platforms or operating systems by using compatibility libraries or tools. Utilize platform detection techniques to modify the code according to the platform or operating system that it runs on. Utilize container and virtualization technologies to isolate the code and its dependencies from platform differences. It’s crucial to remember that compatibility issues can sometimes be resolved by working with the open-source community, submitting pull requests with fixes, and talking to contributors.

In the context of GitHub, compatibility refers to the ability of a project to work seamlessly with other projects, platforms, and technologies. A compatible project is one that can be integrated with other projects, run on different platforms, and be used in conjunction with other tools and technologies without any issues.

For example, a project may be compatible with different operating systems, such as Windows, macOS, and Linux, or with different web browsers, such as Chrome, Firefox, and Safari. A project may also be compatible with different programming languages, databases, and other technologies.

Having a compatible project is important for several reasons. It enables users to integrate the project into their existing workflow, use it with other tools and technologies they already have, and collaborate with others who may be using different platforms and technologies.

To ensure compatibility, projects may need to follow certain standards, use certain APIs, and adhere to certain protocols. Additionally, they may need
to be tested and validated on different platforms and configurations to ensure that they work as expected.

In the context of open-source projects hosted on GitHub, compatibility is an important consideration for both contributors and users of the project. By working together to ensure compatibility, contributors and users can help to ensure that the project remains useful and relevant, and that it continues to grow and evolve over time.

2 RELATED WORK

2.1 Compatibility Problems

Numerous issues have been linked to Android fragmentation [5, 9]. While Pathak et al. have shown that the frequent Android OS upgrades have been the cause of a significant portion of user complaints regarding energy issues [7], Liu et al. have discovered that Android performance bugs could only be spotted while testing some specific devices and Android platforms [11]. Nayebi et al. discovered in a usability study from 2012 that the various display resolutions of devices present design and implementation challenges for Android apps, creating significant compatibility concerns [13]. Our study, which focuses on API-related compatibility issues, is complementary to several related initiatives. FicFinder [21] is the work that comes closest to ours in terms of fixing API-induced compatibility problems. Recently, Wei et al. suggested this tool for Android developers. However, their work differs from ours in a number of ways: 1) Due to human construction, the database (i.e., API-context pairs) used by FicFinder to highlight compatibility issues is tiny and is likely to produce a significant number of false negatives (i.e., missing real compatibility issues). Our approach to CiD is based on an API lifetime model that is constructed automatically and methodically by mining changes between various versions to the Android framework. 2) API compatibility issues, including both forward and backward compatibility, are the main focus of our effort. FicFinder models other device-specific concerns by mining issue reports.

2.2 GitHub Repositories

we classify related work into table. we introduce the most related works to ours. the we Following the illustrative scheme of the generalization process of other work, Fig. 1 shows that in GitHub [6] For manual tests, it is a person, an experienced tester, who will navigate the GitHub project. We use the file extensions of each programming language. Unlike the automated test, the manual test allows you to test or find the GitHub project language of the file extension that contains the application code, in which case, we can say that the project is the developer of this language. EF are considered to categories the projects in GitHub, and used by software developers and programmers as a source code or script file types. A source code file is a human readable text
file that contains a collection of statements or declarations in any of many computer programming languages (e.g. BASIC, PASCAL, DELPHI, C, C++, C#r, COBOL, etc.).

On the other hand, GitHub notes that in its open source community, projects built from code and toolchain shared by others are growing. Thus, the performance of developers at work would be likely to increase by 87% when code reuse is easy and does not introduce friction.

Show Table 1 all of research used Linguist to understand the types of programming language. Linguist takes the list of languages it knows from languages.yml and uses a number of methods to try and determine the language used by each file, and the overall repository breakdown. Linguist starts by going through all the files in a repository and excludes all files that it determines to be binary data, vendored code, generated code, documentation, or are defined as data (e.g. SQL) or prose (e.g. Markdown) languages, whilst taking into account any overrides. If an explicit language override has been used, that language is used for the matching files. The language of each remaining file is then determined using the following strategies, in order, with each step either identifying the precise language or reducing the number of likely languages passed down to the next strategy: Vim or Emacs modeline, commonly used filename, shell shebang, file extension, XML header, man page section, heuristics, naïve Bayesian classification.

The result of this analysis of Linguist method is used to produce the language stats bar which displays the languages percentages for the files in the repository. The percentages are calculated based on the bytes of code for each language as reported by the List Languages API.

“Linguistics is the scientific study of language. It involves the analysis of language form, language meaning, and language in context. Linguists traditionally analyse human language by observing an interplay between sound and meaning.”

Linguist is a Ruby library so you will need a recent version of Ruby installed. There are known problems with the macOS/Xcode supplied version of Ruby that causes problems installing some of the dependencies. Accordingly, we highly recommend you install a version of Ruby using Homebrew, rbenv, rvm, ruby-build, asdf or other packaging system, before attempting to install Linguist and the dependencies.

In addition, Yu Zhang [24] propose the HIGITCLASS framework to classify of GitHub Repositories. Similarly, Song Yu[23] propose a supervised classification model attached to supervised topics model and Naive Bayes classifier to make the Pull Requests (PRs) appear in Github to be classified automatically. A. Enayet in [3] shows a classification system for GitHub issue comments using both sentence-level and word-level embedding models to leverage information from the SwDA dataset.

FEs User an extension language [2, 16] is a programming language interpreter offered by an application program, so that users can write macros or
even full-fledged programs to extend the original application. Extension languages [8] have a C interface (it is usually C, but it could be any other compiled language), and can be given access to the C data structures. Likewise, there are C routines to access the extension language data structures. File extensions [20] are very important for the proper identification of programs that can open and display the correct information with regards to a file.

Table 1 Acomparative research study used to classify projects GitHub with the programming language.

<table>
<thead>
<tr>
<th>Years</th>
<th>authors</th>
<th>method</th>
<th>subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014 [18]</td>
<td>F. Tomassetti, M. Tor</td>
<td>Polyglot-ism (Linguist)</td>
<td>How many languages are used in each software project, Relations between different languages in similarity of GitHub project.</td>
</tr>
<tr>
<td>2015 [12]</td>
<td>P. Mayer, A. Bauer</td>
<td>process of random</td>
<td>Number and type of languages found and the relative sizes of the languages.</td>
</tr>
<tr>
<td>2019 [19]</td>
<td>PH. Merlin, A. Stefik</td>
<td>polyglot prog (Linguist)</td>
<td>Described a pilot experiment on the impact of code switching on software development productivity. Findings in linguistic research suggest that there is a time cost to switching between natural languages.</td>
</tr>
</tbody>
</table>

In our case,

3 Strategies GitHub projects dataset

In the following sections we describe the data set in which we perform our analysis, as well as the lexical sentiment analysis approach we use.

3.1 Data Collection and Features

We gathered 100 repositories that were as close as possible to being fully written in Java due to the uncertainty of how Designite Java would perform. Another requirement we decided upon was that each repository contains at least 3000 lines of code for it to be considered. As mentioned in further sections, this was due to alleviating potential inconsistencies and ensuring that one particular smell would not be disproportionate to other smells. This particular minimum value was selected due to testing the tool with smaller repositories and discovering this value to be suitable and adequate for our sample size. Most repositories examined were libraries and frameworks repository would also be thoroughly checked to ensure that no other languages would be scanned in by our chosen tool.
We collect one data set of GitHub projects envelop different projects domains language. Their statistics are specified in Table 1.

Table 2: Dataset used in this study is organized as shown below.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Input-projects</th>
<th>output-projects</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>200</td>
<td>176</td>
<td>±languageID</td>
</tr>
<tr>
<td>Data 2</td>
<td>500</td>
<td>400</td>
<td>±languageID</td>
</tr>
<tr>
<td>Data 3</td>
<td>1000</td>
<td>853</td>
<td>±languageID</td>
</tr>
</tbody>
</table>

In this study, we produce a data set using the topics projects GitHub and the others project. Our dataset has 1000 projects extracted from a GitHub. These 1000 projects are collected by selecting the top 1,000 projects sorted by classify the file extension of each project. For own use after the classification FE we give projects a programming language specified (see Fig. 1). Bao et al. [10] dataset has 917 repositories, while our data set has 1000 projects. We found 30 projects with not any extension for the 11 top language choose shown in Figure 3.

The original values of the continuous features have very different scales. While modeling the data, some features tend to dominate others because of their higher range. Hence, to reduce the variance between the features and scale all of them to a certain range, standardization is used. Standardization expects the data to have Gaussian distribution and as observed, the features of the data have close to the Gaussian distribution. Standard scaling centers the data points by subtracting them from the mean and scales by dividing them by the standard deviation. Thus standardization is also referred to as ‘center scaling’. Equations 1, 2, and 3 are the equations for Mean, Standard deviation and Standard Scaled values respectively:

We construct a data containing 1000 projects with +2000 files. Figure 3 shows the organization of the futures, when we categorized as either file extension. For each observation, we extract +50 features related to file project extension from GitHub. Figure 3 shows the association evolution for the programming language across the years. These +50 features are comprised of all of extension features and the 10 top contributor features. We used the GitHub to extract the top active language are the features.
The proposed method aims to collect and interpret the data from GitHub projects related to files or documents originating from different geographical programming language like Java, Python, JavaScript, etc.

### 3.2 Language detection Categorizing extension Languages

Programming languages with file extensions in GitHub projects can be broadly categorized based on the type of language they are. Here is a brief overview of some of the most common programming languages you might encounter in a GitHub project:

**Python**: Python is a high-level, interpreted language that is widely used for web development, scientific computing, data analysis, and more. It is known for its readability, ease of use, and vast library of modules. The file extension for Python is `.py`.

**JavaScript**: JavaScript is a high-level, dynamic, and interpreted programming language that is primarily used for web development. It is used for creating interactive web applications, animations, and other client-side scripting. The file extension for JavaScript is `.js`.

**Ruby**: Ruby is a dynamic, object-oriented, and interpreted programming language that is often used for web development, scripting, and system automation. It is known for its elegance and readability. The file extension for Ruby is `.rb`.

**Java**: Java is a high-level, object-oriented, and class-based programming language that is widely used for developing enterprise applications, mobile applications, and other large-scale projects. The file extension for Java is `.java`.

**C**: C is a low-level, procedural programming language that is widely used for system programming, embedded systems, and other applications that require a high degree of control over the hardware. The file extension for C is `.c`.

![Fig. 3](image-url) Top and Evolution of the association between the main selections of programming languages in GitHub until 2021.
C++: C++ is an extension of the C programming language that adds object-oriented programming features. It is widely used for developing high-performance applications, such as games, scientific simulations, and other resource-intensive programs. The file extensions for C++ are .cpp and .cc.

C# (C sharp): C sharp is a modern, object-oriented, and type-safe programming language that is widely used for developing Windows desktop applications, games, and other applications. The file extension for C# is .cs.

Swift: Swift is a powerful, object-oriented, and type-safe programming language developed by Apple for developing iOS, macOS, and other Apple platforms. The file extension for Swift is .swift.

Go: Go is a statically typed, concurrent, and garbage-collected programming language developed by Google. It is widely used for developing scalable and high-performance systems and applications. The file extension for Go is .go.

TypeScript: TypeScript is a statically typed superset of JavaScript that adds optional type annotations and other features to the language. It is widely used for developing large-scale and complex web applications. The file extension for TypeScript is .ts.

These are just a few examples of the programming languages you might encounter in a GitHub project. There are many other programming languages with different file extensions that may be used in a project, depending on the requirements and constraints of the project.

In GitHub, programming language extensions are used to identify the programming language used in a project and to highlight the specific syntax and constructs used in the code. This information is used by GitHub to highlight the code, display it in a consistent manner, and make it easier to read and understand.

The following (show Fig. 4) is a list of common programming language extensions that can be found in GitHub projects:

- .py for Python
- .js for JavaScript
- .java for Java
- .rb for Ruby
- .c or .cpp for C or C++
- .go for Go
- .rs for Rust
- .hs for Haskell
- .scala for Scala
- .php for PHP

It’s important to note that this is not an exhaustive list, and that there may be other extensions for other programming languages used in GitHub projects. Additionally, some projects may use multiple programming languages, and in such cases, multiple extensions may be used.

GitHub provides various tools to detect the programming language used in a project. You can view the primary language of a project by looking at
the "Languages" section on the right-hand side of a project’s GitHub page. GitHub also provides a heatmap that shows the distribution of code written in different languages within the project’s repository.

To measure the similarity between two programming languages, you can compare their syntax, features, libraries, and paradigms. There are several libraries and tools available to automatically measure the similarity between programming languages, but the results may vary depending on the methodology used.

in our study, the method EF give all the nature of programming language used in each projects with the number of files language.

---

Fig. 4 Progression of Programming Language Extension: Expansion of JSON File Extension in the Programming World(A Graphical Overview, A Visual Analysis).
3.3 Issues with github’s programming language

- What is the impact of programming language choice on development speed and project success in GitHub?

**RQ1** The choice of programming language can impact development speed, as some languages have different levels of expressiveness, libraries and tools, which can make certain tasks faster or more convenient to accomplish in certain languages compared to others.

However, the impact on project success on GitHub is not solely determined by the programming language, as other factors such as team size, development processes, user needs and specifications, budget, and project scope also play important roles. Ultimately, the most suitable language for a project is one that meets the project requirements and allows the development team to work efficiently and effectively.

- How does the use of GitHub affect software development and bug tracking in projects?

**RQ2** GitHub can significantly affect software development and bug tracking by providing:

Version control: GitHub provides a platform for version control using Git, which allows developers to track changes, collaborate on code, and manage merge conflicts efficiently.

Collaboration: GitHub makes it easy for developers to work together on projects by providing a platform for code review, commenting, and bug tracking.

Bug tracking: GitHub offers built-in issue tracking, which allows developers to create, manage, and track bugs, enhancement requests, and other project tasks in one place.

Open-source community: GitHub provides a large open-source community, allowing developers to easily contribute to and use existing code and tools.

By providing these features, GitHub can improve the efficiency, transparency, and collaboration in software development and bug tracking, leading to better and faster development processes.

- How does the community and collaboration around programming languages on GitHub impact their adoption and popularity?

**RQ3** The community and collaboration around programming languages on GitHub can significantly impact their adoption and popularity by:

Providing resources: GitHub provides access to a vast collection of resources, including code samples, libraries, and tools, which can make it easier for developers to learn and use new programming languages.

Encouraging collaboration: GitHub makes it easy for developers to collaborate on projects and contribute to open-source libraries, which can foster the growth of a strong and supportive community around a programming language.

Highlighting popular languages: GitHub’s popularity and widespread use make it a prime platform for measuring the popularity and adoption of different
programming languages, as the number of repositories, contributors, and stars can be used as a proxy for a language’s popularity and user base.

Showcasing real-world use cases: GitHub provides a platform for developers to share their projects and solutions, allowing others to see the practical applications of a programming language and inspiring further adoption.

By providing these benefits, GitHub can play a significant role in shaping the popularity and adoption of programming languages and encouraging their growth.

How does GitHub handle compatibility issues between different programming languages and their libraries and frameworks?

RQ4 GitHub handles compatibility issues between different programming languages, libraries, and frameworks by:

Providing version control: GitHub provides version control through Git, which allows developers to maintain multiple versions of their code and libraries and easily switch between them to resolve compatibility issues.

Documenting dependencies: GitHub allows developers to specify dependencies for their projects, including the version of the programming language, libraries, and frameworks required. This helps ensure compatibility and reduces the risk of errors and bugs.

Encouraging collaboration: GitHub provides a platform for collaboration, allowing developers to discuss and resolve compatibility issues with others and leverage the collective knowledge and expertise of the community.

Supporting package management: GitHub integrates with popular package managers like npm, pip, and gems, which allow developers to easily manage and install dependencies, including libraries and frameworks, reducing compatibility issues.

By providing these features and integrations, GitHub helps developers manage compatibility issues between different programming languages, libraries, and frameworks, reducing the risk of errors and bugs and making it easier to develop and maintain software.

How does GitHub’s version control system support and handle compatibility issues in multi-language projects?

RQ5 GitHub’s version control system, Git, supports and handles compatibility issues in multi-language projects by:

Branching and merging: Git allows developers to work on separate branches and easily merge code changes, ensuring compatibility between different programming languages, libraries, and frameworks.

Tracking changes: Git provides a comprehensive history of all changes made to a project, allowing developers to see exactly what changes were made, when they were made, and who made them. This makes it easier to identify and resolve compatibility issues.

Rollback options: Git provides the ability to revert to previous versions of code, allowing developers to easily undo changes that cause compatibility issues.
Conflict resolution: When multiple developers work on the same codebase, Git provides mechanisms for resolving conflicts that arise, including merging, rebasing, and resolving merge conflicts. By providing these features, GitHub’s version control system, Git, helps developers handle compatibility issues in multi-language projects and maintain a stable and reliable codebase.

How does the integration of machine learning models in programming languages on GitHub impact compatibility and scalability issues, and how are they addressed?

RQ6 The most common compatibility problems faced by developers using multiple programming languages in GitHub projects include:

Library compatibility: Different programming languages often have different libraries and tools, and these libraries may not be compatible with each other, causing compatibility issues.

Syntax differences: Different programming languages have different syntax and structures, making it difficult to integrate code written in different languages.

Data exchange format: Different programming languages may use different data exchange formats, such as XML, JSON, or binary data, leading to compatibility issues. Performance: Machine learning models can be computationally intensive, and integrating them into programming languages may impact the performance and scalability of the code.

Model compatibility: Different machine learning models may use different algorithms, data formats, and libraries, which can cause compatibility issues when integrating them into programming languages.

These compatibility problems are typically addressed by:

Converting data: Developers may use conversion tools or libraries to translate data between different formats and ensure compatibility.

Using middleware: Middleware can be used to provide a common interface between different programming languages and libraries, allowing them to interact and exchange data seamlessly.

Choosing compatible libraries: Developers can choose libraries that are compatible with multiple programming languages, reducing compatibility issues.

Writing glue code: Glue code can be written to integrate code written in different programming languages and ensure compatibility.

Choosing optimized libraries: Developers can choose machine learning libraries that are optimized for performance and compatibility with programming languages, reducing compatibility and scalability issues.

Testing and profiling: Developers can test and profile their code to identify performance bottlenecks and optimize the integration of machine learning models into programming languages.

Using standardized interfaces: Developers can use standardized interfaces, such as APIs, to integrate machine learning models into programming languages, reducing compatibility issues.
Encouraging collaboration: Developers can collaborate on projects and contribute to open-source libraries, encouraging the development of optimized and compatible machine learning models for integration into programming languages.

By using these methods, developers can mitigate compatibility and scalability issues when integrating machine learning models into programming languages on GitHub and ensure the stability and reliability of their code.

3.4 Software compatibility

Software compatibility can refer to the capacity of two systems to work together without making any changes to support each other. Software Compatibility is the interoperability between any two software applications.

A variety of software can be used to assess project compatibility, including:

Microsoft Project is a well-known project planning tool that lets you manage each project’s tasks, resources, and schedule as well as assess how well they work together in terms of both time and resources.

Trello: un outil de gestion de projet en ligne qui peut être utilisé pour évaluer la compatibilité des projets en permettant une vue d’ensemble des tâches et des objectifs, ainsi qu’une collaboration en temps réel entre les membres de l’équipe.

Asana is another online project management tool that enables users to plan, monitor, and organise tasks to achieve shared goals while assessing the compatibility of projects.

Basecamp is a web-based project management tool that may be used to assess the compatibility across projects by centralising interactions, tasks, and documentation for effective goal-tracking.

In terms of project management software, it’s crucial to pick the solution that best suits your requirements in terms of functionality, friendliness, and cost.

4 Methodology

GitHub is a popular platform for hosting and sharing software projects, including those that involve programming. You can find many projects on GitHub that use a specific programming language by searching for that language on the GitHub website. Here are a few examples of how you can search for projects on GitHub by programming language:

- Go to GitHub.com and use the search bar to search for a specific programming language, such as "JavaScript". You can also use the "Language" filter to narrow your search results to only show repositories written in a specific programming language.

- at the GitHub trending page for a specific programming language, which shows the most popular repositories written in that language.

- Look at the GitHub Explore page, which features curated collections of popular and trending repositories by language.
Look at the GitHub leaderboard for a specific language, which shows the most active contributors for that language.

Here, we describe the languages and GitHub projects that we collected, and the analysis methods we used to answer our research question. To classify files with a specific extension, you could use a machine learning algorithm that is trained on a dataset of labeled files with various extensions. This algorithm would take in the file and its extension as input, and would output a predicted class or category for the file.

For example, if you were classifying image files, the algorithm might take in a file with the extension ".jpg" and output the predicted class "image". If you were classifying audio files, the algorithm might take in a file with the extension "mp3" and output the predicted class "audio".

There are many different machine learning algorithms that can be used for classification, and the specific algorithm that would work best for your use case would depend on the details of your problem. Some popular algorithms for classification include support vector machines (SVMs), k-nearest neighbors (k-NN), and decision trees.

It’s also important to note that classification is a complex topic and can be challenging to implement. If you are interested in using machine learning for classification, I would recommend consulting a machine learning expert or doing further research on the topic.

To cluster similar projects on GitHub, you could first collect data on the projects you are interested in. This could include information about the languages used in the projects, the number of contributors, the number of stars and forks, and any other relevant data. In this research, we studied the domain of GitHub Project with a different areas (see Figure 2). The implementation of ML methods shows figure 4. involved training one method using to classify GitHub projects with programming language, for this we need a machine learning algorithm and new method to classify the content extension files in each GitHub project. In this way, When using hierarchical clustering it is necessary to specify both the distance metric and the linkage criteria [7].

During the first phase of the study, We used a new method of quick filing and eliminates traditional tasks (read each shared project in the github) read the contents of the project files file by file in order to know the language of the GitHub project. This method is based on the file extension, which means that this method can read a database containing thousands of projects. Then, indicates the percentage of uses of all programming extensions in each project. In addition, to marking the high percentage as the main language of the project. We will give step by step:

1. Prepare database contains more than 200 projects download from GitHub and prepare a json file containing all extensions of all programming languages.
2. Start the EF-algo method.
3. The percentage of results for each SQ type exists for each project.
   
   see figure 5, This is a result for two projects then master tests it for the whole base. The first project is a mix of programming languages, while the
second is a Python project and the result is a Csv file for impleminter in a cluster algorithm.

We extracted random IDs from 1 to the ID of the last created project. We excluded non-existing, private and fork projects (Set A).

In this project, I implement

4.1 Algorithmic structure

In this section, we examine how learned steps are combined with unsupervised algorithms, considering MLs and explicit algorithms as building blocks, we examine how they can be arranged in different models. The following algorithm instructions are a natural way to examine projects from file extensions. In many cases, using machine learning to approach GitHub projects through the programming language may not be the most appropriate approach. Instead, machine learning can be applied to provide additional input to a classification algorithm, as illustrated in Figure 3. For example, machine learning can provide a parameterization of the algorithm as G is the number of projects to classify. Comme indiqué précédemment, la paramétrisation de l’algorithme FE et SVM fournie par ML doit être comprise dans un sens très large.

We end this section by noting that an ML model used to learn a certain GitHub project can in turn use as features extensions of the top 10 programming languages used in GitHub by a FE algorithm.

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Input: Initial Project set, G=1,2,3,.....,n parametric value,θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Ordred G set based on Ranking class score,r.</td>
<td></td>
</tr>
</tbody>
</table>

\[ setr = .ex, G1, 2, ...., n \]

*compute and normalize test for G*

*Repeat step 4 to 8 until G is null*

*train linear SVM with input as G*

*compute the weight vector using*

\[ \omega = n \sum_{i=1}^{n} \beta_i x_i y_j \]

*calculate the ranking r using*

\[ r_i = \theta \ast \omega_i + (1 - \theta) \ast t_i \]

*choose the G with best ranking score, g*

\[ g = \min \{r_i\} \]

*eliminate the minimum ranked from G set*

*and update the new G*

\[ G = G - g \]

\[ R = r + g \]
Based on the algorithm you provided, it appears to be a method for ranking a set of projects based on a combination of a linear support vector machine (SVM) and a parametric value $\theta$. The algorithm takes as input an initial set of projects ($G$), a parametric value $\theta$, and a set of rankings ($\text{ex}$).

The algorithm first computes and normalizes the t.test for the projects in $G$. It then trains a linear SVM using the input set $G$. The SVM is used to compute a weight vector $\omega$, which is then used to calculate the ranking $r_i$ for each project. The project with the best ranking score is chosen, and the minimum ranked project is eliminated from the set $G$. The updated set $G$ is then used to update the rankings in $R$.

It’s important to note that this algorithm is written in pseudo-code and may not be directly implementable as-is. Additionally, the specific details of how the algorithm would be used and the specific results it would produce would depend on the data and the values of the parameters used. I would recommend consulting a machine learning expert or doing further research if you are interested in implementing this algorithm.

### 4.2 EF method

Fig. 5 Methodology employed to identify the EF algorithm and the ML algorithm to classify project GitHub using the maximum of the extension file used. We added a graphical mark ( ) to identify the challenges related to noisiness, which crosscut the categories of GitHub project.

### 4.3 compatibility project in github relate with programming language

The ”Compatibility” section of a project in GitHub is typically found in the project’s documentation, or in the README file, and is used to provide information about the compatibility of the project with different operating
systems, programming languages, and other technologies. This section can be very important for users who are interested in using the project and want to make sure that it will work with their existing infrastructure.

In this section, developers typically list the specific versions of operating systems, programming languages, and other technologies that their project is compatible with. They might also include information about any known issues or limitations, and any additional steps that users need to take to get the project up and running.

It’s worth noting that the compatibility information listed in the "Compatibility" section of a project is not always 100% accurate or up-to-date, so it’s a good idea to double-check with the project’s documentation and other sources before using the project in a production environment.

The "Compatibility" section of a project in GitHub can be used to provide information about the compatibility of the project with different methodologies. This section can be especially important for users who are interested in using the project for a specific methodology, and want to make sure that it will work with their existing practices and processes.

In this section, developers typically list the specific methodologies that their project is compatible with, such as Agile, Waterfall, DevOps, etc. They might also include information about any known limitations or issues, and any additional steps that users need to take to ensure compatibility with their preferred methodology.

It’s worth noting that compatibility with methodologies can depend on a number of factors, including the size and complexity of the project, the tools and technologies used, and the skills and experience of the development team. As a result, the information listed in the "Compatibility" section of a project is not always comprehensive or up-to-date, so it’s a good idea to consult with the project’s documentation and other sources, and to conduct your own testing, before using the project in a production environment.

---

**Fig. 6** GitHub Projects Compatibility Test Scope and focus.
Check the issues page of the repository: Look for similar issues that have been reported and resolved by other users.

Search for similar issues in other repositories: This can help you understand how other developers have resolved compatibility issues in the past.

Consult the documentation for the language or library in question: The documentation may contain information about known compatibility issues and how to resolve them.

Update your code to use compatible versions of programming languages or libraries: This is often the easiest and most effective way to resolve compatibility issues.

Make changes to the code to ensure that it works correctly on different operating systems: This may involve using conditional statements or other techniques to ensure that the code runs correctly on different platforms.

Use a virtual machine or containerization technology: This can help to isolate the environment in which your code runs, making it less likely that compatibility issues will arise.

Collaborate with other developers: Reach out to other developers who have experience with the language or library in question and ask for their help in resolving compatibility issues.

Test your code thoroughly: Before you release your code, make sure to test it on different operating systems, browsers, and devices to ensure that it is compatible.

In this test, the software or application developed is examined to see if it and the projects are compatible with one another and with earlier iterations of the project. It is possible to manually test the compatibility of projects on Github or to use automated tools, as was already mentioned. This operation’s process consists of four steps. Assure yourself that the testing environments and platforms have already been identified before moving forward with the process.

5 Results

5.1 Result of FE Algorithm

In this case there is similar description for the setup of our methodology, in the first part, we should found the FE Algorithm which allows to read and display all the files in the GitHub project. Then, it allows to classify the files by their extensions and in final state to give the number of files in percentages and to give the name of the most used extension. Figure 15.2 shows the results of a simulation comparing random forests successively at the result in figure 5. we make the result in table 1.

First of all, you can replace the detected language for your repository files using Linguist substitutions. In a few words;

- Each repository is labeled with the first language of the language statistics.
Language statistics count the total file size for each detected programming or markup language. Sold files, documentation and generated files are not counted. The language of each file is detected by the Linguist open source project.

Fig. 7 Overview of file extension classification de deux projets Github différents.
Fig. 8 Correspondence between the EF algorithm and the ML algorithm to classify project GitHub using the maximum of the extension file used: Distribution of projects by Programming Domain.

shows Figure 6 the first test result. Part 1 gives all possible results even as files that do not have programming but in the second stage these files were deleted for easy grouping of projects that have the same programming language, for example we notice project use numbers 7, 11, 2 and 17 are all null. While onna removed the output of our new base. EF classification of our method and we will display the results of my main database which contains
more than 100 projects. Then proceed to the second step and group with function..........

5.2 solution Compatibility issues in GitHub

The programming language’s compatibility relates to how well a project’s code source and the programming language being used are compatible. This means that the code must be consistent with the programming language’s used syntax, libraries, and tools.

For instance, if a project is developed using the programming language Python, it is crucial to ensure that the code is compatible with the various versions of Python and that the necessary libraries are installed on the users’ computers. Additionally, if a project is developed using a certain framework, it is crucial to ensure that the code is compatible with the most recent versions of the framework.

To ensure that the project runs consistently and error-free across various operating systems and platforms, it is crucial to take into account the compatibility related to the programming language. Using the test and validation tools offered by GitHub, contributors can also confirm and validate the project’s compatibility.

A GitHub project’s compatibility issues can be solved using a variety of methods. Here are a few examples:

- Testing the code on a variety of platforms and operating systems will help you find any compatibility issues. This can be accomplished by running tests using test automation tools across many platforms and configurations.
- Use libraries or compatibility tools to make the code compliant with the targeted exploitation platforms or operating systems. For instance, by managing the differences between operating system versions by using compatibility libraries.
- Making use of platform detection techniques to modify the code depending on the platform or operating system.

We have proposed two types of classification techniques here. We applied these three techniques on sets of the identical GitHub projects and different sets of projects. On this basis, shows the Correspondence between the EF classification and the K-Means Clustering algorithm of project GitHub.

- Utilize platform detection techniques to modify the code according to the platform or operating system that it runs on. This could be accomplished by using conditional instructions to check the platform or operating system the programme is running on.
- Utilize container and virtualization technologies to isolate the code and its dependencies from platform differences. This can be accomplished by using tools like VirtualBox or Docker to build reproducible development environments. It’s vital to remember that these treatments can be combined for increased effectiveness. To ensure that other developers can understand and apply the solutions used to solve compatibility issues, documentation of these solutions is also crucial.
DISCUSSION CONCLUSIONS

Github is a software development platform that houses open-source and free software projects. Github projects are compatible with a wide range of programming languages, making it a great place to find projects to use and contribute to.

The programming language used by the project must be understood in order to implement a Github project. It could involve popular programming languages like Python, Java, C++, JavaScript, Ruby, PHP, or other less well-known programming languages.

After determining the programming language used by the Github project, you can download and run the project’s source code on your personal computer. Install the development tools necessary for the programming language, such as the compiler, interpreter, or necessary libraries, in order to accomplish this.

It is often advised to read the README or instructions file for the Github project since it contains crucial information on how to configure and run the project. You can also check the problems and merge requests (pull requests) to see if any known issues or updates are currently being made.

In terms of compatibility, Github is compatible with the majority of programming languages used in the software development industry. This includes both well-known programming languages like Python, Java, C++, JavaScript, Ruby, and PHP as well as less well-known ones like Rust, Swift, Kotlin, etc.

It’s crucial to remember that Github is not just for open-source and free software projects. Additionally, private projects may be hosted on Github using subscription fees. In this case, the compatibility will depend on the programming language used for the private project.

The use and success of Github projects depend critically on the programming language’s compatibility with them. In general, Github projects are compatible with a wide range of programming languages, making it a great place to find projects to use and contribute to.

It’s crucial to keep in mind that each Github project has certain requirements for programming languages and development tools. Therefore, before using or contributing to the project, it is crucial to fully understand the programming language used by it as well as the tools required to execute it.

Developers should be aware of the programming languages that Github is responsible for managing and select projects that match their interests and level of expertise. The popularity of the programming language must also be taken into consideration by developers as it may affect the accessibility of online resources and documentation.

In general, Github is compatible with the majority of programming languages used in the software development industry. This includes both well-known programming languages like Python, Java, C++, JavaScript, Ruby, and PHP as well as less well-known ones like Rust, Swift, Kotlin, etc. Additionally, Github is compatible with a wide range of development tools, including
database management systems, web development frameworks, version control tools, etc.

For developers and software users alike, the Github project’s compatibility with the programming language is ultimately a crucial factor. For software developers and users alike, the Github projects provide a wide range of possibilities and potentially invaluable information and resources.

**Supplementary information.** If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

**Acknowledgments.** Acknowledgments are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

**References**


