Software Technology 2024

Γαβριήλ Κοσκινάς[1], Μιχαήλ Αβαγιανός[2], Θανάσης Ράτσικας[3]

[1]Π2019211, [2]inf2021009, [3]inf2021193

Abstract

Abstracts must be able to stand alone and so cannot contain citations to the paper's references, equations, etc. An abstract must consist of a single paragraph and be concise. Because of online formatting, abstracts must appear as plain as possible.

Introduction

This document introduces the software implementation and architecture, the dataset that was used, the analysis, the results, and the conclusion of the aforementioned analysis. It also explains what a software lifecycle is, and suggests the use of Agile as a methodology for software development. Furthermore, it explains in detail how the software will be released to the public, how each phase will need to be adjusted for such release. Finally, it lists the contributions of each team member.

Implementation

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

UML Diagrams

The UML diagrams that were created for the software show the architecture of the software, both the function and file structure respectively, and the user interaction with the software.

Architecture

The figures below show the file structure and the functions architecture of the software. Pyreverse was used to generate the UML diagrams.

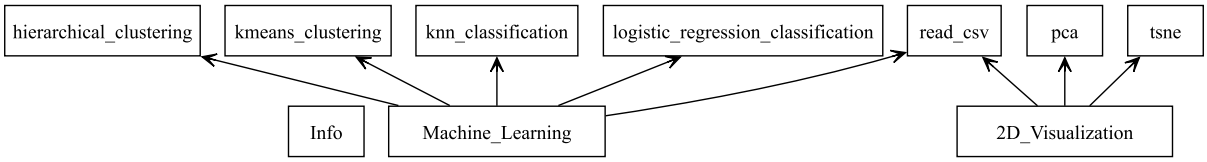

Figure 1: File Structure UML Diagram

The diagram in Figure 1 shows the file structure of the software, where there are three pages: **Info**, **Machine Learning**, and **2D Visualization**. Each page contains the files that are used for the respective tasks.

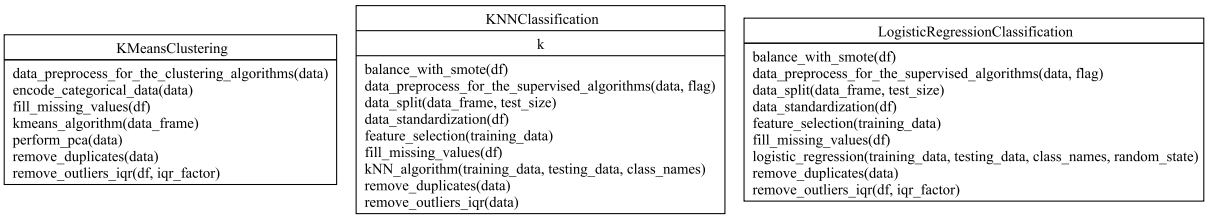

Figure 2: Functions Architecture UML Diagram (1)

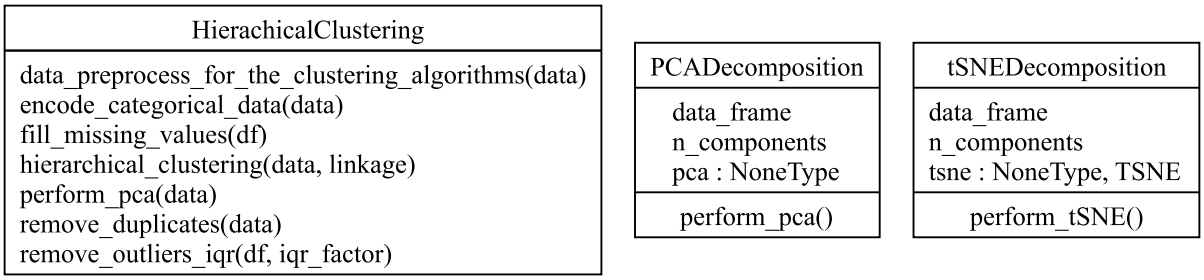

Figure 3: Functions Architecture UML Diagram (2)

The diagrams in Figure 2 and Figure 3 show all 6 classes that were used, their constructors, and the functions that were implemented in each class. The classes are **KMeansClustering**, **KNNClassification**, **LogisticRegressionClassification**, **HierarchicalClustering**, **PCADecomposition**, and **tSNEDecomposition**.

User Interaction

The figure below shows the user interaction with the software. Draw.io was used to create the UML diagram.

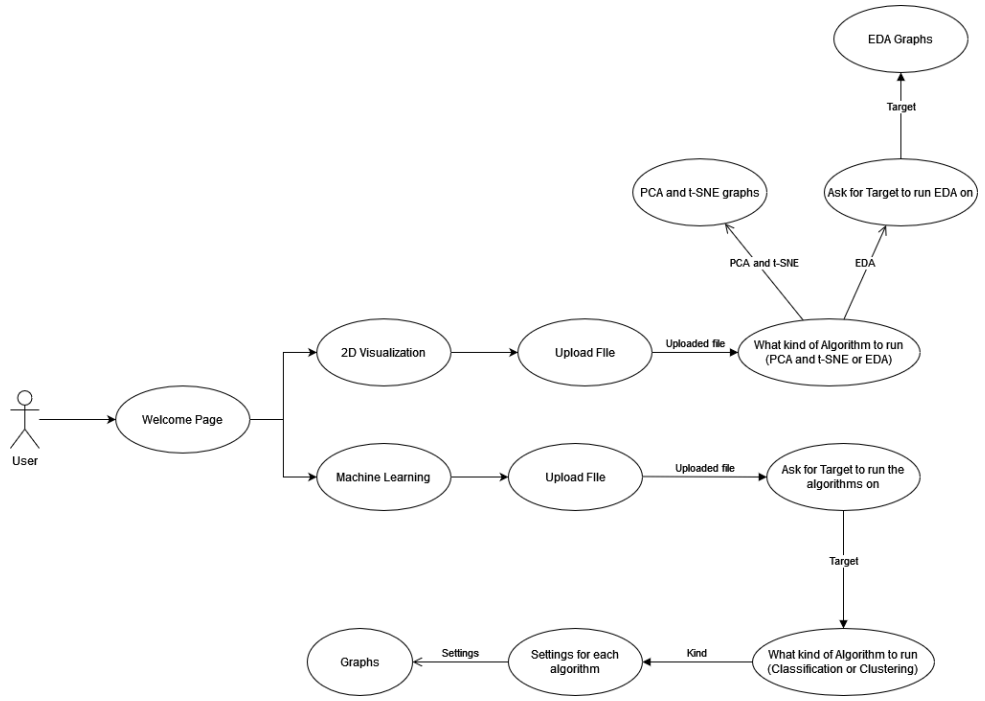

Figure 4: User Interaction UML Diagram

As seen in Figure 4, the user can interact with the software by inputting the data, and then the software will process the data and display the results.

Data

The dataset we used is from the UC Invine Machine Learning Repository, and allows us to be able to Predict Students' Dropout and Academic Success[1].

[1]M.V.Martins, D. Tolledo, J. Machado, L. M.T. Baptista, V.Realinho. (2021) "Early prediction of student's performance in higher education: a case study" Trends and Applications in Information Systems and Technologies, vol.1, in Advances in Intelligent Systems and Computing series. Springer. doi: 10.1007/978-3-030-72657-7_16

Analysis

Results

Conclusion

Software Lifecycle

Software Lifecycle or Software Development Lifecycle (SDLC) is a process used by the software industry to design, develop and test high-quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Agile Methodology

Agile is a type of project management methodology, mainly used for software development, where demands and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change and includes six phases: Planning, Design, Development, Testing, Deployment, and Review.

Releasing to the public

Using the Agile methodology, it requires adjustment to some of the phases mentioned above. The 6-phase cycle will be repeated for each new software iteration.

Planning

During this phase, the team will research the market and the competition to understand the needs of their customers, alongside planning the features and the timeline of the project.

Design

During the Design phase, they will create the app's design, which includes the user interface and user experience, accounting for the information they gathered through research during planning.

Development

In this phase, the team will focus on implementing the features that were planned and designed, while following certain software design principles that allow for easy implementation of new features, patching and bug fixing.

Testing

In the Testing phase, the team tests the features they implemented, and make sure they work as intended, using various testing principles such as unit testing and end-to-end testing. Unit and E2E (end-to-end) testing are automated through use of testing frameworks. Alongside them, they can incrementally test the software with real users, using A/B testing (non opt-in) and beta application clients (opt-in), to gather feedback and improve the software.

Deployment

In the Deployment phase, the team should release the software to the public, allowing for use of testing principles mentioned above, to ensure the software is working as intended.

Review

Finally, during Review, the team will gather feedback from the users, and use it to improve the software, by adding new features, patching bugs, and improving the user experience.

Contributions

- Γαβριήλ Κοσκινάς implemented data fetching, machine learning processing and analysis.
- Μιχαήλ Αβαγιανός created the user interface, the data visualization, the UML and modularized the code.
- Θανάσης Ράτσικας containerized the application, and wrote this document.