CSCM10: Specification

Andy Gray
445348

8/05/20

Swansea University
Prifysgol Abertawe

Abstract

As part of our Masters of Science accreditation, we must complete a research thesis. In has been decided upon to create an educational game centred around Machine Learning (ML), due to the authors desire to gain a deeper understanding of ML and their previous experiences as being a secondary school teacher. We have proposed a game that allows the player to interact with different key ML algorithms and models while providing mediums to help educate and teach the players the understanding of the ML and provide knowledge on how they operate. We will achieve this by creating learning research to accompany the game, as well as links to relevant scientific research to get a deeper understanding. While at the centre of it all, having a fun and engaging game, that uses ML to teach about ML. Through using Python, Pygame and industry-standard accepted libraries and packages, like Tensorflow and Sci-kit Learn. The game will provide key gameplay features that users would expect of games, which will be achieved by using fundamental gamification techniques, to create a fun and engaging game that allows the user to interact directly with the ML models.

1 Introduction

As part of our Masters of Science accreditation, we must complete a research thesis. In has been decided upon to create an educational game centred around Machine Learning (ML), due to the authors desire to gain a deeper understanding of ML and their previous experiences as being a secondary school teacher.

1.1 Overview of the Problem

1.2 Overview of Proposed Solution

1.3 Aims and Objectives

1.4 Structure of the Document

We will first look into background research. Explaining what machine learning with an introduction to what it is, with going into more detail about the different ML techniques. These include supervised vs unsupervised, classification vs segmentation, data-driven approaches, features and dimensionality. An introduction to the machine learning classifiers proceeds the techniques. The classifiers include logistic regression, decision trees, random forests, SVMs and neural networks. We will also look into the literature of gamification and educational games. We will be looking at their impact, and how they get used, what their purpose is how they are adopted and created. We will then explain our proposed methodology, along with our development intentions for the educational game. Including the proposed tools we plan to use and the project management techniques we intend to implement.

2 Background Research

2.1 Intro to Machine Learning

2.1.1 Supervised vs Unsupervised

2.1.2 Classification vs Segmentation

2.1.3 Data-Driven Approaches

2.1.4 Features and Dimensionality

2.2 Introduction to Machine Learning Classifiers

2.2.1 Logistic Regression

2.2.2 Decision Trees

2.2.3 Random Forests

2.2.4 SVMs

2.2.5 Neural Networks

2.3 Introduction to Gamification and Edu-games

2.3.1 Impact

2.3.2 Usage

2.3.3 Purpose

2.3.4 Adoption

2.3.5 Creation

2.4 Edu-games in Computer Science

2.4.1 Basic Computer Science

2.4.2 Machine Learning

2.5 Summary explaining intent to develop the Edu-game for ML

3 Proposed Methodology

3.1 Introduction to Proposed Application

Our main aim is to create an educational game, that aims to teach the main concepts of machine learning. The application will offer multiple game modes like 'claim the screen', 'pin the point on the decision boundary', to name a few. The application will allow the players to interact with the primary ML model to be able to get an understanding of how the application works. Along with giving the user a video explanation and text of how the algorithm works as well as having links to critical scientific papers to gain a better in-depth understanding.

3.2 Overview of proposed Critical Features

3.2.1 Gameplay

3.2.2 Different players? Single vs Multiplayer?

3.2.3 Different Classifiers?

3.2.4 Educational Component

Explanation of scoring? Explanation of model?

3.3 Overview of proposed optional features

3.3.1 Power-ups?

3.3.2 Alternative game modes?

Underlying function for levels? King of the Hill? Domination?

3.3.3 Human vs AI

While the initial implementation is aiming for the main gameplay to be player versus player at first be its priority. However, depending on time requirements, there is a desire to have an option for single-player action and playing against the computer, or AI, although the initial implementation might be very basic in the form of the AI just making 'random guesses'. With potentially being developed into something this more a little more strategic, and is affected by the move the player has just taken.

3.3.4 Unlockables

During the game, when specific actions have happened. The game will reward the player with rewards. The rewards will come in a variety of forms, some in the more traditional gamification methods like badges. However, the main ones will come in the form of code snippets and publication documentation getting unlocked and presented to the player.

Code Snippets
When the plays complete the level, as a reward example code will get presented, allowing the player to see how the algorithm works in practice. Along with additional text to explain step by step what the algorithm is doing.

Publication links
The players will also get presented with links and future read articles to academic science papers and publications for further readings, to get a better fundamental of the algorithm and also where to go to get to the next step.

4 Proposed Mock-up prototype images

5 Proposed Implementation Approach

5.1 Tools

We plan to make the game using Python 3 [**?**]. We plan to use this language due to the vast amount of packages and libraries available. Python has a vast amount of support within the Data Science and Machine Learning community, and it also has packages available to allow the create games and web applications.

Visual Studio Code (VS Code) will be the chief text editor for implementing the game. Due to the nature of text editor, it will allow us to use all the Python libraries as well as any of the iPython notebook files. iPython notebooks are the primary file type

for Jupyter Notebooks whereas VS Code allows user to edit multiple programming languages, which include Python and Jupyter Notebooks, in one place. However, with it being a text editor, it does lack the more advanced features we would expect in an IDE, but these are features we do not need.

Jupyter Notebooks allows the code to be split up into cells, allowing us to be able to test out segments of code, without having to run all of the code in the file, which can be very helpful when trying to develop and implement some modularity features within the application.

We will be using Trello for the kanban tools. "Kanban" is the Japanese word for "visual signal" [**?**]. Using Kanban boards allows us to keep our work visible, this is to allow others to see what it is we are doing, and what is needed to get done. These will allow everyone to see the full picture and keep everyone on the same page.

David Anderson discovered that kanban boards get split into five components: Visual signals, columns, work-in-progress limits, a commitment point, and a delivery point [**?**].

Kanban teams write all their project's work items onto cards, and these are usually one per card. The kanban board gets split into columns, with each column representing an activity which composes the workflow. All the cards change between the workflow until the activity is complete. The column workflow titles can be as simple as to do, in progress and completed. However, David suggests that there should be a work in progress (WIP) limit [**?**]. When a column has reached the limit, of three cards, all team members get expected to focus on the cards in progress. The WIP limits are critical for exposing bottlenecks in the workflow and maximizing flow. WIP limits give an early warning sign that too much work commissioned. Backlogs of ideas are where the ideas of the team and the customers get placed. The moment an idea gets picked up by a team member and work begins, this gets referred to as the commitment point [**?**]. When the product is finished and is ready for deployment, this stage gets referred to as the delivery point. The overall aim of the kanban is to take a card from the commitment point to delivery point as quick as possible.

5.2 Python Libraries

The Python libraries that we will be using are Pygame for game development. [Add a brief description of it here]. This package will be the main one for creating the game's interface and game logic.

For the main implemented machine learning algorithms that are within the game, a package called Scikit-learn will get used. [Add a brief description of it here]. This package will allow us to implement the algorithms [list them here] using the Python language.

5.3 Development

5.3.1 Deployment Format

The main application will be a desktop application. Due to the potential size and scope of the application, the initial aim will be on creating a sleek and smooth learning game and tool, to get the understanding of machine learning across. However, due to the nature of the world at current, with more and more getting done online, a web application version of the application would be feasible. However, this can only be considered when the main application is completed to the expected level, with the desired user experience. Frameworks that would allow us to achieve this would be Python's web library called 'Django'.

5.3.2 Developement style

We will aim to create the app in a modular manner. By creating the application in a modular way, will allow us to create different aspects of the application at the same time. It is allowing us not to be limited to potential bottlenecks in development if particular modules are providing more issues than anticipated. We will then be intending for the Pygame user interface (UI) to be the factor that brings it all together, enabling all the different modules to be triggered based on the requirements of the game during the gameplay.

Module Content

As we will be developing the application modular, we will aim to have a module that will handle the educational content. This module will include the videos and explanations of the algorithms.

Another modular section will for each machine learning model. Each model being implemented and kept within its module section. Keeping them in their module will allow scalability to happen more pragmatically. Making sure that if an ML model is having issues while being implemented within the game, then it can be left out and not impact the overall performance or code base.

Then we aim to have another module that will focus on the interface and UI of the main game and any menu screens. We will aim to have the gamification aspects within an individual module as well.

5.4 Proposed Evaluation Method of the Project

5.4.1 Gameplay Testing

5.4.2 Unit Testing

5.4.3 Systems Testing

5.4.4 User Testing and Feedback Evaluation

6 Proposed Project Management

Project management is crucial for any task that is about to get carried out, even more so the case for software development. As a famous Benjamin Franklin quote says "Failing to plan is planning to fail" [**?**]. With this in mind, we must decide on the right project planning method that compliments our initial software design. From the waterfall method to Rapid application development (RAD) or the more modern methods of agile development, there are many methods that we could choose. We will explain the different methods that we could use and what ones would be best for our solution and intended development method.

6.1 Proposed Life Cycle

The profession of the software developer has existed since the first computers, but the practices and methods for developing software have evolved over timer [**?**]. The approaches have developed over the years to adapt to the ever-changing landscape of software development. The methods, known as software development life cycles (SDLC), vary in approach but fundamentally share the same goal. The main aims of the SDLC are to brake the development up into stage. However, what changes with different SDLC is how these stages get carried out. The different stages are planning, requirements, designing and prototyping, software development, testing, deployment, operations and maintenance [**?**].

The first stage, planning, involves resource allocation, capacity planning, project scheduling, cost estimation and provisioning [**?**]. The primary outcome of this stage is to have an overall plan of what it is we have and what we will need to complete our goal within the constraints like costs and times allowed. The second stage, requirements, is where Subject Matter Experts (SMEs.) guide on what would be needed to carry out the stakeholders' requirements [**?**]. The third stage, design and prototyping, is where the software architects and developers begin to design the software. The outcome of this stage would be documentation on the intended design patterns and design wireframes of the intended final software. The fourth stage, development, is where the software starts to get made based on the decisions made in design and prototyping, following the chosen methodology. The outcome will be testable, tangible software. The fifth stage, testing, is considered the most crucial stage [**?**]. At this stage, it is essential to do all the code quality checking, unit testing, integration testing, performance testing and security testing. The sixth but by no mean final stage is deployment. This stage is when the code is ready to be shipped to the client or uploaded to the required app stores. However, the final stage is operations and maintenance. This stage is about making sure that the software is getting used how

it should and that any bugs that did not initially get picked up in testing are correct and removed from the software.

6.1.1 Waterfall Method

The waterfall method is a model where each section needs to get completed before we can move onto the next stage, like a waterfall flowing down. For example, before we can start analysing the requirements, we need to complete the planning stage. Following the seven critical stages of SDLC, one after each other.

Like all models, they have their advantages and disadvantages. Advantages that this model has is that it is easy to use and follow, and by the way it is all set up, every stage will get finished before the next stage starts. The waterfall method also allows for the project to be easily managed, resulting in easier documentation [**?**]. However, some of the disadvantages are that it is not very useful if the requirements are not very clear at the beginning. Another disadvantage is that once we have moved to the next stage, it is tough to go back to a previous stage to make any changes which therefore creates higher risks to development and has less flexibility [**?**]. The model is best when changes in the project are stable, and the project is small, with the project requirements are clearly defined.

6.1.2 RAD: Rapid Application Development

The overall aim of RAD is to create software projects with higher quality and faster by gathering requirements through workshops or focus groups. Then prototyping the product and then using reiterative user testing of designs early. RAD is the best model to use when we need something created quickly and have a pool of users available to test prototypes. However, this approach can be costly[**?**].

6.1.3 Spiral Method

The Spiral Model – first described by Barry Boehm in 1986 – is a software development methodology that aids in choosing the optimal process model for a given project. It combines aspects of the incremental build model, waterfall model and prototyping model, but is distinguished by a set of six invariant characteristics. The Spiral Model is concerned primarily with risk awareness and management. The risk-driven approach of the spiral model ensures your team is highly flexible in its approach, but also highly aware of and prepared for the challenges they can expect to face down the road. The spiral model shines when stakes are highest and major setbacks are not an option [**?**].

6.1.4 Agile Development

The Agile methodology is a process by which a team can manage a project, which gets achieved by breaking up the project into several stages. It required constant collaboration with stakeholders which leads to continuous iterations of improvement. In essence, Agile development is not a set methodology more of a manifesto that is aiming to uncover better ways to develop software. "Individuals and interactions over processes and tools. Working software over comprehensive documentation. Customer collaboration over contract negotiation. Responding to change over following a plan [**?**]."

6.1.5 Decided Method

The project's requirements have features that lend themselves well to the waterfall methodology. However, we would like to have an element of agile methodology within the development due to the application intending to get created in a modular way. By using the waterfall method it will allow us to have a clear plan and requirements of what is needed, but by using the agile method, we can then rotate between the software development and testing stage.

6.2 Gantt chart

Chart here!!!!

6.3 Risk management

Risk analysis and an overview table Generic and specific risks, including mitigation strategies and impact scores Again, descriptions of the specific risks would be beneficial

7 Summary

This project is probably more Waterfall than any project I've had before. Remember to compare to others and give a justified choice.

References