# Abstract

# 1 Introduction

## 1.1 Project Description

My project aims to automate the testing to help to build new models. The project seeks to use a website platform where you can enter data and make predictions. Using the website, the user can check whether the code is correct. Whenever the user makes new changes to the data, the program will reset the machine learning development cycle by rebuilding the application. Each build will train and test the model and determine whether the tests passed. A continuous integration system is a sandbox environment for testing the model before moving the program onto the deployment stage. There are many areas for the ML continuous integration system: to improve the models in the financial forecast sector.

Context: current project web servers run several ML methods. We need to create a server to automate the running of the ML methods.

The project breaks down into several key concepts:

1. ML Life Cycle: highlights the development cycle for improving an ML model. The developer will use the training and testing ML data files for evaluating the model. Understanding the ML life cycle helps you with enhancing the model validation process.
2. ML Continuous Integration system: provides a platform to take the ML Code and datasets to run automated training. If the tests pass, the platform will execute user-defined downstream processes.

## 1.2. Goals

The project focuses on validating the models with an external source control system (GitHub Actions). After the validation, a user can deploy the model on external sources,e.g., a website, app store or cloud service.

1. Review and configure the ML pipelines for training.
2. Integrate GitHub to monitor the code changes.
3. Implement a tracking method on the datasets.
4. Any changes to the code or dataset will run the ML training pipeline.
5. Optimize for the best ML model by comparing new scores with historical ML models’ scores.
6. Run automated steps (such as sending an email or pushing code to servers) when ML benchmarks pass or fail targets.

# 2. Background

Continuous integration builds the software whenever a developer makes an update [3]. The goal is to run the CI server regularly and to provide a stage for presenting the results. CI server aims to reduce the custom scripts for running different components. A build script is an example of a custom script to help with compiling and building new applications. The developer can apply build script for more generalised scenarios, e.g., outside a continuous integration system. Continuous integration requires testing data as an essential component. In traditional models, the program feedback on the test accuracy scores to the developers to improve the model. Testing scores may cause overfitting because they may mislead the user towards a skewed result. A possible solution is to allow the CI system to select independent testing data for each training process. In my project, the developer will only receive the pass or fail statements as feedback. CI system reruns whenever there's an edit. Tracking the changes in the ML code and datasets can help the customer identify the point to roll back when making a mistake during the development process. A developer should focus on the testing and release stages of the ML development cycle for designing the ML continuous integration system.

## 2.1 Automated Workflow

External platforms help to identify the context for building a CI system and reduce the workload for designing the application [4]. GitHub Action is fantastic for applying to repositories by analysing the users’ repository history:

1. Only a small proportion of the users (0.7% of the 416,266 repositories [4]) used GitHub Actions in their projects. There’s plenty of potential for exploring integrating GitHub Action with a CI system because GitHub Action isn’t a standard tool for automating. The developers can use the GitHub Action history to understand the most typical workflow for planning the projects.

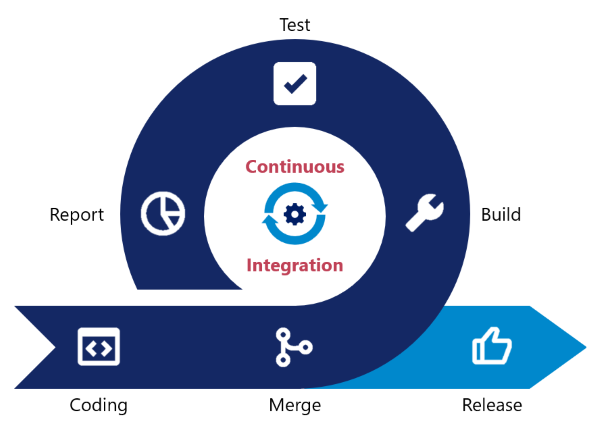
## 2.2 CI Server

A CI server will help with automating the tasks and takes the structure as follows:

* Continuous Change Impact Analysis Process [5] will help with identifying the links between the different components.

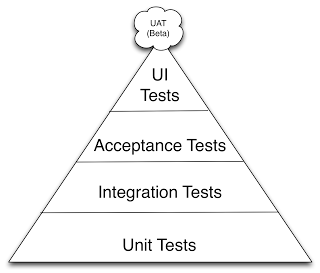
# 3. Notes

## **28-11-2022**



The integration looks at the **CI cycle**:

1. The developer commits code or updates data on the repository. The CI examines the repository for any changes.
2. The CI identifies the changes and rebuilds the program for automated testing. The test will determine whether the changes will be updated.
3. The CI generates a feedback report from the test results and shares it with the project participants.



Using the iterative CI scenario, we can include the testing component in the continuous integration system. You can apply a CI service to configure the testing components into the CI system. An alternative method would be to design your tests for the critical components. Identifying the code coverage helps with determining the untested code sections. For every new function, writing new tests is the best approach to improve test coverage.

# 4. Next steps

Research other papers to increase the variety of references.

1. Academic content for determining whether anyone conducted prior research on the topic.
2. Look for existing software packages on the ML continuous integration tool.

# 5. Sources

1. <https://github.com/iterative/cml>
2. <https://dl.acm.org/doi/pdf/10.1145/3394486.3403290>
3. Continuous Integration (Improving software quality and reducing risk) Chapter 1
4. How Do Software Developers Use GitHub Actions to Automate Their Workflows?
5. Communicating\_continuous\_integration\_servers\_for\_increasing\_effectiveness\_of\_automated\_testing

# 