

Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Automation and Applied Informatics

Achref Mekni

**Graphical user interface to   
manage applying machine learning algorithms in prediction tasks**

SUPERVISOR

Mrad Mohammed Azzouz

" PhD Candidate"

BUDAPEST, 2024

Contents

[ABSTRACT 5](#_Toc167109209)

[ABSZTRAKT 6](#_Toc167109210)

[1 Introduction 7](#_Toc167109211)

[2 Technologies 10](#_Toc167109212)

[2.1 Python Streamlit 10](#_Toc167109213)

[2.2 Keras & tensorflow 11](#_Toc167109214)

[2.3 AWS EC2 12](#_Toc167109215)

[2.4 AWS S3 Bucket 13](#_Toc167109216)

[2.5 AWS Lambda 14](#_Toc167109217)

[2.6 AWS Cognito 15](#_Toc167109218)

[2.7 AWS Cloudwatch 16](#_Toc167109219)

[2.8 Git & Github 17](#_Toc167109220)

[2.9 Visual Studio Code 18](#_Toc167109221)

[3 Literature 19](#_Toc167109222)

[4 Requirements 21](#_Toc167109223)

[4.1 Actors Identification 21](#_Toc167109224)

[4.2 Requirements Specification 21](#_Toc167109225)

[4.2.1 Functional Requirements 21](#_Toc167109226)

[4.2.2 Non-Functional Requirements 22](#_Toc167109227)

[5 Design 23](#_Toc167109228)

[5.1 Use Cases 23](#_Toc167109229)

[5.1.1 Global Use Case 23](#_Toc167109230)

[5.1.2 Manage ANN models 24](#_Toc167109231)

[5.1.3 Manage CNN model 26](#_Toc167109232)

[5.1.4 Visualize Results 29](#_Toc167109233)

[5.1.5 Manage Homepage 29](#_Toc167109234)

[5.1.6 Authentication 30](#_Toc167109235)

[5.2 Machine Learning Part Design 31](#_Toc167109236)

[5.2.1 Viscosity Model: 31](#_Toc167109237)

[5.2.2 Twin-screw Model 33](#_Toc167109238)

[5.2.3 Class Diagram 35](#_Toc167109239)

[5.3 Web Application Part Design 36](#_Toc167109240)

[5.3.1 Application Architecture 36](#_Toc167109241)

[5.3.2 Data Management 37](#_Toc167109242)

[5.3.3 Logging System 39](#_Toc167109243)

[6 Implementation 40](#_Toc167109244)

[6.1 Structure 40](#_Toc167109245)

[6.2 User Interface/Main Components 40](#_Toc167109246)

[6.2.1 SideBar 41](#_Toc167109247)

[6.2.2 Login Page 41](#_Toc167109248)

[6.2.3 Homepage 45](#_Toc167109249)

[6.2.4 Twin Screw Page 50](#_Toc167109250)

[6.2.5 Viscosity Page 60](#_Toc167109251)

[7 Deployment 67](#_Toc167109252)

[7.1 Architecture Overview 67](#_Toc167109253)

[7.2 Infrastructure Setup 68](#_Toc167109254)

[7.3 Deployment Process 70](#_Toc167109255)

[7.4 Benefits of AWS EC2 Deployment 73](#_Toc167109256)

[8 Summary 74](#_Toc167109257)

[References 75](#_Toc167109258)

STUDENT DECLARATION

I, **Achref Mekni**, the undersigned, hereby declare that the present MSc thesis work has been prepared by myself and without any unauthorized help or assistance. Only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word, or after rephrasing but with identical meaning, were unambiguously identified with explicit reference to the sources utilized.

I authorize the Faculty of Electrical Engineering and Informatics of the Budapest University of Technology and Economics to publish the principal data of the thesis work (author's name, title, abstracts in English and in a second language, year of preparation, supervisor's name, etc.) in a searchable, public, electronic and online database and to publish the full text of the thesis work on the internal network of the university (this may include access by authenticated outside users). I declare that the submitted hardcopy of the thesis work and its electronic version are identical.

Full text of thesis works classified upon the decision of the Dean will be published after a period of three years.

Budapest, 24 May 2024

...…………………………………………….

Achref Mekni

ABSTRACT

This thesis presents the development of an AWS cloud-based application designed for creating, training, and testing two types of machine learning models: a Convolutional Neural Network (CNN) viscosity model and three Artificial Neural Network (ANN) twin screw image classifiers. The application aims to streamline the process of model development by leveraging AWS services such as Amazon S3 for data storage, Amazon EC2 for computational power, and AWS Lambda for serverless operations.

The primary objectives were to enable efficient data handling, model training, and validation in a scalable cloud environment. Methodologically, the application supports uploading and downloading datasets to and from an S3 bucket, ensuring secure and scalable storage.

The application successfully facilitates the training of accurate machine learning models, demonstrating significant improvements in processing efficiency and scalability compared to traditional on-premises solutions. This research contributes to the field by providing a robust framework for developing cloud-based machine learning applications, highlighting the potential of AWS infrastructure in advancing data-driven decision-making processes.

ABSZTRAKT

Ez a dolgozat egy AWS felhő alapú alkalmazás fejlesztését mutatja be, amely kétféle gépi tanulási modell létrehozására, betanítására és tesztelésére szolgál: egy konvolúciós neurális hálózat (CNN) viszkozitási modell és három mesterséges neurális hálózat (ANN) ikercsavaros képosztályozója. Az alkalmazás célja a modellfejlesztés folyamatának egyszerűsítése az olyan AWS-szolgáltatások kihasználásával, mint az Amazon S3 az adattároláshoz, az Amazon EC2 a számítási teljesítményhez és az AWS Lambda a szerver nélküli műveletekhez.

Az elsődleges cél az volt, hogy hatékony adatkezelést, modelltanítást és validálást tegyenek lehetővé egy méretezhető felhőkörnyezetben. Módszertanilag az alkalmazás támogatja az adatkészletek feltöltését és letöltését egy S3 tárolóba és onnan, így biztosítva a biztonságos és méretezhető tárolást.

Az alkalmazás sikeresen megkönnyíti a pontos gépi tanulási modellek betanítását, jelentős javulást mutatva a feldolgozás hatékonyságában és méretezhetőségében a hagyományos helyszíni megoldásokhoz képest. Ez a kutatás azzal járul hozzá a területhez, hogy robusztus keretet biztosít a felhő alapú gépi tanulási alkalmazások fejlesztéséhez, kiemelve az AWS infrastruktúrában rejlő lehetőségeket az adatvezérelt döntéshozatali folyamatok fejlesztésében.

# Introduction

Artificial intelligence (AI) has brought about a whole new era of technological innovation, totally transforming industries in all kinds of ways. AI has grown and branched into many areas, from self-driving cars to improving how computers understand language. In this project, we focus on how AI can be developed and used in the pharmaceutical industry, specifically in classifying and predicting images based on the thickness and speed of pharmaceutical droplets.

The pharmaceutical industry plays a vital role in healthcare and is continuously looking for advanced technologies that can enhance drug development, manufacturing, and quality control. The exact characterization and prediction of pharmaceutical droplets could potentially have a huge effect on such processes. That's why this project tries to harness the potential of AI to answer this need. We are developing a solution that merges artificial neural networks and convolutional neural networks to analyze images and predict. The core task is to develop a user-friendly cloud-based graphical user interface enabling seamless interaction of the user with AI models. The GUI will provide an easy way of testing, training, and applying AI models effectively.

In the pharmaceutical world, the measurement of the viscosity of liquid droplets is important during formulation and drug making. Viscosity is an essential property that determines how pharmaceutical formulations work and affects such processes as the release of pharmaceutical formulations, stability, and even the manufacturing process itself. Well, precise measurements of viscosity help assure pharmaceutical researchers and manufacturers that their products are always of the best quality and consistently so. By understanding droplet viscosity, professionals can make informed decisions related to the design of formulas, optimizing processes, and control quality. It is also critical to understanding how droplets will perform and spread in various formulations—critical when delivering drugs. Measuring droplet viscosity accurately is, therefore, of utmost importance to ensure pharmaceutical products are safe, effective, and of a high quality.

Our objective is to develop a CNN-based model that can accurately predict the viscosity values of water-PVP droplets. By training the CNN model on a dataset of droplet images, the model will learn to recognize patterns and features that correspond to different viscosity values. Once the model is trained, it will be able to classify new droplets based on their images and accurately predict their viscosity values. We will begin by analyzing the data, which consists of grouping several photos based on different ratios of water PVP solutions (000, 005, 010, 015). Then, we will move to the model architecture and training which involves selecting the number and types of layers, the number of neurons in each layer, and the activation function to be used. Afterward, we will test our model using a separate testing data set and also check if the model predicts the viscosity of the given test picture correctly. Then, we will show the results such as the Accuracy, MAE, MSE, and RMSE.

Moreover, Wet granulation is a fundamental process in the pharmaceutical industry for the formulation of solid oral dosage forms, such as tablets and capsules. This technique involves the mixing of active pharmaceutical ingredients (APIs) and excipients with a liquid binder to create granules, which are then dried and subsequently compressed into tablets or filled into capsules. Wet granulation offers several advantages, such as improving the flow properties, compressibility, and content uniformity of the blend. It also aids in controlling the release of APIs in the final dosage form.

For this, machine vision can be used as a tool for monitoring continuous twin-screw wet granulation, where the colored tracer would be dissolved in the granulation liquid alongside the API so their concentrations would simultaneously change.

The granule size can be measured using a high-speed imaging camera and image analysis, from which we can deduce the momentary liquid/solid (L/S) ratio

We will start with the data analysis where we have many images splitted into groups based on the feeding speed. Then we will apply an image processing on those pictures to extract several features of the granules which we will store in excel files. Those excel sheets are used later on to train our AI models which were trained to do the classification of the pictures based on the feeding speed (revolutions per minute). Three Artificial neural networks (ANN) algorithms were used which are MLPClassifier, Decision Tree Classifier and Kneighbors Classifier.

Subsequently, the project will explore the machine learning algorithms employed and the environment in which they will be applied. It is essential to elucidate how these algorithms will effectively process and analyze data relevant to pharmaceutical droplets, leading to accurate predictions that can aid decision-making within the industry.

The collected data will undergo necessary preprocessing and visualization steps to ensure its suitability for AI model training and testing.

A core aspect of this project involves testing multiple methods for predictions, allowing for a comparative analysis of the results. This comparative approach will facilitate the selection of the most accurate and effective predictive models, ensuring their practical utility within the pharmaceutical industry.

Furthermore, to provide a user-friendly and accessible interface for industry professionals, a cloud-based user interface (web app) will be developed, with a particular emphasis on cloud solutions such as Amazon Web Services (AWS). This interface will serve as a central hub for seamless interaction with AI models, enabling users to test, train, and apply these models efficiently. The GUI will not only facilitate the process but also serve as a platform to represent and visualize the results.

We will employ Streamlit in Python to develop a user-friendly web application. Streamlit allows you to create an intuitive interface for users to seamlessly interact with your AI models, making it easier for researchers and pharmaceutical professionals to test, train, and apply AI models efficiently.

To efficiently handle the vast amount of image data related to pharmaceutical droplets, we will use AWS S3 buckets to store and retrieve these images. AWS S3's secure and scalable object storage ensures that your image data is readily available for model training and testing. This integration ensures that your project maintains a robust and reliable data storage and retrieval system, crucial for AI model performance.

In addition, I will deploy the Streamlit-based web application on an AWS EC2 instance which can provide the reliability and scalability. By hosting the web app on an EC2 instance, we ensure that the AI models and the user interface are accessible to users 24/7, regardless of the user load.

In summary, the combination of Streamlit for the user interface, AWS S3 for data storage, and AWS EC2 for deployment enhances the effectiveness and accessibility of your AI-based pharmaceutical droplet prediction project, aligning with the project's goals of making AI technology accessible and practical for the pharmaceutical industry.

In conclusion, this project embarks on a journey to harness the power of AI for the pharmaceutical industry, specifically focusing on image classification and prediction related to pharmaceutical droplets. The development of AI models, integration of machine learning algorithms, and the creation of a user-friendly cloud-based GUI form the core components of this research, collectively aiming to provide a valuable tool for enhancing pharmaceutical processes. With the potential to revolutionize drug development and manufacturing, this project stands as a testament to the potential of AI in solving real-world challenges and driving innovation within the pharmaceutical sector.

# Technologies

## Python Streamlit

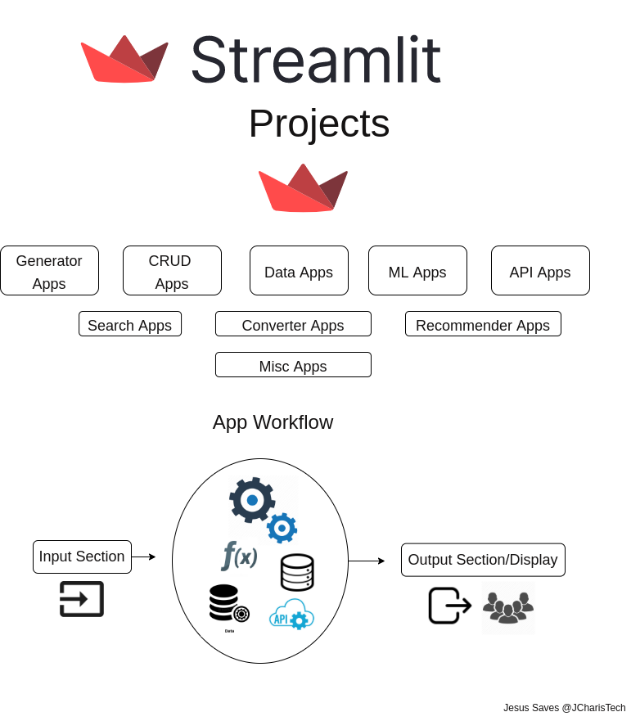


Figure 1: Streamlit

Streamlit is an open-source Python library used for creating web applications for data science and machine learning projects. It's designed to make it simple for data scientists, engineers, and researchers to build interactive and customizable web apps directly from Python scripts.

It has a lot of useful features like interactive dashboards, visualizations, and data applications with relatively little code. It also has built-in widgets and components that make it easy to add features like sliders, buttons, text inputs, and charts to the web app.

The reason why I have chosen streamlit is because it allows developers to create interactive web applications with minimal code. It’s simple and intuitive API allows users to focus on the ML model, visualizations, and data analysis, rather than spending time on complex web development. It also simplifies the deployment of ML models as web applications.

## Keras & tensorflow

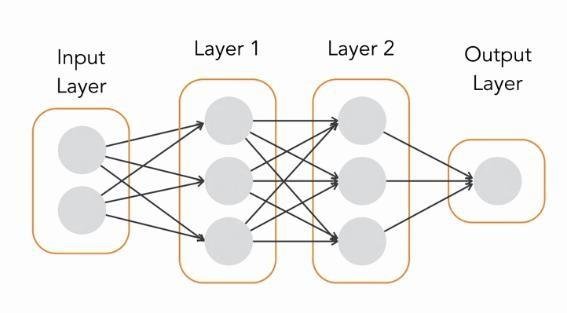


Figure 2: Sequential Keras Model

For building my sequential Machine Learning models i decided to use Keras and TensorFlow which are two powerful and popular libraries used in the field of deep learning and artificial intelligence, particularly in the development and implementation of neural networks.

Since TensorFlow 2.0, Keras has been integrated as the high-level API within TensorFlow, meaning that Keras can now be accessed directly from TensorFlow, providing a simpler and more user-friendly approach to building neural networks while maintaining the power and versatility of TensorFlow's core capabilities.

In Keras, the Sequential model is an easy way to build neural networks by stacking layers one after another.

Layers are added sequentially, with one following the other. Each layer is connected to the previous and the next layer in the stack.

## AWS EC2

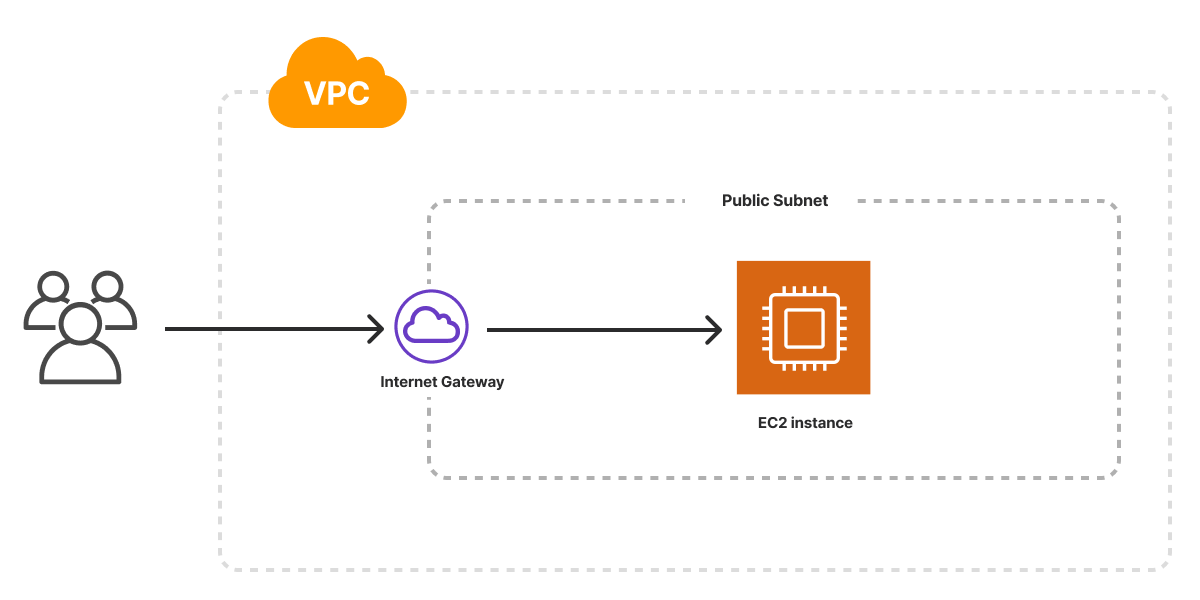


Figure 3: EC2 Architecture

Elastic Compute Cloud is a core and fundamental web service provided by Amazon Web Services (AWS). It enables users to rent virtual computing resources in the cloud, allowing them to run applications and workloads on a scalable infrastructure without the need to invest in physical hardware.

It allows me to create and manage virtual machines known as instances. These instances can be configured with different types of computing power, memory, and storage based on the user's requirements.

Moreover, it is secure because i have control over the security configurations of their EC2 instances. AWS provides various security features like security groups and key pairs to control network access and secure data.

## AWS S3 Bucket

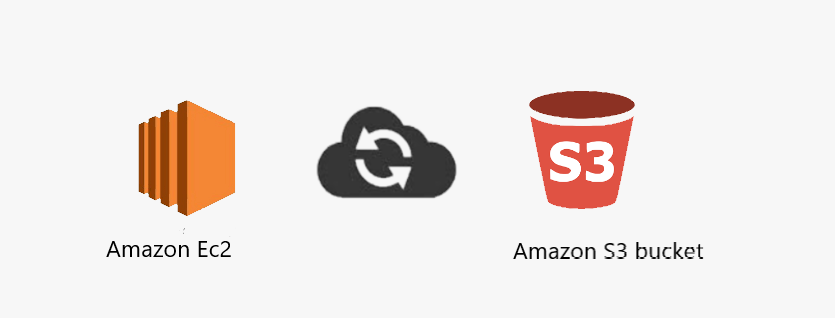


Figure 4: Amazon S3 bucket & EC2 relation

For storing the images that I will use to train the models I decided to use Amazon Simple Storage Service (Amazon S3) which is a scalable and highly durable object storage service offered by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data from anywhere on the web. Amazon S3 is commonly used for data storage, backup, and archiving, as well as for hosting static websites, distributing large amounts of data, and serving as a reliable and cost-effective storage solution for various applications.

The reason why I am using it is because S3 seamlessly integrates with other AWS services, making it a central component in many cloud-based architectures. It can be used in conjunction with services like AWS Lambda.

It also offers a range of security features, including access control mechanisms using bucket policies and Access Control Lists (ACLs), server-side encryption, and integration with AWS Identity and Access Management (IAM) for fine-grained access control.

## AWS Lambda

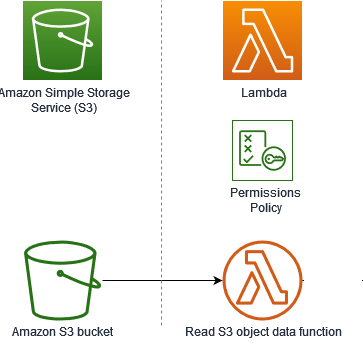


Figure 5: Usage of AWS Lambda with Amazon S3 Bucket

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). It allows you to run code without provisioning or managing servers, handling the infrastructure details, and automatically scaling based on the demand or the events triggering the function. With AWS Lambda, you only pay for the compute time consumed by your code.

Lambda functions automatically scale to handle varying workloads, providing high availability and performance.

It can be triggered by various AWS services and custom events, enabling you to build applications that respond to changes in data, execute code in response to HTTP requests, or perform other automated actions.

## AWS Cognito

A logo of a company

Description automatically generated

Figure 6: Authentication using Cognito

Amazon Cognito is a fully managed identity and user management service provided by Amazon Web Services (AWS). It enables developers to add user sign-up, sign-in, and access control to web and mobile apps quickly and easily. Here are some key features and components of AWS Cognito that made me choose this service:

* User Pools:

They enable you to create and maintain a user directory that can scale to hundreds of millions of users. They support various authentication methods, including username/password, social identity providers (such as Google, Facebook, and Amazon).

* Authentication and Authorization:

Cognito handles user authentication and authorization, including multi-factor authentication (MFA), password reset, and account recovery. It provides a secure way to authenticate users using industry-standard protocols such as OAuth 2.0 and OpenID Connect.

Cognito allows you to define fine-grained access control policies to manage user access to your application's resources.

* Secure and Scalable:

Cognito is built to be secure and scalable, handling millions of users and providing high availability and reliability.

It encrypts sensitive user data at rest and in transit, and it provides built-in protection against common security threats, such as brute-force attacks and cross-site scripting (XSS) attacks.

* Integration with Other AWS Services:

Cognito integrates seamlessly with other AWS services, such as Amazon API Gateway, AWS Lambda, Amazon S3 allowing you to build scalable and secure serverless applications.

## AWS Cloudwatch

A diagram of a company logo

Description automatically generated

Figure 7: Cloudwatch & Lambda

Amazon CloudWatch is a monitoring and observability service provided by Amazon Web Services (AWS). It helps you collect, monitor, and analyze metrics, logs, and events from your AWS resources and applications in real-time. Here are some key features and components of AWS CloudWatch that made me choose this service:

* Metrics:

CloudWatch collects and stores metrics data for your AWS resources and applications. Metrics are numeric data points that represent the performance, utilization, and behavior of your resources over time.

* Logs:

CloudWatch Logs enables you to collect, monitor, and analyze log data from your applications and systems. It allows you to centralize logs from multiple sources, such as Amazon EC2 instances, AWS Lambda functions, and containers running on Amazon ECS or Kubernetes.

When the user does an event, a lambda function is triggered which immediately sends a log containing the event details to cloudwatch as seen in figure7.

* Events:

CloudWatch Events enables you to respond to changes in your AWS environment and trigger automated actions in response to events. Events can be generated by AWS services, such as EC2 instance state changes or S3 bucket events.

## Git & Github

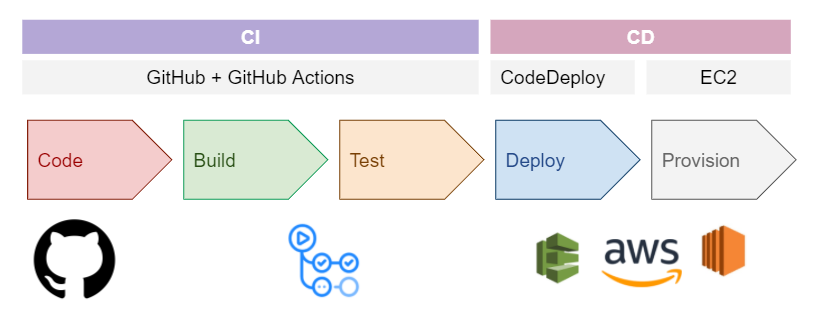


Figure 8: Github for continuous integration

For the version control system, I decided to use Git & Gitlab since our company uses Gitlab.

It is very secure since it uses SSH protocol to communicate with Git in a very secure way. we don’t need to use username and password each time since we are using SSH private & public keys to access the remote server of Gitlab.

Moreover, it helps in tracking the file changes and stores each update that you pushed so you will have a record of all actions done and that is why you can go back to specific versions in case you made a mistake in the most recent one.

It also supports CI/CD which helps us to continuously test, fix errors and improve the product.

It also makes the teamwork and collaboration simpler by allowing many changes coming from different sources to be merged into one source.

## Visual Studio Code

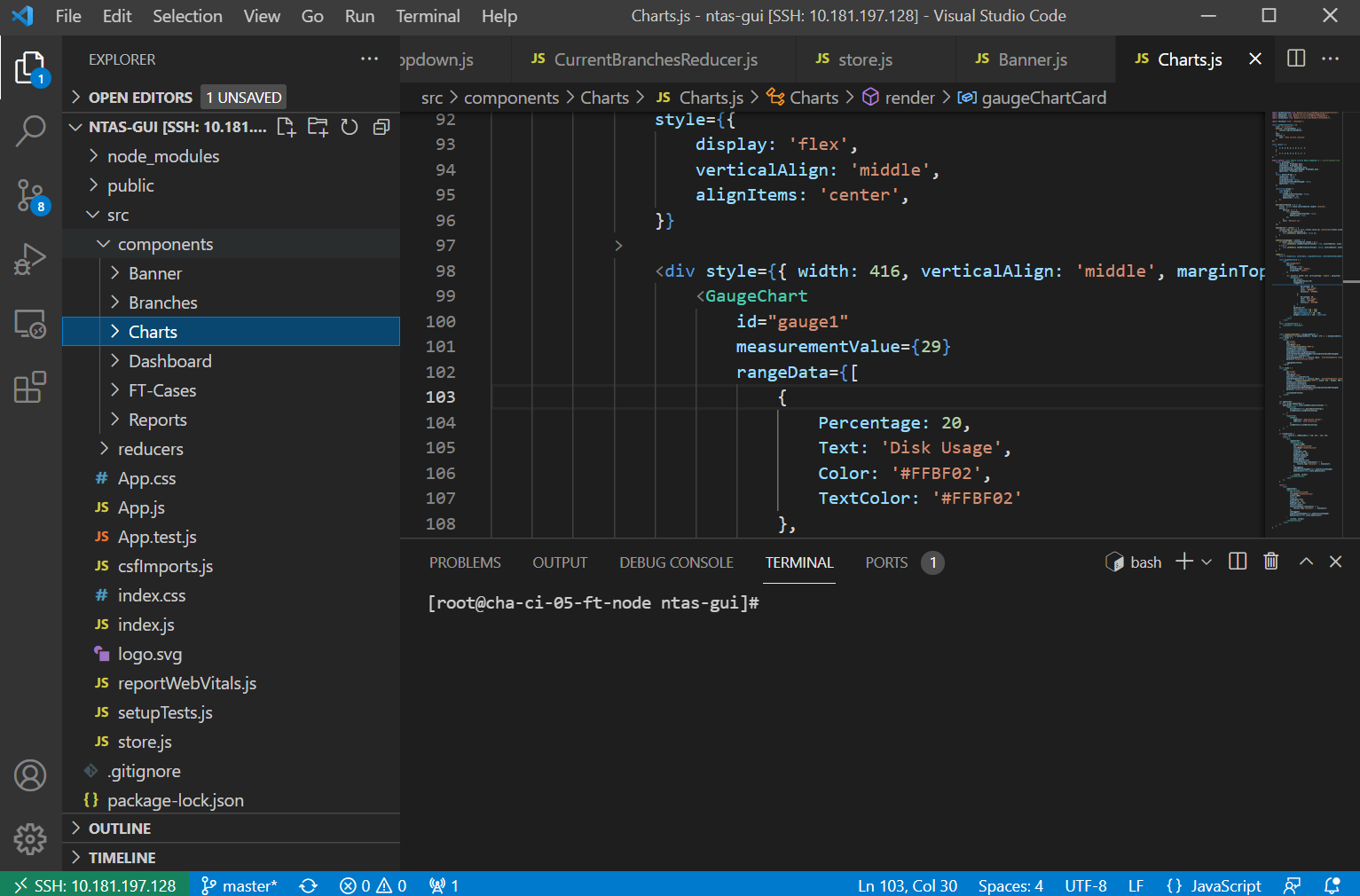


Figure 9: Visual Studio Interface

I decided to choose Visual Studio Code as an editor which is a lightweight yet powerful source code editor. It has a vast library of extensions that enhance its functionality for various programming languages, frameworks, and tools. Whether you're working with Python, JavaScript, Java, or any other language, you can find extensions to improve your development experience.

Despite its lightweight nature, VS Code provides powerful IDE features such as IntelliSense (code completion), debugging, version control integration (e.g., Git), and built-in terminal support. These features streamline the development process and boost productivity.

Since the Project is huge so it contains multiple files and components. Visual Studio Code helped me to manage components by separating them in different folders. It helped to keep a clear structure of the project in terms of organizing all the included items.

Moreover, the application will be deployed to an EC2 virtual machine and visual studio supports SSH connection to a virtual machine and allows us to access its content.

# Literature

The pharmaceutical sector places paramount importance on liquid viscosity measurement, utilizing tools such as capillary viscometers, orifice viscometers, rotational viscometers, vibrational viscometers, and ultrasonic viscometers. Viscosity, a key fluid property, profoundly influences processes critical to pharmaceutical manufacturing. Accurate viscosity determination is essential for achieving precise control and optimization in the production of pharmaceutical droplets.

The assessment of liquid viscosity holds significant importance in various industries. Various instruments, including capillary viscometers, orifice viscometers, rotational viscometers, vibrational viscometers, and ultrasonic viscometers, can be employed for this purpose. Among these, the capillary viscometer stands out as the most widely utilized due to its cost-effectiveness and simplicity. This method involves measuring the time required for a defined volume of liquid to traverse a narrow-bore tube under specific pressure conditions [7]. Despite their prevalence, these invasive methods are associated with high costs, rendering them unsuitable for continuous viscosity measurement conducive to in-process monitoring and timely interventions in case of errors. Consequently, there is a pressing need for a cost-effective and time-efficient alternative.

One promising approach for viscosity measurement involves the estimation of viscosity using machine learning and image processing algorithms based on liquid droplet characteristics. Numerous researchers have explored correlations between liquid viscosity and droplet properties. For instance, H. Zhu et al. established a correlation between extensional viscosity and spray droplet sizes in polymer spray solutions [8]. Gotaas et al. investigated the impact of viscosity on droplet-droplet collision outcomes [8]. Wang et al. demonstrated a logarithmic increase in droplet diameter in vertical gas-liquid annular flows with rising liquid viscosity [9].

In addition to these correlations, researchers have explored the application of image processing techniques for liquid viscosity measurement. Kheloufi et al., for instance, measured the fall height of a ball in falling ball viscometers by capturing video scenes during its descent and utilizing them to compute viscosity [10]. Santhosh et al. successfully estimated viscosity by capturing refracted images of a laser through a liquid-containing tube using a camera. These images were processed using thresholding, filtering, and histogram techniques, and an artificial neural network model was employed to establish the relationship between the resultant data and viscosity [11].

In addition to viscosity, the thesis explores machine vision-based methods for real-time quantification of ultralow drug content during continuous twin-screw wet/dry granulation and tableting. This extends the scope of AI application in pharmaceutical processes, aiming for enhanced accuracy and efficiency.

Solid oral dosage forms, such as tablets and capsules, can undergo various continuous processing methods like direct compression or dry granulation. Among these methods, wet granulation remains a widely adopted approach. Its popularity lies in its capability to produce agglomerated particles or granules with superior downstream properties, particularly beneficial for subsequent processes like tableting. Enhanced critical material attributes, including flowability, bulk density, and compactibility, contribute to the consistent production of high-quality tablets. In the context of low-dose drug products (where the Active Pharmaceutical Ingredient <API> content is below 2% w/w), additional challenges, such as meeting content uniformity requirements, necessitate careful consideration in selecting the appropriate manufacturing technology.

The advantages of continuous processing have led to a surge in studies and publications on Twin-Screw Wet Granulation (TSWG). These studies predominantly focus on evaluating the impact of specific process parameters, such as Liquid-to-Solid ratio (L/S), screw configuration, barrel fill level, and overall process settings. Beyond these technological considerations, the distribution of granulating liquid's significance and the exploration of implementing Process Analytical Technology (PAT) tools coupled with real-time process control have also been subjects of investigation.

Machine vision can be used as a tool for monitoring continuous twin-screw wet granulation, where the colored tracer would be dissolved in the granulation liquid alongside the API so their concentrations would simultaneously change.

In recent years, the integration of Artificial Intelligence (AI) and machine learning (ML) into pharmaceutical processes has emerged as a transformative force, offering unprecedented insights and efficiencies. We will focus on the pivotal role of cloud-based user interfaces in hosting machine learning models within the pharmaceutical industry.

The overarching goal is to explore how these interfaces facilitate seamless interaction, testing, training, and application of AI models, specifically tailored for tasks related to pharmaceutical droplets and granules.

# Requirements

## Actors Identification

Our application will only be used internally by me (user) and externally by the people interested in using a cloud-based machine learning tool in the pharmaceutical industry in order to facilitate the models’ trainings, testing and data analysis.

## Requirements Specification

To have a good functional cloud-based machine learning web platform that could serve the users, there are number of elements qualified as requirements for the project.

### Functional Requirements

* **AI Model Preparation:**

Develop Convolutional Neural Network (CNN) and Artificial Neural Network (ANN) models tailored for pharmaceutical droplets.

* **Viscosity Classification and Analysis:**

Train our AI models to do the classification of pictures based on the liquid viscosity.

* **Screw Speed classification and Analysis:**

Train our AI models to do the classification of the pictures based on the screw speed (revolutions per minute).

* **Data Study and Analysis:**

Collect relevant data for training and testing the AI models.

Perform thorough data analysis, including preprocessing and visualization, to enhance the quality of input data for the models.

* **Cloud-Based User Interface (Web App) Development:**

Design and develop a user-friendly cloud-based Graphical User Interface (GUI) hosted on a platform like AWS.

* **Results Representation:**

Implement a mechanism on the user interface to represent and display all relevant results generated by the AI models.

* **Testing of Models and Interface Functionalities:**

Conduct rigorous testing of the developed AI models to ensure their reliability.

Test the functionalities of the cloud-based user interface, including its responsiveness, ease of use, and overall performance.

### Non-Functional Requirements

In addition to functional requirements, the application also needs to satisfy these requirements to improve the quality of its functionality, such as:

* **Security:**

Data Protection: Ensure the security and confidentiality of pharmaceutical data, complying with relevant regulations and standards.

User Authentication and Authorization: Implement secure login mechanisms and access controls to protect against unauthorized use.

* **Performance:**

Response Time: The system should provide timely responses to user inputs, ensuring a responsive and efficient experience.

* **Scalability:** The solution should handle varying workloads and adapt to changes in user numbers or dataset sizes without compromising performance.
* **Simplicity**: Our application should be simple to use
* **User Experience:** User experience (UX) focuses on having a deep understanding of users, what they need, what they value, their abilities, and their limitations since the goal of our application is to ensure easing the job for the users.

# Design

## Use Cases

### Global Use Case

The detailed study of the specifications allows us to identify several use cases. The use case diagram visually represents the interactions between different actors and the system under consideration. In figure 3.1 below, we represent the main use cases to have a global view of the functioning of our application, as well as the possible interactions that can take place. The tasks of the actor (user) include managing the ANN models, managing the CNN model, Visualizing and comparing results, managing the AWS S3 bucket and finally creating and downloading Excel Datasets.

A diagram of a diagram

Description automatically generated with medium confidence

Figure 10: Global Use Case Diagram

### Manage ANN models

**Feed Models With Training Data**

**Feeding the model with diverse, representative, and well-labeled training data is crucial for training a model that generalizes well to new, unseen data. In our case, the user will be able to select an excel dataset stored locally which will be used by the application to build the needed dataframe.**

**In the context of machine learning, a data frame is a two-dimensional tabular structure that organizes data into rows and columns. Each row typically represents an observation or data point, and each column represents a feature or variable. The data frame serves as the input to the machine learning model, providing a structured format for training and testing.**

**For instance, our model will be able to classify images of medicine droplets based on their screw speed, so the training dataset would include parameters extracted from images of droplets that belong to different categories. The model learns from these examples by adjusting its internal parameters to minimize the difference between its predictions and the actual labels in the training data.**

**Profile Data**

Analyzing a dataset to determine its composition, quality, and structure is known as data profiling. Prior to starting more complex analysis or modeling operations, it helps data scientists and analysts in gaining understanding of the data. In our case, we will use Streamlit Data Profiling which is a feature within Streamlit that allows you to generate exploratory data analysis (EDA) reports for your datasets in a user-friendly and interactive way.

The user will be able to create an extensive report for his dataset with just one button click. Missing value analysis, distribution plots, correlation matrices, summary statistics, and more are frequently included in these reports. Users can dynamically explore the data by choosing variables or features of interest and seeing them visualized in real-time due to Streamlit's interactive nature.

This feature can be especially helpful in helping you quickly understand the properties of your data and see any potential problems or patterns that could need more research or preparation.

**Create & train Image Classifiers**

The user will be able to create and train 3 models: decision tree, K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP) that are used to classify images of droplets based on their rotation speed.

Decision Tree:

Decision tree is a popular machine learning algorithm used for classification tasks. It works by recursively partitioning the feature space into smaller regions based on the feature values. At each step, it chooses the feature that best splits the data into classes. Decision trees are easy to interpret and understand, making them a good choice for classification tasks.

K-Nearest Neighbors (KNN):

KNN is a simple and effective algorithm for classification tasks. It works by finding the k nearest data points to the input sample in the feature space and then assigning the majority class among those neighbors as the predicted class. KNN is non-parametric and lazy-learning, meaning it doesn't learn a discriminative function from the training data but instead memorizes the training instances.

Multi-Layer Perceptron (MLP):

MLP is a type of artificial neural network (ANN) with multiple layers of nodes (neurons) between the input and output layers. Each node in the network performs a weighted sum of its inputs and applies an activation function to produce the output. MLPs are capable of learning complex patterns in data and are widely used for classification tasks.

**Dataset Cleaning**

The user will be able to see the training/testing accuracy of each model and its performance metrics.

By making such comparisons, users can easily evaluate and compare the performance of different image classifiers, helping them make informed decisions about which classifier is best suited for their specific use case or dataset.

After analyzing the results, the user can go back and make some changes to the dataset by dropping certain columns that affected badly the training process of the models.

**Download Models**

Once Ready, the user will be able to download any model he would like in order to use it in the future. The generated classifier will have such a format (ex: decision\_classifier.pkl)

**Test Models**

Before Training and testing the model, the user should split the dataset into training, validation and testing set. The web application has a slider widget that allows the user to select the ratio of the dataset splitting (ex: 80% for a training & validation sets and 20% for a testing set).

The training set is used to train the model, the validation set is used to tune hyperparameters and assess model performance during training, and the test set is used to evaluate the final performance of the trained model.

The testing step provides an unbiased estimate of the model's performance on unseen data. It will calculate various evaluation metrics to assess the model's performance on the test set. Common metrics for classification tasks include accuracy and precision.

By systematically testing models and evaluating their performance using appropriate metrics, the user can ensure that your machine learning system performs well in real-world scenarios and generalizes effectively to unseen data.

**Load Existing Model**

The web application has a feature that allows the user to upload an already created ANN model which does the image classification based on rotation/screw speed that is stored in a file that has this format (ex: mlp\_classifier.pkl).

After loading the model, the user can apply testing on it and can afterwards evaluate the metrics displayed.

### Manage CNN model

**Feed Model With Training Images**

The user should have locally a dataset of medicine droplet images labeled with their corresponding viscosity levels. It's essential to have a diverse and representative dataset that covers a range of viscosity levels and variations in droplet appearance.

The Data preprocessing is already handled by the application to ensure that the images are in a format suitable for training the CNN. This may include tasks such as resizing the images to a consistent size, normalizing pixel values to a standard range (e.g., [0, 1]), and augmenting the dataset to increase its diversity and reduce overfitting.

The user should split its images into training, validation and testing sets.

**For instance, our model will be able to classify images of medicine droplets based on their viscosity. By learning from a large number of labeled images and adjusting its parameters based on the observed errors, the CNN gradually improves its ability to classify images and generalize to unseen data.**

**Create and Train Model**

The user will be able to create a sequential keras model that contains different layers. He can also choose the number of epochs for training his model.

During the training, the CNN model learns to extract relevant features from the images and make predictions based on these features. The training process involves iteratively adjusting the model's parameters to minimize the difference between its predictions and the true labels in the training data.

**Evaluate Metrics:**

Once the training is done, the user will evaluate the performance of the model using the displayed metrics. This step provides a quantitative measure of how well the models are performing.

The metrics displayed are:

loss\_value which is the value of the loss function, and accuracy which is the computed accuracy of the model on the test dataset.

Mean Squared Error (MSE):

MSE is one of the most common metrics used for regression tasks.

It calculates the average of the squared differences between the predicted and true values.

The formula for MSE is:

* is the predicted value of the 𝑖 th sample

Mean Absolute Error (MAE):

MAE is another common metric for regression tasks.

It calculates the average of the absolute differences between the predicted and true values.

The formula for MAE is:

Root Mean Squared Error (RMSE):

RMSE is similar to MSE but takes the square root of the average of the squared differences.

It is useful because it provides an error metric in the same units as the target variable.

The formula for RMSE is simply the square root of MSE:

​

Coefficient of Determination ():

It measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

It provides a measure of how well the model fits the data.

Its values range from 0 to 1, with 1 indicating a perfect fit and 0 indicating that the model does not explain any of the variability in the target variable.

The formula for ​ is:

* is the mean of the observed values.

**Download Model**

Once Ready, the user will be able to download the generated and trained CNN model in order to use it in the future. The generated file will have such a format (ex: my\_model.keras).

**Test Model**

The user should have a separate dataset for testing the CNN model. This dataset should be representative of the real-world data that the model will encounter in production but should not overlap with the training or validation datasets.

Once the testing is finished, the user can analyze the results obtained from testing the model. This includes examining the loss value and any evaluation metrics to assess how well the model performed on the test dataset.

**Load Existing Model**

The web application has a feature that allows the user to upload an already created CNN model which does the image classification based on viscosity that is stored in a file that has this format (ex: my\_model.keras).

After loading the model, the user can apply testing on it and can afterwards evaluate the metrics displayed.

### Visualize Results

The application provides some statistics and data collection to the user which will allow him to have an overview on the data correlation, the predicted values versus the actual ones, graphs to show how the model looks like from inside in terms of layers.

The training, validation and testing error measurements, prediction and accuracy values will be also measured and displayed.

### Manage Homepage

**Manage AWS S3 Bucket**

The user will be able to store pictures, excel files and models to S3 buckets using the web interface or manually within the AWS console.

handling different file formats appropriately should be ensured.

The user will be also able to download the pictures by selecting a category (ex: Viscosity\_CNN) in order to receive the sets of images that are already prepared and that can be used for training and testing the model or the generated excel files that were previously preprocessed and uploaded.

Appropriate permissions will be set for the uploaded objects. Configuring access control lists (ACLs) or bucket policies may be needed based on the security requirements.

**Create and Download csv datasets**

The user will be able to create a csv dataset from the droplet images he uploads to use later for training the ANN image classifiers.

The user can also download already existing and prepared datasets from the S3 bucket instead of creating a new one.

### Authentication

Authentication is a process that verifies the identity of a user, system, or entity attempting to access a particular resource or service. It is a fundamental aspect of information security and is crucial for ensuring that only authorized individuals or systems can gain access to protected resources. Authentication is typically achieved through the use of credentials, such as usernames and passwords, tokens, or digital certificates.

The user should be able to login using the username and password.

I chose to utilize AWS Cognito due to its robust and secure authentication features. AWS Cognito provides a comprehensive solution for managing user identities and access control, offering seamless integration with various web and mobile platforms. By leveraging AWS Cognito, I am able to ensure that my application's authentication process is reliable, scalable, and adheres to industry best practices. With features such as multi-factor authentication, user sign-up and sign-in functionality, and user attribute management, AWS Cognito simplifies the complexities of user authentication, allowing me to focus more on the core functionalities of the application.

## Machine Learning Part Design

A pie chart with numbers and symbols

Description automatically generated

Figure 11: machine learning part design

### Viscosity Model:

1. Define the Objective:

The objective is to develop a CNN-based model for accurately predicting the viscosity values of droplets of water-PVP solutions. By training the CNN model on a dataset of droplet images, the model will learn the patterns and features that correspond to different viscosity values. Once trained, the model will be able to classify new droplets based on their images and predict their viscosity values with high accuracy.

2. Collect and Prepare Data:

Our data is a collection of pictures that are split into 11 classes based on different ratios of water PVP solutions. Each class has 3 subclasses called AfterDetachment, AtDtachment & BeforeDetachment. For training & testing I have chosen to use the BeforeDetachment pictures in each category.

The viscosity of the various samples was determined using an Anton Paar DMA 4500 M viscometer. Through the samples in the capillary tube, we calculated the rolling time of a steel ball with a 1.5 mm diameter. The capillary's angle was 45 degrees, and the measurement's temperature was 25 C.

3. Define Metrics for Success:

The metrics that will be measured are the prediction accuracy, MAE, MSE, RMSE and R2 score

4. Split the Data:

* Training data size: 3784 images.
* Validation data size: 946 images.
* Testing data size: 1271 images.
* Viscosities: the data is collected in an Excel spreadsheet that looks like this in the picture down below

A table with numbers and a number of objects

Description automatically generated with medium confidence

Figure 12: viscosity data

5. Select Algorithm:

The chosen algorithm Convolutional Neural Network (CNN) which is a Deep Learning algorithm that can take images as input and identify different objects in the image by giving them learnable weights and biases. CNN needs a lot less preprocessing than other algorithms for classification.

6. Train the Model:

Train the selected model using the training dataset. Adjust hyperparameters as needed. Monitor the training process and use validation data to prevent overfitting.

7. Evaluate Model Performance:

Assess the model's performance on the test set using the predefined metrics. Analyze the results and iterate on the model or data preprocessing if needed.

8. Deployment:

Prepare the model for deployment. This involves converting the trained model into a format suitable for deployment, and integrating it into the web app.

### Twin-screw Model

1. Define the Problem:

The objective is to develop machine learning models for image classification based on the screw speed. Artificial Neural networks algorithms will be applied and also Convolutional Neural network algorithm will be applied and then we will compare the results.

2. Collect and Prepare Data:

Our data is a collection of pictures which are splitted to 6 classes based on the screw speed (revolutions per minute):

5.7 rpm: 2796 pictures

5.7 rpm vissza: 3193 pictures

6.1 rpm: 2908 pictures

6.1 rpm vissza: 2876 pictures

6.5 rpm: 3011 pictures

The pictures contain a dark background were on top there are some captured dried granules that has different sizes. Those granules were obtained after applying wet granulation with basic screw speeds: 5.7, 6.1, 6.5 rpm.

3. Define Metrics for Success:

The metric that will be measured is the prediction accuracy.

4. Split the Data:

we split our data to training and validation data (25% of the pictures will be used as validation data). For testing, 6007 pictures will be used.

5. Select a Model:

Three Artificial neural networks (ANN) algorithms were used which are MLPClassifier, Decision Tree Classifier and Kneighbors Classifier.

6. Train the Model:

Train the chosen model on the training data. During training, the model learns patterns and relationships within the data. This involves adjusting the model's parameters to minimize a loss function, which quantifies the difference between predicted and actual values.

7. Evaluate Model Performance:

Evaluate the model's performance on the test set to assess its generalization to new, unseen data. Common evaluation metrics include accuracy, precision, recall, F1 score, and others, depending on the problem.

8. Deployment:

Once satisfied with the model's performance, we will deploy it to our web application where it can be trained, tested and make predictions on new real-world data.

### Class Diagram

A computer screen shot of a computer program

Description automatically generated with medium confidence

Figure 13: Class Diagram

## Web Application Part Design

My web application is a python streamlit application which is deployed on AWS cloud using services like EC2, lambda, identity and Access Management and S3 bucket.

Other services like Cognito, Lambda, Cloudwatch, Identity and Access Management and S3 buckets were used.

### Application Architecture

A diagram of a computer program

Description automatically generated

Figure 14: web application architecture

**User Interface (UI) Layer:**

Streamlit Script: The core of a Streamlit application is a Python script. This script contains the code for defining the UI components and specifying the app's behavior. Streamlit scripts are typically written in a top-down, procedural style.

Widgets and Components: Streamlit provides a range of widgets (e.g., sliders, buttons, text inputs) that you can use to interact with your application. These widgets are used to gather user input and trigger actions in response.

**Application Logic Layer:**

Data Processing: it includes data loading, preprocessing, splitting images, creating dataframes and excel files. This layer grants access to certain data from a data source like AWS S3 bucket with which we can apply CRUD operations like Creating, Reading, Updating and Deleting data.

Business Logic: Streamlit scripts often include business logic for handling user input, making decisions, and triggering actions. This logic is usually embedded within the script and executed in response to user interactions. Moreover, The application will handle the data retrieving and uploading from and to AWS S3 bucket.

**Visualization Layer:**

Charts and Visualizations: Streamlit allows you to integrate data visualizations seamlessly. Libraries like Matplotlib and Plotly will be used to create charts and plots. The visualizations are embedded within the Streamlit script and are updated dynamically based on user input.

**Integration with External Services:**

Data Sources: Streamlit applications can interact with external services, databases, or APIs such as AWS S3 bucket, AWS lambda service. Data can be loaded from these sources within the application logic layer.

Machine Learning Models: trained model can be integrated for predictions or classifications. Streamlit can be used to showcase model results and predictions.

**Deployment Layer:**

Deployment Platform: Our Streamlit application will be deployed on AWS using the EC2 service alongside other services like lambda, s3 bucket and identity and Access Management. The deployment process is simplified, and Streamlit apps can often be deployed with minimal effort.

### Data Management

In our application we will be using a lot of pictures and excels containing the data needed for training. Which means that we need to find a way to store these data somewhere instead of uploading them manually each time we want to train a model. For this purpose, I have decided to use AWS S3 Bucket alongside AWS Lambda to manage the data including both data upload and retrieval based on user input.

A close-up of a data transfer

Description automatically generated

Figure 15: Data Management

**1. Data Upload (User Input):**

* Create an S3 Bucket:

Begin by creating an S3 bucket to store the data. This can be done through the AWS Management Console or programmatically using the AWS CLI or SDKs.

* Configure Bucket Policy:

Set up a bucket policy or Access Control Lists (ACLs) to control who can upload data to the bucket. This ensures that only authorized users can write to the bucket.

* User Uploads Data:

When a user uploads data, a Lambda function is invoked to handle the upload event logging. The users can also upload data to the S3 bucket through the AWS Management Console.

**2. Data Retrieval (User Input):**

* User Requests Data:

Users request specific data based on their input. This input will be passed through the user interface. They can download folders of pictures related to each type of model, excel datasets and also trained models.

* Lambda Function invoked:

When a user requests data, a Lambda function is triggered to manage the uploading event logging.

* User Receives Data:

The S3 bucket sends the requested data from based on the user input and returns it to the user so it can be downloaded locally.

### Logging System

Logs consist of records or entries generated by software applications, systems, or devices to capture information about events, activities, errors, or other relevant occurrences. Log data is essential for troubleshooting, monitoring, auditing, and analyzing the behavior and performance of systems and applications. It often includes timestamps, event descriptions, error codes, user actions, system responses, and other contextual information that can be useful for understanding the operational status and diagnosing issues within a system or application.

In my case, the logging system will be implemented through AWS CloudWatch and Lambda which involves the generation of log data each time an upload or download operation occurs within an Amazon S3 bucket. When such an event takes place, a Lambda function is triggered to handle it. This Lambda function is configured to write a log entry detailing the specifics of the event to AWS CloudWatch Logs.

A logo with arrows pointing to the center

Description automatically generated with medium confidence

Figure 16: logging with cloudwatch and lambda

# Implementation

## Structure

A screenshot of a computer

Description automatically generated

Figure 17: files & folders structure

Each part of the web application is in a file that is packed in a separate folder to make dealing with changes in the future much easier.

The code is divided into classes that contains the necessary functions for data processing, model training & testing, metrics calculation and results visualization.

As I mentioned previously, for building my sequential Machine Learning models I decided to use Keras and TensorFlow.

For the web application, I used python streamlit library.

The Homepage.py file is the main entry point of the application.

## User Interface/Main Components

We can see that this layer is composed of several components which will outline the view.

### Sidebar



A hand with a blue background

Description automatically generated

Figure 18: Navigation Sidebar Design

The side drawer as seen on Figure 4.2 contains all the necessary tabs for the user to be able to navigate to different pages of the web platform. Whenever the user clicks on a certain tab, the script related to that page will be loaded.

### Login Page

The login page was initially designed and created using streamlit components and AWS Cognito service.

On AWS Cognito I managed to create a user pool that allows authentication using username and password where the password should follow some minimal requirements like at least 8 characters length, contains at least 1 number, 1 special character, 1 uppercase and 1 lowercase letter.

A screenshot of a password policy

Description automatically generated

Figure 19: password requirements

The multi factor authentication is not enabled but it can be enabled in the future.

The user account recovery can be done through email only by clicking on forget password, entering your email and then you receive a recovery mail.

A screenshot of a computer

Description automatically generated

Figure 20: Account recovery

Configuring how the user pool sends email messages to users in AWS Cognito involves setting up email settings within the user pool settings. This process enables Cognito to send various types of email messages to users, including verification codes for sign-up, password reset instructions, and other account-related notifications.

A screenshot of a computer screen

Description automatically generated

Figure 21: sending emails to users

The user can create his account through the website, but initially I added a user to the user pool manually through the aws console.

A screenshot of a computer

Description automatically generated

Figure 22: User pool

Once the account is created, the user will receive an email notification containing his details and looking like this:

A close-up of a password

Description automatically generated

Figure 23: notification email

When the user enters his username and password, a login request is sent to check if the user exists and if his credentials are correct or not. If they are correct the login form disappears, and the user is redirected to the homepage. If the credentials are wrong or the user doesn’t exist, then an error message will appear stating the issue.



Figure 24: successful login

The login function:

def login():

    st.title('Login')

    username = st.text\_input('Username')

    password = st.text\_input('Password', type='password')

    if st.button('Login'):

        try:

            # Authenticate user

            response = cognito\_client.admin\_initiate\_auth(

                UserPoolId='us-east-1\_63QQlSUm0',

                ClientId='6akm6miipm3qlsp8f7vkjfq9kh',

                AuthFlow='ADMIN\_USER\_PASSWORD\_AUTH',

                AuthParameters={

                    'USERNAME': username,

                    'PASSWORD': password

                }

            )

            st.write(response)

            st.success('Login successful!')

st.session\_state.user\_logged = True

 # Access token can be retrieved from response['AuthenticationResult']['AccessToken']

        except ClientError as e:

            st.write(e)

            st.error('Login failed. Please check your credentials.')

The login Form:

A blue and white stripes

Description automatically generated

Figure 25: Login Form

### Homepage

The homepage contains 3 main parts:

**Uploading Data**

This page is dedicated to upload training & testing images, models and datasets to save them in our S3 Bucket. The user should initially select the type of upload ('Images', 'Models', 'Datasets'). For Models and Datasets, it is straight forward but if the user selected Images, then he should choose the name of the folder ('RotationANN', 'ViscosityCNN').

If he selects 'RotationANN', then he has to write down the screw/rotation speed:

A screenshot of a computer

Description automatically generated

Figure 26: Upload Files

If he selects 'ViscosityCNN', then he must select between 2 sub-folders ('training', 'testing'):

A blue and pink stripes

Description automatically generated

Figure 27: sub\_folders selection

Once the data is uploaded to the S3 Bucket, the user will receive a message stating that the files were successfully uploaded.

The upload operation is implemented using the boto3 python library which provides an interface to interact with Amazon S3. Before each operation, the file types will be checked to prevent the cases where the user mistakenly uploads excel files in the image’s sections and vice versa.

The Images section accepts only .png type of files, for the datasets a .csv type and for the models .pkl

**Downloading Data**

This page is dedicated to downloading datasets, models, training & testing images for both model types. The user should select the folder he would like to download, and all the content will be there locally. Once the download is finished, a message stating that the operation was successful will appear.

A screenshot of a computer

Description automatically generated

Figure 28: File Download

On the Amazon console, if we check the S3 Bucket we will find all the folders and categories listed there. As an admin, I can make changes through the console like uploading or downloading manually or deleting unused elements.

A screenshot of a computer

Description automatically generated

Figure 29: S3 Bucket structure

Each action done will trigger a lambda function call that will log the event using the Amazon CloudWatch service.

A close-up of a computer screen

Description automatically generated

Figure 30: Event notification using Lambda

**Creating Datasets**

This page is dedicated to creating excel datasets for the Artificial Neural Network image classifiers.

The user can select which rotation/screw categories he would like to use to generate the data.

For example, he can pick from the 3 main categories (5.7 rpm, 6.1 rpm, 6.5 rpm)

A screenshot of a computer

Description automatically generated

Figure 31: Dataset (csv) Creation

The user can also add new custom categories by following the same name format (x.y rpm).

A screenshot of a chat

Description automatically generated

Figure 32: adding a new category

Once you click start, the dataset generation will be launched, a status widget will appear to give you an idea what is happening in the background and that you have to wait for the process to finish:

A white rectangular object with black lines

Description automatically generated

Figure 33: Dataset Generation Spinner

When the whole process is finished, a data frame will appear which contains data extracted from the pictures such as:

("Folder", "Average area of granules", "Size of largest area", "Size of smallest area", "Average perimeter of granules", "Size of largest perimeter", "Size of smallest perimeter", "Average Blue of Granules", "Average Green of Granules", "Average Red of Granules", "Average Solidity", "Average Orientation Angle")

A screenshot of a data sheet

Description automatically generated

Figure 34: Dataframe

### Twin Screw Page

The Twin Screw is a page that handles the upload of the dataset (excel, csv file), the pandas dataframe generation, the profiling of the data inside the dataframe, the classifiers training and testing then the option of downloading them.

It contains radio buttons that will lead us to the sub-pages:

* Upload
* Profiling
* Modelling
* Download
* Load Existing Model

**Upload Dataset**

On this page, a file\_uploader component will handle the upload of the dataset then a dataframe will be created and displayed underneath it.

A screenshot of a data sheet

Description automatically generated

Figure 35: upload a dataset

Cleaning a dataframe is very important for several reasons like:

* Data Quality: Cleaning ensures that your data is accurate, complete, and consistent. This improves the quality of your analysis and the reliability of your results.
* Removing Missing Values: Data often contains missing values, which can affect the performance of machine learning models or statistical analyses.
* Removing Duplicates: Duplicates in data can lead to biased results or inaccurate statistics. Cleaning involves identifying and removing duplicate records or observations.

The web application provides a feature that allows the user to drop the columns he wants after analyzing or testing the dataframe.

A screenshot of a data analysis

Description automatically generated

Figure 36: Cleaning a Dataframe

In our case based on what we see in the figure 35 we notice that the size of smallest area and smallest perimeter contains mostly 0’s so it won’t be relevant for training our model, so it is better to drop them.

Also, after converting the csv file to a pandas dataframe an index column named (Unamed:0) was generated. This column doesn’t give any good information to the models so I will drop it.

Based on the generated correlation heatmap, we notice that the average blue and the average green, the average blue and the average red plus the average green and the average red have a very strong correlation, so I decided to drop the average red and/or blue columns.

Also, the size of the largest area and the size of the largest perimeter granules have a very high positive correlation (almost 1), so we can drop one of them.

If two or more columns are highly correlated (i.e., they contain similar information), keeping both columns may lead to multicollinearity issues in statistical analyses or redundant information in machine learning models. In such cases, you may choose to drop one of the correlated columns.

A screenshot of a computer

Description automatically generated

Figure 37: Before cleaning

**Data Profiling**

After clicking on profiling a whole report will be generated from that data using this function:

from streamlit\_pandas\_profiling import st\_profile\_report

streamlit\_pandas\_profiling library is used to generate and display a pandas profiling report within a streamlit app. This is particularly useful for quickly exploring and visualizing the characteristics of a dataFrame.

The st\_profile\_report function takes a Pandas dataframe (data) as an argument and generates a pandas profiling report for that DataFrame. This report includes various statistics, visualizations, and insights about the data, such as summary statistics, distribution of values, missing values, correlations, and more.

Some parts of the whole report are shown below. One of them shows the new correlation matrix after cleaning the data which is a visual representation of the correlation between numeric variables. Each cell in the table represents the correlation between two variables.

The values range from -1 to 1.

Correlation matrices help in identifying patterns and relationships between variables.

The new heatmap seen on figure 38 contains less high correlated elements:

A screenshot of a computer

Description automatically generated

Figure 38: After Cleaning

The next graph on figure 39 shows how these parameters are interacting between each other.

The interaction graph allows the user to visually inspect how the relationship between the variables changes based on the levels of the other variable. It helps in understanding whether the effect of one variable depends on the level of another variable. If there is a significant interaction effect, the lines or bars will not be parallel.

A screenshot of a graph

Description automatically generated

Figure 39: Interactions Graph

The next figure 40 gives an overview of the dataset statistics like number of variables, variable types.

A screenshot of a data analysis

Description automatically generated

Figure 40: Dataset Statistics

**Modelling (Creating, Training & Testing Image Classifiers)**

On this page, the user will be able to modify the data split ratio by giving how much percentage he is willing to give for the training and validation set and how much percentage will be left for the testing set as seen on figure 41.

A screenshot of a computer

Description automatically generated

Figure 41: data split ratio for image classifiers

After setting the ratios, the user can click Start Modelling where the image classifiers will be created, trained and tested their metrics scores will appear:

A screenshot of a cell phone

Description automatically generated

Figure 42: classifiers accuracy value

These results are obtained after dropping the average red and the size of largest granules and using a large dataset.

As figure 43 shows, the user can also see additional metrics related to each classifier like the confusion matrix and the classification report which contains precision, recall and f1-score.

A screenshot of a graph

Description automatically generated

Figure 43: Additional Metrics

On the next figure we notice a graph that shows a comparison between the 3 classifiers accuracy:

A graph of different colored rectangles

Description automatically generated

Figure 44: Accuracy Comparison

Testing an Artificial Neural Network (ANN) model involves evaluating its performance on a separate dataset that it hasn't seen during training. In our case, we are testing it by default on 20% of the dataset we loaded first. Here's how the user should proceed step by step:

* Prepare Test Data:

Separate your dataset into features (inputs) and labels (outputs).

Ensure that the test dataset is distinct from the training dataset to avoid bias.

* Train Model or Load a Pre-Trained one:

Load the trained ANN model that you want to evaluate.

* Perform Prediction:

Use the loaded model to make predictions on the test dataset.

Pass the features of the test dataset through the model and obtain the predicted outputs.

* Evaluate Performance:

Analyze the outputs and the metrics.

* Iterate if Necessary:

If the performance is not satisfactory, you may need to go back to training and fine-tuning your model.

Adjust model hyperparameters, try different datasets.

**Download Items**

On this page, the user can download a trained classifier after analyzing and comparing their accuracy and metrics. As we can see on figure 45, when the download starts, an animation will appear on the screen and after it ends a success message will appear stating that the model was downloaded. The model can be found locally stored in a file that has such a name (mlp\_classifier.pkl).

A screenshot of a computer

Description automatically generated

Figure 45: Download Classifier

The user can also download a cleaned dataframe if he made some changes to the original dataframe by dropping certain columns. On the figure 46, we can see that an animation starts when the download process starts and after it is done, a success message will appear on the screen.

A computer screen shot of a computer chip

Description automatically generated

Figure 46: Download Cleaned Dataframe

**Load Existing Model**

This page allows the user to just display the trained model testing data and metrics or to upload an existing model and test it using the selected dataset.

If the user keeps the toggle off as seen on figure 47, he can select a model that was previously trained, and it will display its related testing metrics.

If the models were not trained, it will display a warning saying that the user should train the selected model before testing.

A screenshot of a test

Description automatically generated

Figure 47: use a trained model

The user can also turn the toggle on and load an already existing and trained classifier.

A screenshot of a computer

Description automatically generated

Figure 48: Load Model

### Viscosity Page

The Viscosity page handles the upload of images to create and train a CNN model, downloading it locally, loading the model and evaluating its performance.

It contains 2 subpages that we can reach using a radio button on the side bar:

* Create Model
* Upload & Test Model

A screenshot of a screenshot of a robot

Description automatically generated

Figure 49: Radio Button

**Create Model**

Here, the user can upload his images to train the model. Once the images are uploaded, the first 10 will be displayed on the web page with their viscosity value as a caption to each one of them.

A group of light bulbs

Description automatically generated

Figure 50: First 10 images

If no image is uploaded a warning message will appear.



Figure 51: Warning message

\After that, the user can modify the training/validation ratios before starting the training process.

The default ratios are 80% for training set and 20% for validation.

A screenshot of a computer

Description automatically generated

Figure 52: Set training and testing ratios

As we can see on the figure 53, a report will be generated about the training and validation data size, how many samples are in each viscosity category alongside the viscosity value.

A screenshot of a data

Description automatically generated

Figure 53: Training Report

The user can also set the number of epochs before starting the training. The number of epochs in training a model refers to the number of complete passes through the entire training dataset during the training process. An epoch is a single pass through the full dataset, after which the model's parameters (weights and biases) are updated based on the learning algorithm being used (e.g., gradient descent). In our case we will set it to 100.

A blue and pink stripe

Description automatically generated

Figure 54: Set epochs number

The process starts by creating a keras sequential model with:

* Input layer with shape (91, 53, 3)
* Convolutional layer (32 filters, 3x3 kernel, ReLU activation)
* MaxPooling layer (2x2 pool size)
* Flatten layer (converts 2D matrix to 1D vector)
* Dense layer (64 units, ReLU activation)
* Dropout layer (50% dropout rate)
* Output Dense layer (1 unit, ReLU activation)

The model layers will be plotted on the web page as seen on figure 55:

A diagram of a function

Description automatically generated with medium confidence

Figure 55: Model Design

An early stopping callback function is implemented to stop training if the validation loss does not improve for 15 epochs.

The model is compiled with the Adam optimizer and Mean Squared Error (MSE) loss function.

The training starts on the training data, validation on the validation data, and it uses the early stopping callback.

If training history is not empty, plots the training and validation loss over epochs using Matplotlib as we can see on figure 56. We notice that in our case the MSE loss dropped with each epoch and it went from 1 till 0.15 and the validation MSE loss went from 1 to 0.05 which indicates that the model's predictions are getting closer to the actual values.

A graph of loss and loss

Description automatically generated

Figure 56: MSE Loss Graph

Later, the model prediction and evaluation on both training and validation data starts and the results will be displayed as shown in the next figure 57.

A white and purple background with black text

Description automatically generated

Figure 57: Error on data

The training error measurements:

A screenshot of a error

Description automatically generated

Figure 58: training metrics

The validation error measurements:

A screenshot of a error

Description automatically generated

Figure 59: validation metrics

When the training completes, the model will be saved locally in a keras file named like this (my\_model.keras).

**Upload and Test Model**

This page focuses on uploading an already trained model and a testing dataset of images to test the model and display the metrics for the user to analyze it.

Once the files are loaded, the prediction on the testing dataset will start then the web application will display the results. We can see that on the figure 60:

A screenshot of a computer

Description automatically generated

Figure 60: Viscosity Testing Results

The UI will also display a graph that shows the predicted values vs the actual ones so the user can see if the model performed well or not. In the first graph on figure 61, we notice that the model performed well since the predicted values are very close to the actual ones.

In the second graph on figure 62, from 0 to 400 sample the predicted ones were accurate but for the other samples from 400+ the values were not close to the actual ones, so the model didn’t perform well. There can be an issue with the training, or the model need to be further trained with a wider dataset.

A graph with blue dots and red lines

Description automatically generated

Figure 61: predicted vs actual (1)

A graph with blue lines and white text

Description automatically generated

Figure 62: predicted vs actual (2)

# Deployment

Deploying machine learning (ML) models as web applications has become increasingly important for researchers and enterprises in the current era of digital transformation. The deployment procedure guarantees the ML model's scalability and accessibility in addition to demonstrating how it may be used in real-world scenarios.

This section delves into the deployment of our ML-based web application on an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance, detailing the architecture, deployment process, and the benefits accrued.

## Architecture Overview

Our web application on AWS EC2 is built to ensure scalability, robustness, and ease of maintenance. Fundamentally, the architecture is made up of:

1. EC2 Instance: A virtual server hosted in the AWS cloud, providing computing resources to run our application.
2. Web Application: The ML-based web application that we developed using python streamlit, serving as the user interface to interact with the ML models.
3. ML Models: The trained machine learning model, integrated into the web application to perform predictions or analyses based on user input.
4. Data Storage: AWS S3 bucket to store the images, the generated dataframes and models .
5. Load Balancer & Auto Scaling: To manage traffic and ensure high availability, an Elastic Load Balancer (ELB) is employed along with Auto Scaling groups, allowing the application to scale horizontally based on demand.

## Infrastructure Setup

The initial phase of our deployment journey revolved around establishing a robust infrastructure on AWS to host our web application and ML model. The following steps encapsulate the infrastructure setup:

1. Virtual Private Cloud (VPC) Creation: A VPC called thesis\_vpc was created to provide an isolated and secure environment for our application. This VPC acts as our virtual network, allowing us to define IP address ranges, subnets, and route tables.

A screenshot of a computer

Description automatically generated

Figure 63: created vpc

1. Public Subnet Creation: Within the VPC, a public subnet was established with the CIDR block 10.0.0.0/24. This CIDR block provides a range of 256 IP addresses, allowing ample room for our EC2 instance, Internet Gateway, and other resources. The subnet was designated as public to ensure our EC2 instance has a public IP address and can communicate directly with the internet.

A screenshot of a computer

Description automatically generated

Figure 64: created subnet

1. Internet Gateway (IGW) Creation: An Internet Gateway was created to enable communication between our VPC and the internet. The IGW serves as a gateway to route traffic from our EC2 instance to the outside world and vice versa.

A screenshot of a computer

Description automatically generated

Figure 65: created internet gateway

1. Attachment of Internet Gateway to VPC: The Internet Gateway was attached to our VPC to allow internet access to resources within the VPC. This attachment ensures that our EC2 instance can communicate with the internet for software updates, data retrieval, and other essential operations.
2. Route Table Creation: A route table was established with a route for the CIDR block 10.0.0.0/16. This route table acts as a set of instructions for our VPC, directing traffic within the 10.0.0.0/16 range. The broader CIDR block allows flexibility for future expansion, accommodating additional subnets and resources within the 10.0.0.0/16 range.

A screenshot of a computer

Description automatically generated

Figure 66: created route table

1. Adding Route to Internet Gateway in Route Table: A route to the Internet Gateway was added to the route table. This route ensures that all outbound internet-bound traffic from our EC2 instance is directed through the Internet Gateway, allowing our application to communicate with external services and users.

## Deployment Process

The deployment process of our ML web application on AWS EC2 can be segmented into several key steps:

1. Instance Provisioning: Initially, an EC2 instance is provisioned with the appropriate compute, memory, and storage resources based on our application's requirements. An Amazon Linux with a 64-bit (x86) architecture was selected as an Amazon Machine Image (AMI) to serve as the base for our application environment. The instance type i selected was t2.micro which provides 1vCPU and 1GiB Memory. The storage was set to 8 GiB general purpose SSD (gp3) which is a type of storage volume offered by Amazon Web Services (AWS) for its Elastic Block Store (EBS) service. EBS provides block-level storage volumes for use with EC2 instances. The General Purpose SSD (gp3) volume type is designed to offer reliable and cost-effective performance for a wide range of workloads. Moreover, A key pair (public-private) was created in order to allow us to ssh into the instance. The SSH traffic was allowed from anywhere (0.0.0.0/0).

I create a security group where the Inbound rules are set this way:

* port 3398 for connections through Remote Desktop Protocol (RDP):
* port 22: enables Secure Shell (SSH) interaction and permits virtual machine (VM) remote administration access.
* Port 80: It functions as the communication gateway for HTTP requests and responses between server computers and clients and runs on the application layer of the TCP/IP networking model.
* Source: Anywhere (0.0.0.0/0): allows traffic from any IP address



Figure 67: Inbound rules

And 1 Outbound rule by default which is associated to all types of protocol and sends traffic anywhere using any port or port range:

A screenshot of a computer

Description automatically generated

Figure 68: outbound rules

1. Environment Setup: The necessary software and libraries, including Python, Python related packages, Linux packages and other dependencies required for our ML models and web application, are installed on the EC2 instance. For the Python related packages, i created a file called **requirements.txt** which contains everything needed. I also created a script called **setup\_external\_libs.sh** which acts as an entrypoints to the virtual machine. The script starts by granting rights to these 2 files so they can be executed and used, then updates apt-get, install all the necessary packages and graphviz which is used for plotting graphs and images, creates an aws config file that contains my access id and key and finally start the web application.

A screen shot of a computer

Description automatically generated

Figure 69: requirements.txt

A screenshot of a computer program

Description automatically generated

Figure 70: setup\_external\_libs.sh

1. Application Deployment: The web application code and all the files are deployed to the EC2 instance. This involves cloning the from github repository, setting up the directory structure, and configuring the application to run on the chosen web framework.
2. Load Balancer & Scaling Configuration: An Elastic Load Balancer (ELB) is configured to distribute incoming traffic across multiple EC2 instances. Auto Scaling groups are set up to automatically adjust the number of instances based on the load, ensuring optimal performance and availability.
3. Testing & Monitoring: Rigorous testing is conducted to validate the application's functionality, performance, and scalability. Monitoring tools like Amazon CloudWatch are configured to track metrics and logs, set alarms, and gain insights into the application's health and performance. In Addition, 2 AWS lambda functions were created to control and log the details of any upload or deletion that happens on our S3 bucket. All the logs can be checked on Amazon CloudWatch.

## Benefits of AWS EC2 Deployment

Deploying our ML web application on AWS EC2 offers a myriad of benefits:

* Scalability: With Auto Scaling and Load Balancing, our application can handle varying levels of traffic, ensuring responsiveness and availability even during peak loads.
* Flexibility: EC2 provides a wide range of instance types and configurations, allowing us to choose the most suitable environment for our ML model and application.
* Security: AWS offers robust security features, including Virtual Private Cloud (VPC), security groups, and encryption options, ensuring that our application and data remain secure.
* Cost-Effective: Pay-as-you-go pricing model of AWS EC2 ensures cost-effectiveness by allowing us to pay only for the resources consumed, without any upfront costs.
* Reliability: AWS guarantees high availability and reliability with its global infrastructure, ensuring that our application remains accessible and operational at all times.

# Summary

The application is designed to operate on Amazon Web Services (AWS), leveraging its cloud infrastructure to handle the development, training, and testing of machine learning models. The application supports two types of models:

Convolutional Neural Network (CNN) Viscosity Model: This model is used to predict the viscosity of substances based on input data.

Artificial Neural Network (ANN) Twin Screw Image Classifiers: These are three distinct ANN models designed to classify images related to twin screw processes.

Users can upload their datasets and models to an Amazon S3 bucket, ensuring secure and scalable storage. This is crucial for training the models as it allows for easy access to large amounts of data.

They can also download the datasets and trained models from the S3 bucket for further analysis or for use in other applications.

References

1. [Indirect monitoring of ultralow dose API content in continuous wet granulation and tableting by machine vision - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S0378517321008140)
2. [Integrated twin-screw wet granulation, continuous vibrational fluid drying and milling: A fully continuous powder to granule line - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S037851732031111X)
3. What is Git & why should you use it? (2021.november) <https://www.nobledesktop.com/blog/what-is-git-and-why-should-you-use-it>
4. Why did we build Visual Studio Code? (2021.november) <https://code.visualstudio.com/docs/editor/whyvscode>
5. [Processes | Free Full-Text | Classification of Droplets of Water-PVP Solutions with Different Viscosity Values Using Artificial Neural Networks (mdpi.com)](https://www.mdpi.com/2227-9717/10/9/1780)
6. [Welcome to AWS Documentation (amazon.com)](https://docs.aws.amazon.com/)
7. Brooks, R.; Dinsdale, A.; Quested, P. The measurement of viscosity of alloys—A review of methods, data and models. Meas. Sci. Technol. 2005, 16, 354. [The measurement of viscosity of alloys—a review of methods, data and models - IOPscience](https://iopscience.iop.org/article/10.1088/0957-0233/16/2/005)
8. Zhu, H.; Dexter, R.; Fox, R.; Reichard, D.; Brazee, R.; Ozkan, H. Effects of polymer composition and viscosity on droplet size of recirculated spray solutions. J. Agric. Eng. Res. 1997, 67, 35–45. [Effects of Polymer Composition and Viscosity on Droplet Size of Recirculated Spray Solutions - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0021863497901517?via%3Dihub)
9. Gotaas, C.; Havelka, P.; Jakobsen, H.A.; Svendsen, H.F.; Hase, M.; Roth, N.; Weigand, B. Effect of viscosity on droplet-droplet collision outcome: Experimental study and numerical comparison. Phys. Fluids 2007, 19, 102106. [Effect of viscosity on droplet-droplet collision outcome: Experimental study and numerical comparison | Physics of Fluids | AIP Publishing](https://pubs.aip.org/aip/pof/article-abstract/19/10/102106/938495/Effect-of-viscosity-on-droplet-droplet-collision?redirectedFrom=fulltext)
10. Wang, Z.; Liu, H.; Zhang, Z.; Sun, B.; Zhang, J.; Lou, W. Research on the effects of liquid viscosity on droplet size in vertical gas–liquid annular flows. Chem. Eng. Sci. 2020, 220, 115621. [Research on the effects of liquid viscosity on droplet size in vertical gas–liquid annular flows - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0009250920301536?via%3Dihub)
11. Kheloufi, N.; Lounis, M. An Optical Technique for Newtonian Fluid Viscosity Measurement Using Multiparameter Analysis. Appl. Rheol. 2014, 24, 15–22. 11.
12. Santhosh, K.; Shenoy, V. Analysis of liquid viscosity by image processing techniques. Indian J. Sci. Technol. 2016, 9, 98693. [Analysis of Liquid Viscosity by Image Processing Techniques (indjst.org)](https://indjst.org/articles/analysis-of-liquid-viscosity-by-image-processing-techniques)
13. Amazon Web Services Documentation [Welcome to AWS Documentation (amazon.com)](https://docs.aws.amazon.com/)