

MS ESDS Capstone Essay

BUILDING A SCALABLE ML OPS CICD PIPELINE

Name: Vatsalya Anand

Course: MS Engineering Science (Data Science)

Email:vatsalya@buffalo.edu

UB Person Number: 50495187

1. Introduction

This project is an end-to-end Machine Learning Operations (MLOps) initiative designed to incorporate Continuous Ingestion and Continuous deployment (CI/CD) Pipelines which can be scalable and efficient. The process highlights modular coding was used for all the processes like data ingestion, transformation, and model trainer. After successful creation of the machine learning model, it was seamlessly integrated with Flask, HTML and CSS to have a web front end where the user could interact with it. Docker was utilised to create an image and container and with the help of GitHub Action it was pushed to the Amazon Elastic Container Registry (Amazon ECR). And finally, was deployed on the Elastic Compute Cloud (EC2) instance. This model gets automatically updated whenever any change is made in the main branch of GitHub making it fully autonomous and up to date using CI/CD.

2. Background

Nowadays more and more companies are pushing towards cloud solutions. Making models in Jupyter Notebook or Google Colab with Python is just scratching the surface on much wider application. While they are great tool for analysis of data and making insights out it, deploying it in a much scalable and production level is challenging. When trying to use the model in real world application we need to think in a different approach. CI/CD help automate the process of making changes in the code and deploying it right away with as little of friction as possible. This head reduces the pressure on teams as there is no certain release day when changes go live. Even a small change gets reflected as long as there is no failed test or error in the source file. These CI/CD approaches also reduces a significant cost of the project due to less human intervention. Most well known examples of companies that are using CI/CD pipelines are Netflix and Facebook. Netflix uses an inhouse platform Spinnaker which manages their pipelines. Using this they are able to deploy changes in matter of minutes to the actual customers rather than waiting for days.

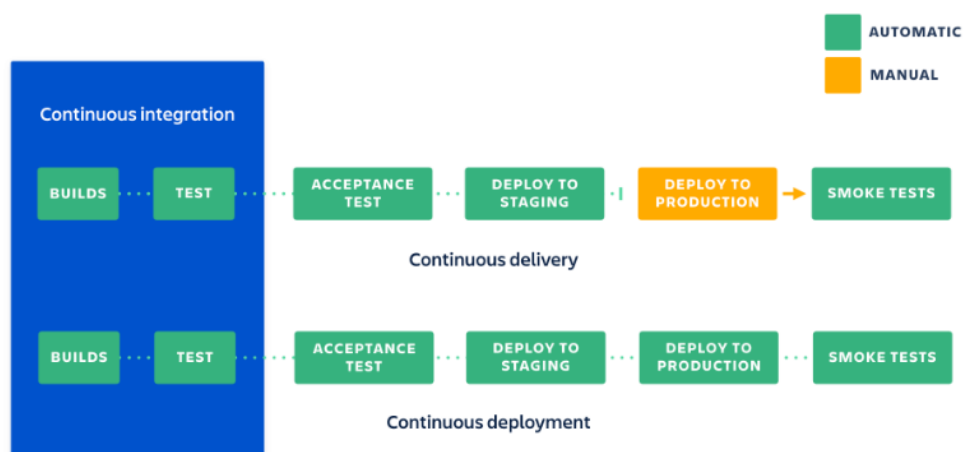


Figure 1: Relation between different stages of CI/CD (Atlassian)

The programming language that was used in this project was Python, Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS). Python is a high level interpreted programming language best known for readability and versatility. It is widely used for Data Science, Machine Learning and Software Development. HTML is standard language for making webpages. Can be used to make structures on the web page like headings,

paragraph, links, multimedia etc. CSS is stylesheet language used to describe the HTML. It lets us define the look and feel of HTML by controlling the layout, fonts, colours etc.

When building a project usually several libraries and dependencies are used. Some of the major ones are Pandas, NumPy, Plotly, Scikit-learn and Flask. Pandas is a library used for data manipulation and analysis. It allows us data structure and operations manipulation of numeric tables. NumPy in python which helps us manage data in multi-dimensional arrays and matrices. It also has a large number of mathematical functions that are used for operations on the arrays. Plotly is a graphing library used for making publication level graphs like scatter plot, histogram, box plots, heat maps to name a few. Scikit-learn is the go-to library for machine learning in Python. It provides simple and efficient tools for prediction of data. It has the built in function for different ml models like regression, classification, clustering etc as well as preprocessing. Flask is a micro web framework written in Python. It is straightforward approach of building web applications either small or big using Python.

The ML models that were used are Linear Regression, K Neighbours Regressor, Decision Tree Regressor, Gradient Boosting Regressor, XG Boost Regressor, Cat Boost Regressor, Ada Boost Regressor, Random Forest Regressor. Linear Regression is one the most simplest algorithm. It makes a relationship between independent and dependent variables. Due to its simple nature its is light on computational devices and easy to interpret. K Neighbours basically averages the values of the nearest K neighbours around it. We can change the value of K based on the problem and obtain different result of the same dataset. But it could be computationally expensive for larger dataset. Decision Tree forms model in a tree structure manner. It basically divides dataset in smaller chunks and then predicts the value by simple decision rules. Gradient Boosting is a boosting algorithm that builds trees one by one. Each of the tree in front correct the mistakes of the previous one. Usually known for high accuracy and able to handle much complex relationships. XG Boost or Extreme Gradient Boosting is even more optimised version of Gradient Boosting. Its well known for its speed and performance and mostly used for competitions. It also uses much more regularized model pattern to mitigate overfitting. Cat Boost or Categorical Boosting is very similar to XG Boost and is a gradient boosting algorithm. Its main use it to handle categorical type of data even without much of preprocessing. Like XG Boost its also known for speed and performance. Ada Boost or Adaptive Boosting is an iterative algorithm. It adjusts weights of falsely classified instances so that next classifiers focus on other instances. It unites weak performers to form a strong performer. Random Forest algorithm builds multiple decision trees while training then combines them to get the best accuracy. It is usually very accurate and is mostly immune to the overfitting while also being able to handle larger datasets.

While deploying on a cloud instance it may happen that these dependencies is not present which may create an issue. Here Docker comes in play. It is an open source platform that helps make an image of the project and containerise it so that it could be deployed on the client's server/cloud without any dependencies or version control issues. It revolutionised the software development as now none can claim that a particular project just runs on their system. Its efficient portable and scalable. Images are templates used to create containers. Containers are the runnable instances of image and has everything that is required to run a project anywhere from code to libraries to system tools.

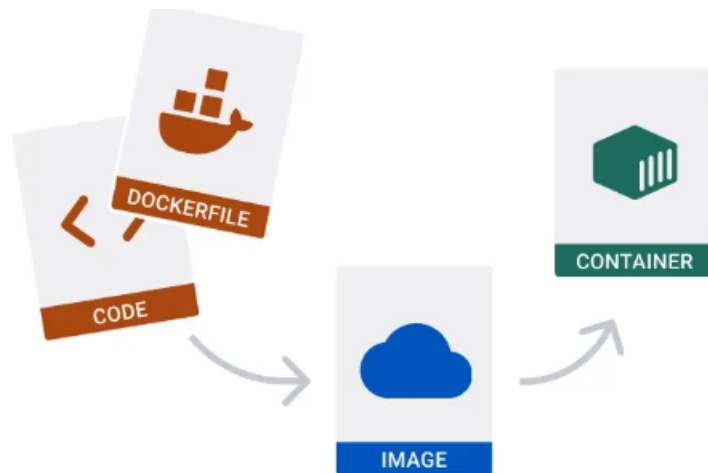


Figure 2: Using Docker (Docker)

GitHub Actions is being used to deploy the model on the cloud using the CI/CD. It makes it easy to automate the workflow. If there is any change in the repository it automatically does all the job and deploys the latest version of the cloud. In Continuous Integration it goes through all the steps like checking the code, running the code etc. While in Continuous Delivery it does installing of utils file, logging into the Amazon ECR, building and pushing the image etc. Finally in the Continuous Deployment it pulls the latest images, runs the docker image to server and cleans the previous images and containers.

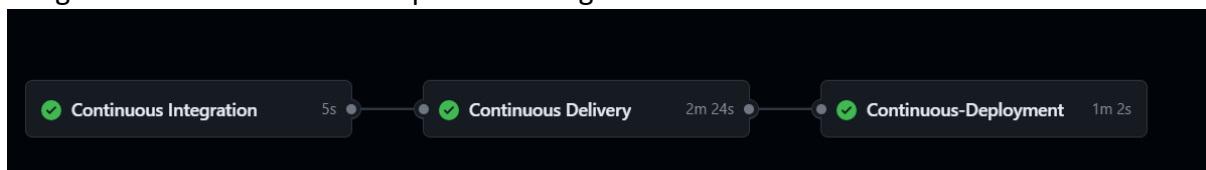


Figure 3: CI/CD deployment on GitHub Actions (GitHub)

Nowadays one of the most popular cloud providers is Amazon with its AWS. AWS is a on demand cloud computing platform with its servers across the globe for quick and easy access. It is a lucrative product for companies as it works on “Pay-as-you-go” model, only charging for the number of resources being used. Many of the big companies from different sectors are now using AWS ranging from software to travel to manufacturing. Some of them are Airbnb, Canva, Formula 1, Sony, Starbucks, Toyota, Verizon to name a few.

The ECR is container registry where the Docker image can be hosted. It’s known for its security, scalability, and reliability. It even supports private repositories with permission-based approach of accessing it. It has a image scanning that could scan vulnerabilities keeping the actual image and hence the final product safe. Cross Regional support is there so an image can be hosted on other servers but could be accessed from anywhere in the world.



Figure 4: Working of Amazon ECR (AWS)

EC2 is one of the AWSs' flagship web services. It was launched in 2006 and is one of the key building block of public cloud infrastructure. It is a resizable environment on the cloud where we can deploy our final app or websites. It helps eliminate the need to buy a powerful hardware upfront. As its scalable it reduces the need to forecast traffic in advance. Sony is one of the biggest spender of EC2. With an estimated monthly spend of \$11M. And Adobe and Facebook following behind it with \$7.5M and \$5.6M respectively.

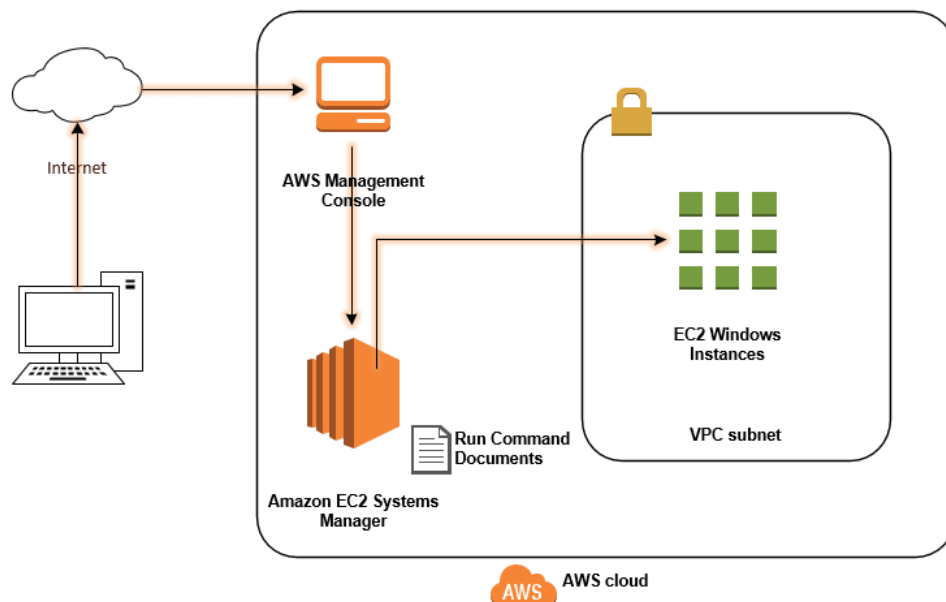


Figure 5: Operation of an EC2 instance (AWS)

AWS Identity and Access Management (IAM) is used for to give permission to user or group within the AWS platform. It is granular and can be done on individual level. It has options for multi factor authentication using software or hardware-based authentications. Its an essential service to safeguard the overall AWS platform experience. As most of the times certain group of individuals shouldn't have access to certain service which could create a disruption of the whole product. Only people qualified and experts of a service should be given the access.



Figure 6: Analysis of AWS IAM (AWS)

3. Methods

When starting this project first thing was making a project structure. A GitHub repository was set as this was crucial later on when trying to deploy the project. While on the system side the environment was made. Python 3.10 was the version used for this project. Once the environment was set with necessary files. The folders with all the files were committed to the GitHub making the connection complete. Then a setup.py file was made which outlines the basic setup for the whole as well as requirements.txt was formed to make sure it contains all the libraries like NumPy, Pandas, Plotly, scikit-learn etc that will be needed during the project making.

After this components directory was made which contains the components of modular coding like data ingestion, data transformation and model trainer. Then the pipeline directory was made which consists of the train and predict pipelines. Apart from these exception, logger and utils files were made for the smooth operation of the project.

Now we took the dataset. The dataset is simple student performance metrics. A fairly easy dataset was used to make sure all the processes work as intended then we could always scale up the dataset by taking more complex problem statements.

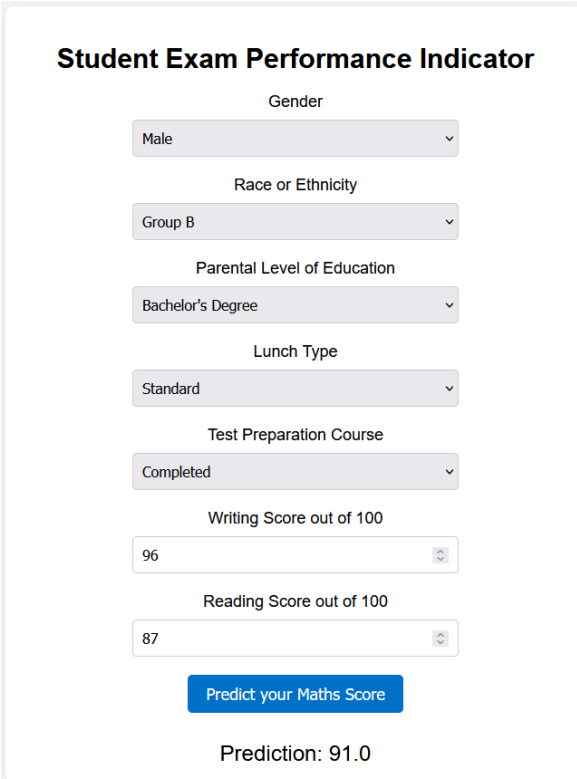
The data ingestion file reads the 'data.csv' files that contains all the details about the performance metrics of the students. It then splits it into test/train and saves it as csv file. Next the data transformation takes the data that data ingestion file made and does the Exploratory Data Analysis (EDA) processes on it to make sure the data is well processed. It does EDA processes like make numerical columns and categorical columns as well as make numerical and categorical pipelines. It finally saves the data as a pre-processed pickle file. Now in the model trainer file all the ML models were defined that was used like Random Forest Regressor, Decision Tree Regressor, Gradient Boosting Regressor, Linear Regression, K Neighbours Regressor, XG Boost Regressor, Cat Boost Regressor, Ada Boost Regressor. We check the accuracy as well as R2 value to find out the feasibility of each of the model. A threshold of 60% was setup so that if none of the models have more than that we display that none of the model performed well for giving good results. Then all the models were compared, and the best model with highest accuracy and R2 value was saved as model pickle file.

Once the initial model was made hyperparameter tuning was done on each model to get even better evaluation metrics. Like doing cross validation to ensure the reliability of model. Using Randomized Search if the parameter is in a large space. Parallelizing searches

to speed up the tuning process. Using early stopping techniques for XG or Cat boost to support it. Updating learning rates to converge better. Using different loss functions like liner, square or exponential while updating the weights. Changing the max depth of each tree. These were some of the hyperparameter tuning done.

The prediction pipeline was setup. This takes in the model and preprocessor pickle file that were obtained earlier from model trainer and data transformation and loads them in ready for predictions. Next up it takes in user input like gender, race, different scores and returns predicted value. Pipeline is also designed to handle errors and ensure the robustness.

Once the backend of the coding was done it was time to make a user-friendly front end which could be used by a normal user even if they don't possess a coding knowledge. To do this an app was made with the help of Flask and HTML. Giving user a platform to interact and get prediction values. In addition to core functionalities a bit of attention was also given to the aesthetics of the app to improve the accessibility. Like giving drop down for the categorical inputs like Gender, Race etc. and giving numeric input field for numerical values.



Student Exam Performance Indicator

Gender
Male

Race or Ethnicity
Group B

Parental Level of Education
Bachelor's Degree

Lunch Type
Standard

Test Preparation Course
Completed

Writing Score out of 100
96

Reading Score out of 100
87

Predict your Maths Score

Prediction: 91.0

Figure 7: Front End of the App

Now that the front end is ready backed up with the ML model. Next in the process is deploying it. A docker file is setup which will make the docker image and containerize the whole project making a seamless workflow. Yaml file contains the workflow that define what all steps need to be taken during the CI/CD deployment.

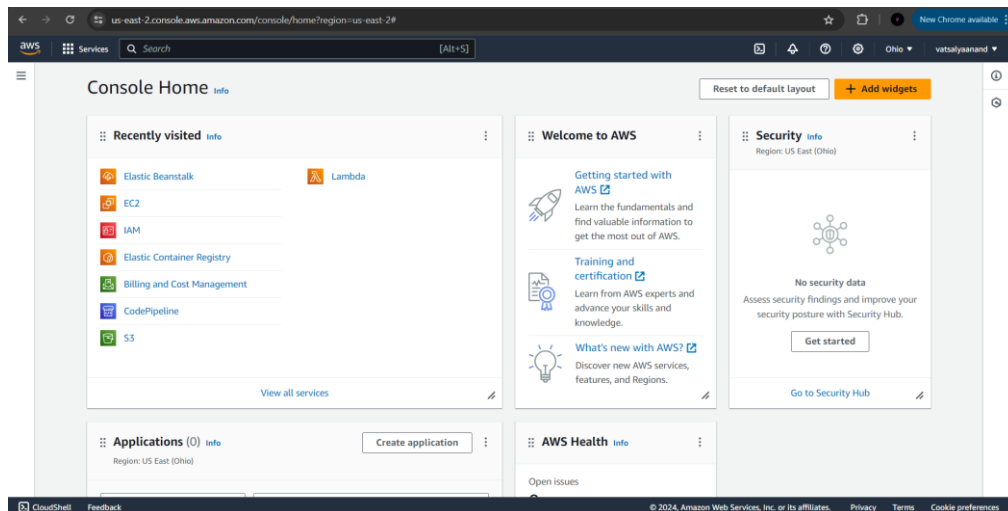


Figure 8: AWS Console Home (AWS)

We can now start using the AWS cloud.

Using the IAM; users or groups are defined in the AWS console to gain access for certain functionalities on the AWS cloud. In our case the access is given for the EC2 and ECR. Some of the permissions are EC2 start stop or terminate instance, create tags etc. For ECR get authorisation token, get download URL from layer, initiate and upload layer.

Then in ECR a new repository is created where we will deploy our docker image. This repository is secure and scalable as well as allows it to be easily accessible and manageable.

Next EC2 instance server is set up. Using an Ubuntu server here with 64bit and taking medium tier machine. Now using EC2 instance command prompt we upgrade the server and download docker and configure the ECR repository.

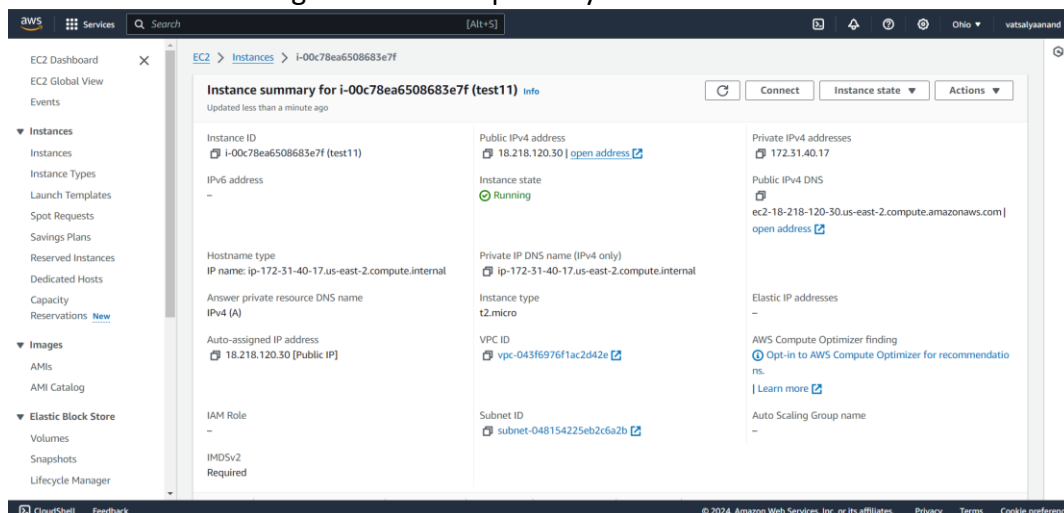


Figure 9: An EC2 Instance (AWS)

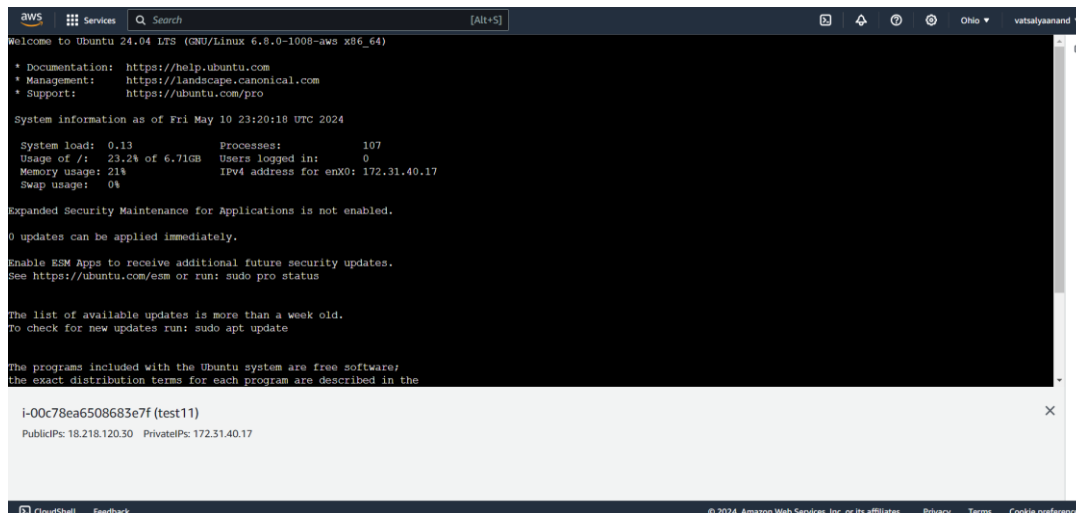


Figure 10: An EC2 Command Prompt (AWS)

Now the GitHub Actions is setup. First, we use the commands within the console of the EC2 to create a self-hosted Runner. Then we add the secret keys like `AWS_ACCESS_KEY_ID`, `AWS_REGION` and others to the GitHub which will help GitHub to effectively navigate the created EC2 instance.

Now the app runner will automatically do the continuous integration, delivery, and deployment of the project whenever any changes are committed to the main branch of the GitHub repository completing our CI/CD project.

Finally, our app is deployed on the cloud (Figure 6).

Some of the best practices that we can do are code quality, adhering to the best practices for coding like using PEP 8 guidelines for our python code. Version control, using the Git for version control so that we could track the changes if anything is required to be reversed in the future in a case of mishap. Ensuring automated testing in the CI/CD pipeline to catch errors and stop potential breakage of the flow. Documentation of the code so that if the project gets big, we can make sure that everything is streamlined and understandable even for others.

Word Count: 2580

References

- [1] "Continuous Integration vs. Continuous Delivery vs. Continuous Deployment," *Atlassian*, [Online]. Available: [Accessed: March 3, 2024].
- [2] "Python 3.10.4 Documentation," *Python.org*, [Online]. Available: <https://docs.python.org/3.10/>. [Accessed: March 3, 2024].
- [3] "About pandas," *pandas*, [Online]. Available: <https://pandas.pydata.org/about/>. [Accessed: March 3, 2024].
- [4] "Pandas (software)," *Wikipedia*, [Online]. Available: [https://en.m.wikipedia.org/wiki/Pandas_\(software\)](https://en.m.wikipedia.org/wiki/Pandas_(software)). [Accessed: March 3, 2024].
- [5] "NumPy," *Wikipedia*, [Online]. Available: <https://en.m.wikipedia.org/wiki/NumPy>. [Accessed: March 7, 2024].
- [6] "Python Graphing Library | plotly," *plotly*, [Online]. Available: <https://plotly.com/python/>. [Accessed: March 9, 2024].

- [7] "scikit-learn: machine learning in Python — scikit-learn 1.0 documentation," *scikit-learn*, [Online]. Available: <https://scikit-learn.org/stable/>. [Accessed: March 10, 2024].
- [8] "Flask (3.0.x) - Flask documentation," *Flask (A Python Microframework)*, [Online]. Available: <https://flask.palletsprojects.com/en/3.0.x/>. [Accessed: : March 10, 2024].
- [9] "HTML Standard - Global attributes," *W3Schools*, [Online]. Available: https://www.w3schools.com/tags/ref_standardattributes.asp. [Accessed: March 10, 2024].
- [10] "Flask Project Structure and Setup," *Real Python*, [Online]. Available: <https://realpython.com/flask-project/>. [Accessed: March 10, 2024].
- [11] "Run a container," *Docker Documentation*, [Online]. Available: <https://docs.docker.com/guides/walkthroughs/run-a-container/>. [Accessed: March 15, 2024].
- [12] "GitHub Actions Documentation," *GitHub Docs*, [Online]. Available: <https://docs.github.com/en/actions>. [Accessed: March 17, 2024].
- [13] "GitHub Actions: Automate, customize, and execute your software development workflows right in your repository," *GitHub*, [Online]. Available: <https://github.com/features/actions>. [Accessed: March 20, 2024].
- [14] "Amazon ECR | Container Registry | Amazon Web Services," *Amazon Web Services, Inc.*, [Online]. Available: <https://aws.amazon.com/ecr/>. [Accessed: April 2, 2024].
- [15] "Amazon EC2 – Cloud Server & Hosting | AWS," *Amazon Web Services, Inc.*, [Online]. Available: <https://aws.amazon.com/pm/ec2/>. [Accessed: April 3, 2024].
- [16] "AWS EC2 Instance - DEV Community," *DEV*, [Online]. Available: <https://dev.to/genialkartik/aws-ec2-instance-57gb>. [Accessed: April 23, 2024].
- [17] "How to deploy container image from ECR to EC2 instances using AWS Code Deploy agent?," *SupportSages*, [Online]. Available: <https://www.supportsages.com/how-to-deploy-container-image-from-ecr-to-ec2-instances-using-aws-code-deploy-agent/>. [Accessed: April 27, 2024].
- [18] "CI/CD - GitHub Resources," *GitHub Resources*, [Online]. Available: <https://resources.github.com/ci-cd/>. [Accessed: May 9, 2024].