**Documentation of the Project**

**Project 6 : Cancer Prediction.**

**Folder Name & Path : MLOps-Beginner to Advanced MLOps on GCP-CICD\_ Kubernetes Jenkins/5\_Project**

**Highlights of the Project :**

1. **Use of MLFlow with DagsHub for Online Experimentation (CometML)**
2. **Making Kubernetes Cluster on Local machine using MiniKube for Deployment**
3. **Training with Kubeflow Pipelines**
4. **Use of DockerHub, in previous project we have used Google Container Registry, now we use DockerHub.**

Targeted audience are Hospitals. First usecase is Patient Monitoring, Resource Allocation(Less staff of Hospitals)

**WorkFlow of the project:**

Here **Data Ingestion** part is not explained again, we can implement the same steps as mentioned in the previous projects.

1. **Project setup:** We define project and folder structure, custom Exception, logging, create virtual environment.
2. **Jupyter Notebook Testing:** We will do notebook testing on EDA, Data processing , and all required steps.
3. **Data Preprocessing :**
4. **Model Training :**
5. **Data and Code Versioning:** It is used to track the data at various steps. **DVC**, can be used when we have huge data. **Github**, can be used when we have less data. Here we are using **Github**.
6. **Experiment Tracking :** We will use **MLFlow** and **DagsHub**, Experiment tracking online.
7. **User App With Prediction Pipeline:** To build a front end app for Users. We will Use Flask, HTML.
8. **Kubeflow Setup** : We will setup **MiniKube**, **Kubectl**, **Kubeflow** pipelines on Local PC.
9. **Dockerize the Project** : We will **dockerize** the project using Docker and push it to **DockerHub** .
10. **Pipeline Using KubeFLow :** Once we build the pipeline, it will generate **Yaml** file, then We will upload that Yaml file in the **KubeFlow** dashboard(next section 11). Basically the Yaml file will have the code logic generated in the form of Yaml file, we wil write code in python but it will convert it to Yaml , we have upload that Yaml file in Kubeflow dashboard and run the Pipeline.

1. **Uploading and Running the PIpeline :**
2. **Done.**

Dockerized project --> Pushed to Docker Hub.--> Now image is in Docker Hub,I.e all the project is in form of Docker Image. --> Now we will build training pipeline for this Image , not for the PC code. --> In Pipeline making step, we build the pipeline, this will generate Yaml file --> Now upload this Yaml file(Training file) in Kubeflow dashboard. --> Now run this pipeline . Once we start running the pipeline, it will trigger the image and start processing the code inside the image. NOw the existing image will be replaced by New image with Processed code. Advantage of this image is , we can deploy this image anywhere we wish, in Google Keburnetes Engine, Virtual Machine, AWS, Google Cloud Run or anywhere. Trained image is important.

**Now Starts with Project Code Setup Implementation.**

**1. Project Setup Implementation :**

Create a folder for the project on Laptop. From this folder path type CMD and enter, then run code . , this will take you to VS Code. Now Go to VS Code, open terminal

* Create virtual environment
* Python -m venv venv
* Another way of Creating virtual environment

Conda create -p venv python==3.10.0 -y

* Activate virtual environment
* Venv\Scripts\activate
* Conda activate venv/
* Create a setup.py file.
* Now start with project structure. Create below files/folders. To make any folder as a package, create **\_\_init\_\_.py** file
* Create **requirements.txt**
* Create **setup.py**
* Create **src** folder. To make this folder as a package create **\_\_init\_\_.py** file.
* Create **Config** folder. To store configurations.
* Create **Notebook** folder
* **Static** folder
* **Utils** folder
* **Templates** folder
* **Artifacts** folder. To store the CSV files, processed files, model.
* **Pipeline** folder

Now src folder should be treated as a package, that means you define some function in one file and import it in another file. For that run “**pip install -e .**” in terminal.

**2. Jupyter Notebook Testing :**

In VS Code, Extensions install Jupyter Extension. Now Create **notebook.ipynb** file in **Notebook** folder. Create a new cell and just run, select the environment. Again run empty cell for installing the **Ipykernal.**

**3. Data Preprocessing :**

In **Src** folder create a **data\_processing.py** file. Add the required code in this file. Define **init** constructer, **load\_data**, **preprocess\_data**, **feature\_selection,**  **split\_and\_scale\_data, save\_data\_and\_scaler, run** methods.

Data processing with Feature store is done.

4. **Model Training :**

In **src** folder crate a **model\_training.py** file. Create a class **ModelTraining** in the file. Initialise the parameters. **Load\_data, train\_model, evaluate\_model.**

5. **Experiment Tracking**

We will track the experiment using **MLflow** and **DagsHub, this becomes as CometML.** In model\_training.py file , import MLflow, mlflow.sklearn

Run **Python src/model\_training.py** . Now in the terminal only run **mlflow ui** , this will provide an URL for mlflow dashboard. This mlflow deployment on local..

Now we will deploy mlflow using Dagshub. Go to browser, search for Dagshub and login. Logged into Dagshub using Github credentials. **rajasekharvardhi@gmail.com**,**RaJuArJuN$89**. In dagshub dashboard --> **Create** --> **New repository** --> **Blank Repository** -->**Provide a name for Repo** (**mlops\_project\_5**) --> **Create Repository**.

Now push the code to Github. Createa a new Repository : MLOps\_Udemy\_Project\_5 <https://github.com/RajaSekhar899/MLOps_Udemy_Project_5>. Now run below code in VS Code terminal for pushing the code from VS Code to Github repo.

Now create a **.gitignore** file, add all the folders which are not required, add **venv, logs, MLOps\_Project\_5.egg-info.**

Now initialize the git repository. Run below commande:

- **git init**

- Now login to github and create a new repository.

Using github account credential , [rajasekharvardhi@gmail.com,](mailto:rajasekharvardhi@gmail.com,) RaJuArJuN$89.

Created **MLOps\_Udemy\_Project\_5** Repository.

- **git branch -M main**

- **git remote add origin [https://github.com/RajaSekhar899/MLOps\_Udemy\_Project\_5.git](https://github.com/RajaSekhar899/MLOps_Udemy_Project_3.git)**

**- git add .**

**- git commit -m “pushing code to git”**

**- git push origin main**

We have pushed our code to Github.

Now go to Dagshub, in the project created, click on connect Repository --> select Github --> follow steps to Authorize the account. Once authorized , in DagsHub dashboard, Remote --> select Experiments --> Copy the url.

Now in VS Code Terminal --> run below commands

**set MLFLOW\_TRACKING\_URI=https://dagshub.com/RajaSekhar899/MLOps\_Udemy\_Project\_5.mlflow**

**set MLFLOW\_TRACKING\_USERNAME=RajaSekhar899**

For password Dagshub settings --> tokens --> generate new token --> provide name --> generate --> copy the token

**set MLFLOW\_TRACKING\_PASSWORD=26e2651da7f6578c81694b0ccbd3a5a2c316546e**

NOw run the model\_training file, **python src/model\_training.py.** This will provide an URL for viewing the experiment in Dagshub.

With this we are done with tracking the experiment in Dagshub.

1. **User App Building:**

Now let’s start with User App building. Now create folders **static, templates** in Project directory. Now create **index.html** file in **templates** folder and **style.css** file in **static** folder. Add the required code in this files. We can gett index.html and styles.css code from chatgpt.

Create an **Application.py** file in project directory. Add the required code. Then run **python application.py** , this will provide an URL for web app.

1. **Kubeflow Setup** :  **:**

This is the important step in the project. Now let us start with local saetup of Kubeflow pipeline.

- Install Docker desktop if not already done. Make sure Docker desktop is running in the background. There are 2 kubernetes, **Docker kubernetes** and **minikube.** Here we will be using Minikube. Make sure docker kubernetes is off in docker dashboard, if not settings --> kubernetes --> turn off kubernetes.

**Step 1 :** Install minikube (local kubernetes)

Now let us install minikube. Go to browser, search for minikube install windows, <https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download>, download <https://storage.googleapis.com/minikube/releases/latest/minikube-installer.exe>, install this.

- Now in laptop, search for powershell

--> run **minikube version**, for checking version.

- In powershell run

--> **minikube start,,,**

--> **minikube status** now check status  **.** Now to verify you can check in docker dashboard.

**Step 2** : Install Kubectl

Now let us install kubectl. Go to browser, search for kubectl install <https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>, let us install this through chocolaty.

Now before installing kubectl, we need to install chocolatey <https://chocolatey.org/install>,, follow these link [https://docs.chocolatey.org/en-us/choco/setup/#more-install-options](https://docs.chocolatey.org/en-us/choco/setup/" \l "more-install-options),,

- In powershel run **powershell.exe,** this will install chocolatey

**-** To verify run **choco.**

- Once it is done Now, follow these commands to install Kubectl,

- **choco install kubernetes-cli**

**- kubectl version --client**

**- kubectl version,** for verifying

**Step 3**: Now setup Kubeflow

Go to browser , search for kubeflow local deployment <https://www.kubeflow.org/docs/components/pipelines/legacy-v1/installation/localcluster-deployment/>

- In powershell run

--> **kubectl cluster-info**

**-** Now deploy kubeflow pipelines

--> **set PIPELINE\_VERSION=2.14.3**

**--> kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/cluster-scoped-resources?ref=$PIPELINE\_VERSION"**

**--> kubectl wait --for condition=established --timeout=60s crd/applications.app.k8s.io**

**--> kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/env/platform-agnostic?ref=$PIPELINE\_VERSION"**

**-->** In command promptrun **kubectl get pod -A,** let’s wait for the images to be created

--> **kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80**

**-->** Now in browser search for **<http://localhost:8080/,>**  this is where our kubeflow pipeline is running.

--> Make sure powershell is running background.

With this Kubeflow setup is done.

1. **Dockerize the Project** :

Create a **Dockerfile** in project directory. Now in VS Code terminal command prompt

- **docker build -t my-mlops-project-5-app . ,** to create an image

- **docker run my-mlops-project-5-app ,** This will provide an URL for the container, but it’s not working, but its fine because we will be pushing our image to Docker hub.

- **docker login**

**-** Also login Docker hub in browser

**-** Credentials :- [rajasekharvardhi@gmail.com,](mailto:rajasekharvardhi@gmail.com,) RaJuArJuN$89

**- docker tag my-mlops-project-5-app rajasekhar8995/my-mlops-project-5-app:latest -** To tag the image I.e renaming **my-mlops-project-5-app** to **rajasekhar8995/my-mlops-project-5-app:latest**,  **rajasekhar8995** is Docker hub user name, **my-mlops-project-5-app** is Image name

**- docker push rajasekhar8995/my-mlops-project-5-app:latest -** To push the image

- Now go to docker hub and look for the pushed image in Repositories.

Now create a file **mlops\_pipeline.py** file in **Kubeflow\_pipeline** folder. Add the required code for creating pipelines. Now go to VS code terminal, run

-  **python kubeflow\_pipeline/mlops\_pipeline.py,**

**-** Once this ran , this will create **mlops\_pipeline.yaml** file in project directory

Now go to Kubeflow dashboard which is at URL **<http://localhost:8080/> ,** --> **Pipelines** --> **Upload Pipeline** --> select **Create a new pipeline** , provide a name for pipeline , choose upload a file , choose file , upload the yaml file that is been created above --> **Create.**

Now we have created the pipeline. Now we need to run the pipeline, In the created pipeline, click on **Create run.** --> Provide a name to run name --> (if we want to create an experiment click on Experiments --> click on create experiment --> provide a name to experiment --> next. ) --> choose Experiment that created just now --> choose run time as required (one off)--> **start.**

With this we have done with Kubeflow pipelines.

1. **Done**