Project creation

Page No.:
Date:

Demo pipeline

① create a folder (Any name)
Select and right click on folder and click "open with VS code"

or

→ open the folder copy the path of it
→ open command prompt and enter "cd folder path". ↳
→ enter "code ." it will open VS code

or

→ open VS code → click on file → open folder

② Whenever we are creating a project the first thing we required is virtual environment

→ click on Terminal at the top → click "New Terminal"
→ Expand dropdown in Terminal and click on "command prompt".
→ use anaconda base enviroment.

Note: ['cls" is the command for clearing the content of Terminal]

Note2: If we are having anaconda in our system we need to type "conda" in Terminal

→ we will be able to find packages, commands

[If we are getting error "conda is not recognized as internal or external command", then we have to follow the below steps to solve the error:

① click on view → command palette

↓

Search for → python : Select interpretor.

↓

Select the base interpretor path (python 3.10.9 ('base'))

base environment

→ How to create virtual environment inside the conda.

Commands:

① conda env list

↳ It will fetch all the environments in our system.

② for creating New environment.

* conda create -p venv python=3.8 ↳

After creating environment we have to click "y"

-p creates environment in local directory

-n creates environment in default directory.

③ For activating the environment

* conda activate venv → It is throwing error because our environment is not present in default directory

→ Right click on "venv" and copy the relative path.

* conda activate "path of the environment".

↳ Here path doesn't consist of more than two words.

[Base environment is the by default environment which conda provides to us for creating a project but we are not going to use this base environment. If we are using this environment it will be messy. If we are going to create 100's of projects each and every requirement we will install in a base environment definitely it is not a good thing. For this what we have to do is to create a virtual environment with respect to the project]

→ After environment is ready, Now we have to Create a git hub repository

→ Once Github repository created Now we have to ~~create~~ connect Vscode with repository

① For to check whether our Vs code is pointing to any repository or not from Vs code

* git remote -v

② For initializing git repository

* git init

↳ initialize the git repository.

③ For to add the file into staging area

* git add . → For to add all files in staging area ~~of~~ ~~your~~ including venv

[If i don't want to add this venv into my github

~~In~~ In this case we will add venv into .gitignore file.

whatever the files we don't want to add we will add into gitignore file"

④ For undo the above command (For to reset whatever the tings we have added in staging area).

* git reset

→ After resetting add venv to the .gitignore file and save it.

→ Again add and check the following ~~or~~ command

* git add . → this time files inside venv will not add to staging area.

⑤ commit the files to git by using following command

* git commit -m "commit name"

Ex:- git commit -m "first commit"

⑥ For to check our git is pointing to which branch.
* git branch

⑦ For to change the ~~branch~~ git branch
* git branch -M main.

⑧ For adding the origin
*git remote add origin https://github.com/dineshbabuvaddineni/project-pipeline.git

⑨ For to push our code into the repository.
* git push -u origin main.

core ml pipeline.
① Data ingestion → DE (data ingestion pipeline) (csv, json) (scrapping)
② EDA
③ pre processing (FE)
④ Model Building (ML algo) ⇒ supervised / unsupervised
⑤ Evaluate ⇒ confusion matrix (roc/auc, ~~f1~~ p, r)

↳ This core pipeline we can create (or perform) in a jupyter notebook.

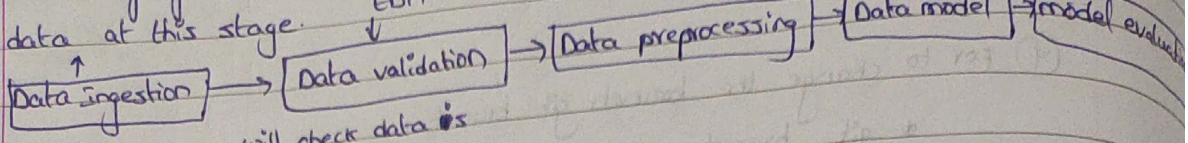
→ whenever we are trying to create a industry level project we can't use use jupyter notebook. Instead of this we will use .py file (python file) because here we have to write oops concept, there we have to do modular coding and this core ml pipe is connected with each other apart from this we also have different files, folders to complete infrastructure.

pipeline — collection of components.

Date:

Fig: core ML pipeline.

we are going to collect data at this stage.



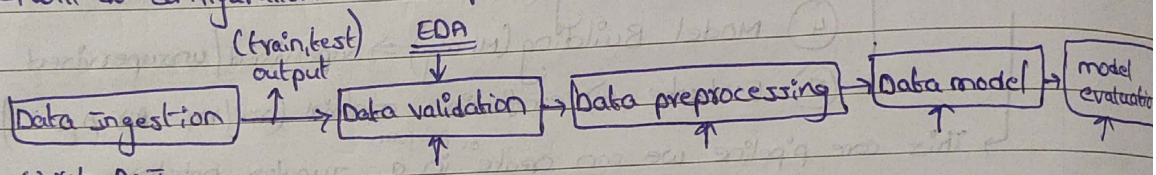
will check data is validate or not

(→ Analysis data doc, No. of rows, columns, Data types we are going to react

Real time pipeline of the machine learning

① configuration of the component
② Artifact :- output of the component

→ will do configuration each component.



Write D.I

we will do configuration

config File

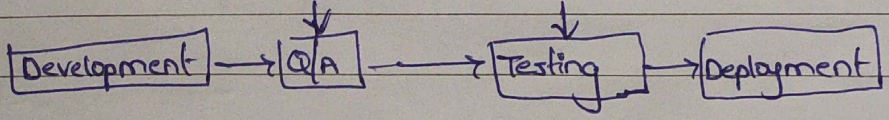
Train, Test

meta

→ whenever we are creating a pipeline logging and Exception handling is important.

Apart from these infrastructure wise

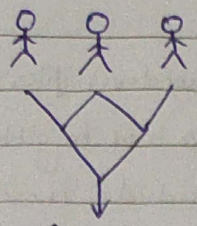
→ Git, Github, docker, package of the component, cloud



Development → Deployment.

IDE { Vscode, Pycharm, python }

Development → Deploy



Github
(Central repo)

AWS/Azure/Gcp

machine learning (mlops)

(Github)
CI CT CD → (AWS/Azure/Gcp)

(CI/CD)

machine learning development.

CI → CD

local → github Github → AWS/Azure/Gcp

① module packaging

(PYPI

Data ingestion
preprocessing
ML
} These modules we can package → we can upload → PYPI/conda

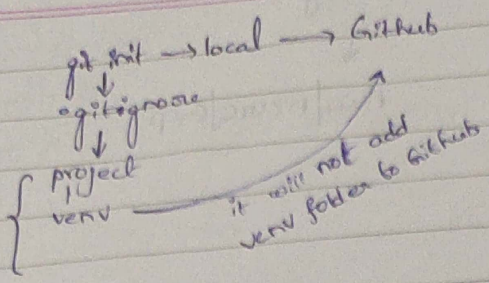
From here anyone can download and use in their system.

② Docker

- docker container
- " image
- " compose
- " hub

→ ~~what~~ we are going to create ~~project~~ a package

As a singularity we are getting our project and we can deliver our project to everyone.



→ [whatever files we don't want to add to github, we will include those file names in .gitignore file]

Requirements.txt

→ we have to write package names what we want to install

ex: pandas
numpy
flask

→ save the file.

for to check whether our requirements.txt file is there in ~~python~~ directory

* dir

→ for to run the requirements.txt file.

* pip install -r requirements.txt

If we ~~include~~ want to install the package in current environment for that we included -e.

pip install -m requirements.txt

project Demo pipeline.

Page No.:
Date:

- Demo pipeline
  - notebook
    - Test.ipynb (set the kernel venv)
  - venv
  - census_income
    - components
      - data_ingestion.py
      - data_preprocessing.py
      - data_validation.py
      - model_evaluation.py
      - model_trainer.py
      - __init__.py
    - Config
      - __init__.py
      - artifact.py
      - config.py
    - exception
      - __init__.py
      - exception.py
    - logging
      - __init__.py
      - logging.py
    - pipeline
      - __init__.py
      - training_pipeline.py
      - prediction_pipeline.py
    - utils
      - __init__.py
      - utils.py
  - __init.py
  - .gitignore
  - README.md
  - requirements.txt
  - setup.py