Module 3: AST-1

TITLE: Transitioning from Research to Production Environment

LEARNING OBJECTIVES:

At the end of the experiment, you will be able to transition from the research environment to the production environment. You will understand the concept of modularization and convert the model developed in Jupyter Notebook into different modules tailored to specific functionalities: Data Manager, Training, Pipeline, Predict, etc.

You will be able to understand and implement the following aspects:

1. VS code setup and understanding interface components
2. Creating virtual environment, installing requirement, running .py & .ipynb file
3. Understanding __name__ == "__main__" and building our own modules
4. Modularization: Converting a ML model from research environment format (Notebook) to product environment format (.py files)
   - A) Creating different modules specific to functionality: Data Manager, Training, Pipeline, Predict
   - B) Understanding folder structure
   - C) Separating configuration files
   - D) Concept of YAML file

INTRODUCTION:

What is a production code?

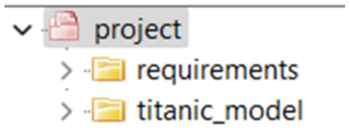
Production code is designed to be deployed to end users as opposed to research code, which is for experimentation, and building proof of concepts, and research code tends to be more short-term in nature.

The following are considerations with production code:

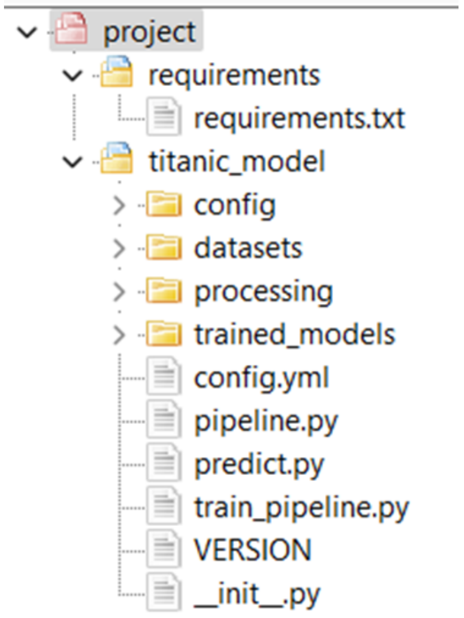
- Testability and maintainability: Dividing code into modules that are more extensible and easier to maintain & test.
- Separating configuration from code where possible, and ensuring that functionality is tested and documented.
- May need to refactor inefficient parts of the code base and finally, reproducibility.
- Ensuring version control with clear processes for tracking releases and release versions, requirements, a mark of which dependencies and which versions are used by the code.

- Ensure that code adheres to standards like PEP8 so that it's easy for others to read and work.
- Scalability and performance are also important areas to consider: The production code needs to be ready to be deployed to infrastructure that can be scaled.In modern web applications, this typically means containerization for vertical or horizontal scaling.

Understanding the folder structure in production environment:



```
project
  > requirements
  > titanic_model
```



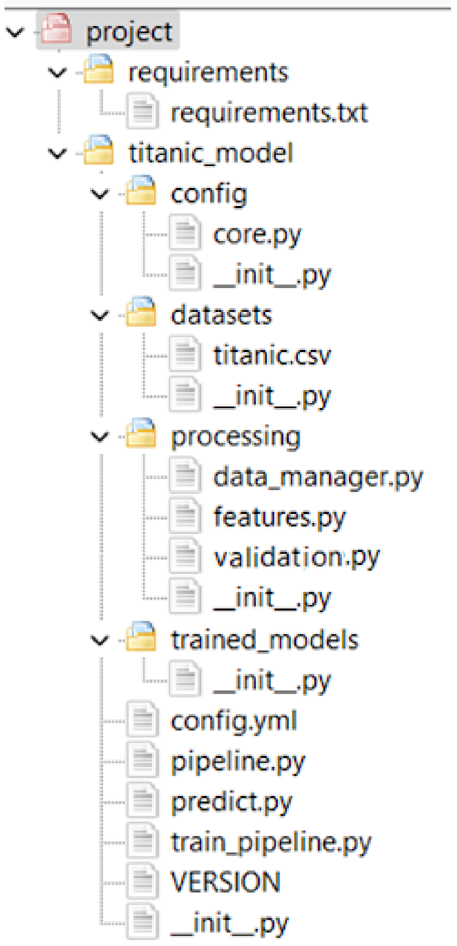
```
project
  requirements
    requirements.txt
  titanic_model
    > config
    > datasets
    > processing
    > trained_models
    config.yml
    pipeline.py
    predict.py
    train_pipeline.py
    VERSION
    __init__.py
```

As per the new considerations for production-level code, the following are the reasoning behind the structure of the production code -

1) Reliability/Reproducibility
2) Maintainability & Adaptability
3) Testability
4) Scalability
5) Version control

Understanding the functionality of each file and folder

Given below is the project folder structure showing all files and sub-folders. In the explanation, the numberings are given to show how each component is connected one after the other in a logical sequence, and the explanation is written in front of the corresponding file in the folder structure for ease of traceability of each file location.



```
project
    requirements
        requirements.txt
    titanic_model
        config
            core.py
            __init__.py
        datasets
            titanic.csv
            __init__.py
        processing
            data_manager.py
            features.py
            validation.py
            __init__.py
        trained_models
            __init__.py
        config.yml
        pipeline.py
        predict.py
        train_pipeline.py
        VERSION
        __init__.py
```

1. requirements: Dependencies-libraries, packages.

2. core.py : It accesses the config.yml file & setups the path for dataset folder trained models etc**.**

- **datasets:** Raw dataset is stored here.

3. data_manger.py: It contains pre-pipeline functions, and functionality for loading of data, saving & loading of trained models, and removing old version models.
4. features.py : It contains custom built transformation classes if any.
5. validation.py: To validate the user input data before feeding into the model for prediction. This is imported and used inside predict.py.
- **trained_models:** The trained model is stored inside this folder

6. config.yml: All the configurations and setting parameters, features name are stored here.
7. pipeline.py : It contains the complete pipeline using transformation classes and modeling algorithm
8. train_pipeline.py : Model training and saving of model
9. predict.py: provide inference for any new sample by loading the saved model.

- **VERSION** : Contains the version tag.

- **__init__.py** : This file lets the Python interpreter know that a directory contains code for a Python module. An __init__.py file can be blank. Without this, importing modules from another folder into your project is not possible.

PROCEDURE

The following steps involve downloading the project folder and uploading it to VS code, creating a virtual environment within the project, installing necessary dependencies, and executing specific scripts for training and prediction. It allows us to effectively configure the project and execute it within the VS code environment.

Step1: Environment Setup in your local system : Go through the **Environment setup document** and complete the VS code installation and along with the Python in your system.

Step 2: Go through the **Experiment Phase-1** **and** **Experiment Phase-2** colab notebooks and try to understand different steps viz. data exploration, pre-processing: pre-pipeline & pipeline processing, custom transformation class and final pipeline building. Note understanding these steps are essential to transition from the research phase to the production phase.

Step 3: Download the **Project folder** and upload in your VS code and follow the steps below:

1. Upload the Project folder in VS code

2. Inside Project create virtual environment :

   A. Open terminal, it should point to **…….\projects>**

   B. For creating virtual environment run the following command :

   ```
   python -m venv   path of project folder\venv
   ```

   For example:

   ```
   ….\project> python -m venv C:\Users\karna\Desktop\MLOps_M3_M4\project\venv
   ```

   C. Go inside venv\Scripts by command : ….> `cd venv\Scripts`
   Now type activate and enter : .... project\venv\Scripts> `activate`

   D. Now **press** : ctrl +shift+p; select Interpreter which contains **venv**
   Close the terminal and open new terminal, we are inside ven like this:
   (venv) ………………………………..\project>

3. Go inside requirements[`cd requirements`] folder and run :
   …> `pip install -r requirements.txt`
4. Come out of requirements folder [`cd..`] and go inside titanic_model folder and run :
   `py train_pipeline.py`

5. Now run : `py  predict.py`

Note: If you are getting errors while creating and activating a virtual environment follow the last few steps mentioned in the **Environment setup document**.