[Structuring ML Projects]

[2021]

[VERSION 0.1]

[TCS]

Authored by: [Yaswant Krishna, Baliram Pinate]

Contents 

Frameworks/Tools 

1) Data Preprocessing:
   - Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis.
   - Dask: A low-level scheduler and a high-level partial Pandas replacement, geared toward running code on compute clusters.
   - Rapids: A collection of data-science libraries that run on GPUs and include cuDF, a partial replacement for Pandas.
2) Data Visualization:
   - Static Visualization:
     i) Matplotlib.
     ii) Seaborn.
   - Dynamic Visualization:
     i) Plotly.
     ii) Bokeh.

Start of Project 

These are the following points that one needs to keep in mind before starting a project.

- Understanding the business problem.
- Understanding the data and code requirements.
- Data requirements like will the data fit in memory or is there any external data source through which we need to access the data.
- Code requirements like what is the preferred programming language and are the required frameworks available to achieve the desired objective.
- Are remote code versioning tools like github or gitlab available?
- Once the above steps are understood these are the things to do.
    - Creating a virtual environment for project.
    - Creating the folder structure as mentioned in the document.
    - Creating a git repository.
    - Following the code guidelines when writing code.
    - Documenting things.

Creating a Virtual Environment 

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use.

Every project must have a virtual environment and before the start of a project you need to create a virtual environment for that project. There are multiple ways to create a python virtual environment.

Method 1: Using python command venv.

Method 2: Installing virtualenv package using python pip tool.

Method 3: Using conda package manager. (Recommended)

Using the above methods, you can create a virtual environment. The below links help you with the commands to create virtual environments.

Method 1:

https://docs.python.org/3/library/venv.html

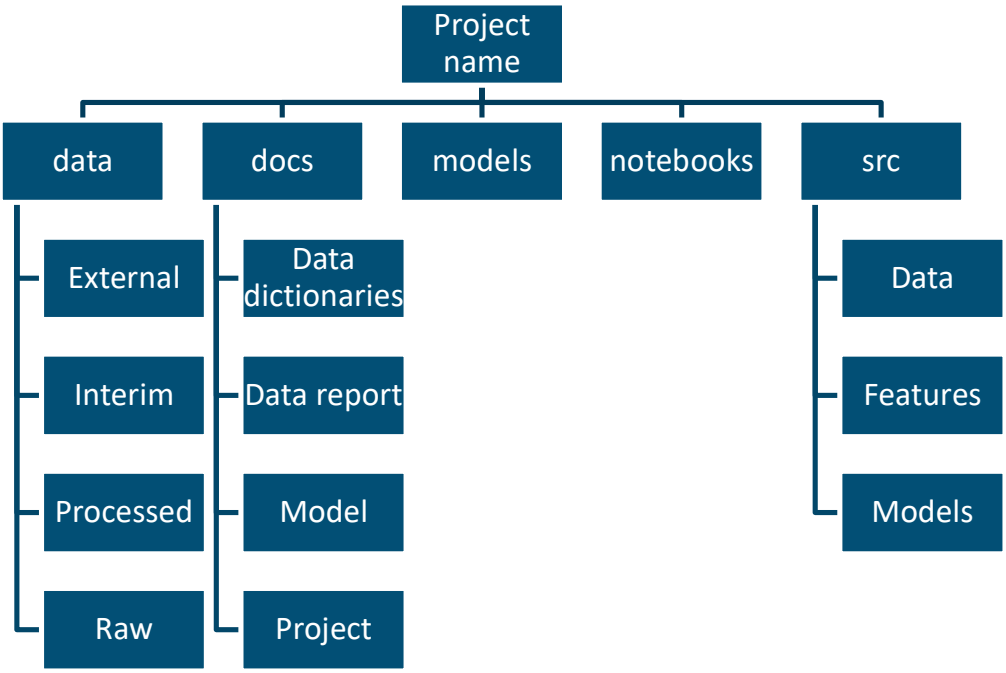
Method 2:

https://www.geeksforgeeks.org/python-virtual-environment/

Method 3:

https://www.geeksforgeeks.org/set-up-virtual-environment-for-python-using-anaconda/

Folder Structure 



Project name: Folder for hosting all files of a typical data science project.

Data: Folder for hosting code.

- External: Third party data.
- Interim: Transformed intermediate data, not ready for modelling.
- Processed: Prepared data, ready for modeling.
- Raw: Immutable original data.

Docs: Folder for hosting all documents of a project.

- Data dictionaries: Place to put data description documents, typically received from a client.
- Data report: Location to place documents describing results of data exploration.
- Model: Folder for hosting all documents and reports related to modelling.
- Project: Folder for hosting project documents and reports such as project planning docs and presentations.

Models: Serialized models.

Notebooks: Jupyter notebooks for exploration and prototyping.

SRC: Folder for hosting project source code.

- Data: Folder containing scripts to download/generate data.
- Features: Folder containing scripts to transform data for modelling.
- Models: Folder containing scripts to train and predict.

Coding Guidelines 

The Zen of Python:

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Imports: Imports are always put on top of the file. Imports should be grouped in the following order.

1. Standard library imports.
2. Related third party imports.
3. Local application/library specific imports.

Naming Conventions:

1. Variable names should be lowercase and readable.
2. CONSTANTS should always be in uppercase.
3. Function names should be lowercase and use underscores to improve readability.
4. For class names use CapWords convention.
5. Modules should have short, all lowercase names.

Comments: Comments that contradict the code are worse than no comments.

1. Block Comments: Block comments generally apply to some (or all) code that follows them and are indented to the same level as that code.
2. Inline Comments: Use inline comments sparingly. An inline comment is a comment on the same line as a statement. Inline comments should be separated by at least two spaces from the statement. They should start with a # and a single space.
3. Documentation Strings: Write docstrings for all public modules, functions, classes, and methods.

Exception Handling: Always use try and except blocks wherever necessary to catch exceptions and avoid runtime errors.

Useful Python Packages to Make your Life Easier 

s.no	Package	Remarks	Link
1.	missingno	Helps to visualize missing values in a pandas dataframe easily.	https://pypi.org/project/missingno/
2.	tqdm	Instantly make your loops show a smart progress meter and helps you know how much time it takes to execute that loop.	https://pypi.org/project/tqdm/
3.	prophet	If you are working on time series data. Then this library makes your work easier.	https://facebook.github.io/prophet/
4.	pycaret	If you want to test various models on the prepared data using a few lines, then this library helps you the most.	https://pycaret.org/
5.	streamlit	Streamlit turns data scripts into shareable web apps in minutes.	https://streamlit.io/

END of Project 

Once a project comes to an end these are the steps you can follow:

- Try to check if everything related to project is present in the git repository.
- Try creating a docker image of the environment and test the project by creating the same project environment in your system using the created docker image.
- Try creating a video recording of the entire project and upload the recording to the corresponding git repository.
- Once everything looks good delete the project virtual environment you have created in the beginning this will help you save space locally.
- If you have invented any new techniques or processes during the project, try creating a IDF document to patent the new techniques.

Learning Resources 

1) GIT:
   - Fresco Play is a good place to start learning GIT. It has everything one needs to know to feel comfortable using GIT.
2) Data Processing:
   - Pandas: Python for Data Analysis by Wes McKinney is available in O'Reilly eBooks in ultimatix. This book covers everything you need to learn about pandas, and it is written by pandas' creator himself.
3) Data Visualization:
   - Matplotlib and Seaborn: The documentation pages of this packages are a cool place to learn.
4) Model Building:
   - Scikit learn: Sklearn documentation is a good place to start.
5) Explainable AI:
   - LIME and SHAP: Analytics Vidhya and towards data science blogs are good.