1. **Set up the github**
   1. **new environment**
   2. **setup.py**
   3. **requirements.txt**
2. **Src folder and build the package**

**Create repo in GITHUB:** MLPro

**Open Anaconda prompt:**

cd G:\My Drive\DheepthiReddy\Work\DataScience\DS\_KN\Projects

G:

Code . - (This will launch a VS Code instance)

**VS Code:**

Open new terminal

Access conda prompt from the terminal by selecting command prompt in the terminal window.

**Creating a new env:**

conda create -p venv python==3.8 -y

venv: environment name

python==3.8: version used of python

-y: yes to all the instalments that happen later on

To clear screen:

Cls –

To activate environment:

conda activate venv/

connecting to git:

git init

**create a README.md file in the folders**

git add README.md

git commit -m "README.md file created"

git status

git branch -M main

git remote add origin <https://github.com/Dheepthi-Reddy/MLPro.git>

git push -u origin main

create a .gitignore file in the project and choose the template as python in the github browser dashboard and commit the changes

take a pull in the venv:

git pull

**create setup.py:**

It is responsible in creating the ML project as a package and can be deployed in Pypi

\_\_init\_\_.py: to find the folder as a package

Write in setup.py and reuirements.txt

pip install -r requirements.txt

**mlproject.egg-info** folder is created after the command was run

1. **Logger**
2. **Exception**
3. create **components** folder in src and a init.py file to it:components folder is for all the modules that are to be created in it.
   1. **data\_ingestion.py** file is in components folder: code to read the data, like separating train data and test data
   2. **data\_ingestion.py** file is in components folder: code to read the data, like separating train data and test data
   3. **data\_transformation.py** file in components folder: code related to transformation like changing categorical features to numerical features, handling one-hot encoding and…
   4. **model\_trainer.py** file in components folder: this file to train the model and kinds of the model used and to call confusion metrics to solve classification problem….
4. Create **pipeline** folder in src: kind of pipeline that needs to be created like, training and prediction pipelines.
   1. **train\_pipeline.py** file is created in pipeline folder: has code for all the training pipeline and from this pipeline we try to call all the components of components folder
   2. **predict\_pipeline.py** file is created in pipeline folder:
5. **exception.py**, **logger.py**, **utils.py**(common functionality) files in src folder.
6. Execute logger.py in terminal:

python src/logger.py

a new log folder is created with a .log file in the format we created

1. **Problem Statement**
2. **EDA**
3. **Model**
4. Create a notebook folder:
5. Add a data folder and place the student performance dataset in it:

The dataset has all kinds of features in it, like categorical, numerical, Nan and etc…

1. Add the .ipynb files:
   1. EDA File:

EDA is better when tried in ipynb file. To run the EDA file connect to venv environment(ctrl+shift+p), then install ipykernel.

**Data Checks to perform:**

* + 1. Checking missing values
    2. Check duplicates
    3. Check data type
    4. Checking the no of unique values of each column
    5. Check statistics of data set
    6. Check various categories present in the different categorical column

**Feature Engineering:**

Creating new features using existing features or choosing only few features that are more important

Example of feature engineering in this dataset:

Adding columns for “Total score” and “average”

These new features are dependent features as they are depending on the existing features.

1. Now run requirements.txt file again:

->Add matplotlib to requirements.txt

Before working on the EDA ipynb file

pip install -r requirements.txt

1. Run requirements.txt file again:

- Add scikit-learn, catboost, xgboost to requirements.txt

Before working on Model training ipynb file

pip install -r requirements.txt

1. Work on Model training:

Perform One-hot encoding to represent categorical features as numerical features.

One-hot encoding and StandardScaler are performed one after the other on a column to transform it

**Data Ingestion:**

The first is to read the dataset, the dataset is collected from different sources, these data sources can be various types.

We create artifacts folder and using train\_test\_split(from sklearn.model\_selection) method we divide the data.csv file into test and train.

We create a artifacts folder in which these data.csv, train.csv, test.csv files are placed.

Run the dataIngestion file:

python src/components/data\_ingestion.py

before pushing add artifacts folder to .gitignore file

**Data Transformation:**

We have different kinds of features in our dataset like, categorical, numerical features…,

The purpose of data transformation is to do:

Feature Engineering(FE), data cleaning, converting categorical features to numerical features.

We create DataTransformation class and:

1.Take numerical and categorical features separately

Then we create pipelines where:

* + - 1. we handle missing values.
      2. StandardScalar

2.combine these two with the help of scikit learn preprocessing

**Utils file & requirements.txt:**

To save the pickle file:

Create a common save function in utils file and add ‘dill’ to requirements.txt and download by running the file.

**Data\_ingestion:**

Importing data\_transformation in data\_ingestion

To check if the data transformation is working, run data\_ingestion file

**Model Training:**

Write training code and import it in dataIngestion.py

Push to github