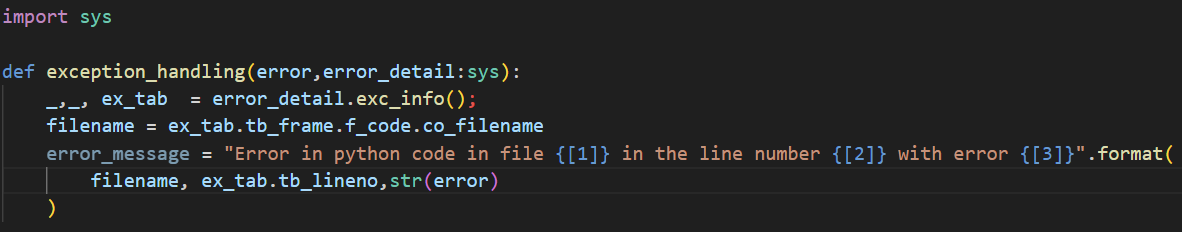
Steps Followed-

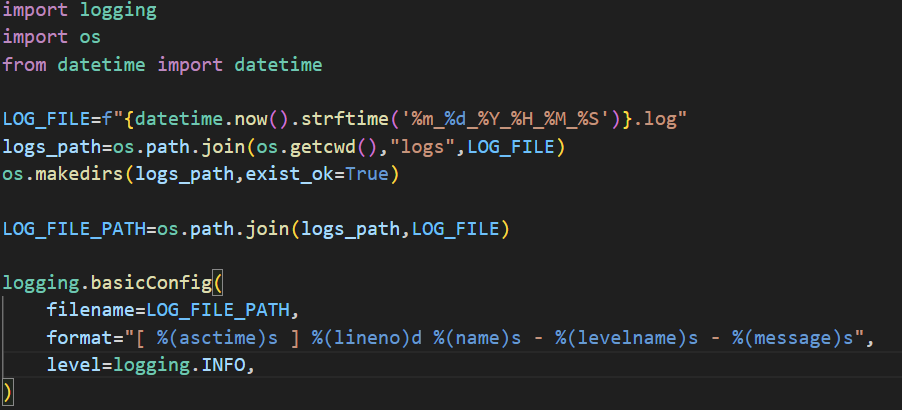
* Created Github Repo named MLOPS.
* Created a Python Environment using : conda create -p venv python==3.12.2 -y , -y is used to give permission to install anything required so it won’t ask while running.

On creating the environment directly in VSCODE terminal it was giving me an error that conda not found so I created the environment in Anaconda prompt directly.

* Connected Local folder MLOPS to github.
* Created a .gitignore file in github.
* Then we will create Requirement.txt
* Then setup.py is created. Setup.py is is helpful in converting our python application into a package which can be later installed and used on any machine. Also this package can be deployed on pypi.
* After Setting up setup.py we will create a folder src in which a file \_\_init\_\_.py which can be used to build a package.
* Add ‘-e .’ at the end of requirements.txt file to automatically execute setup.py
* Install requirements using “ pip install -r requirements.txt”
* **Lecture 2**
* Created a folder named “Components” in src which will contain modules for different functions like data ingestion, handling etc.
* Also added \_\_init\_\_.py file in the components folder.
* [**https://chatgpt.com/share/728fc3e2-8ffd-450b-a88a-7a9aeb4ad936**](https://chatgpt.com/share/728fc3e2-8ffd-450b-a88a-7a9aeb4ad936)*Relation b/w setup.py and \_\_init\_\_.py .*
* Created a file name *data\_ingestion.py* which will be used to fetch data from databases or any external source. Also it will contain code for train test split and validation split.
* Created a file named *data\_transformation.py* which will be used to transform the fetched data using techniques like categorical handling, missing values handling.
* Created *model\_trainer.py* which will contain code for training the model and validating.
* Now a new folder called *“pipeline”* is created which contain 2 pipelines one is *model\_trainer.py* which trigger various modules present in components directory for training purpose.
* The second pipeline would be *model\_testing.py* which will trigger modules present in components directory for testing purpose.
* Created 3 files in src folder named exception.py , logger.py , utils.py.
* In exception.py we will use “sys” library.

**

The *error\_detail.exc\_info()*  provides 3 details and 3rd one is the most useful. It will provide details like in which file the error has occurred , in which line it has occurred etc.

* In logger,  
  <https://chatgpt.com/share/bc55a8b5-b4c9-4239-a562-ddfd2914ffee>  
  

* **­­­LECTURE 3**
* Included a dataset “*Student Performance indicator*”.
* Problem Statement - This project understands how the student's performance (test scores) is affected by other variables such as Gender, Ethnicity, Parental level of education, Lunch and Test preparation course.
* Dataset information

- gender : sex of students -> (Male/female)

- race/ethnicity : ethnicity of students -> (Group A, B,C, D,E)

- parental level of education : parents' final education ->(bachelor's degree,some college,master's degree,associate's degree,high school)

- lunch : having lunch before test (standard or free/reduced)

- test preparation course : complete or not complete before test

- math score

- reading score

- writing score

* **Life cycle of Machine learning Project**

- Understanding the Problem Statement

- Data Collection

- Data Checks to perform

- Exploratory data analysis

- Data Pre-Processing

- Model Training

- Choose best model

* **Data Check to perform**

- Check Missing values

- Check Duplicates

- Check data type – *data.info()*

- Check the number of unique values of each colun – *data.nunique()*

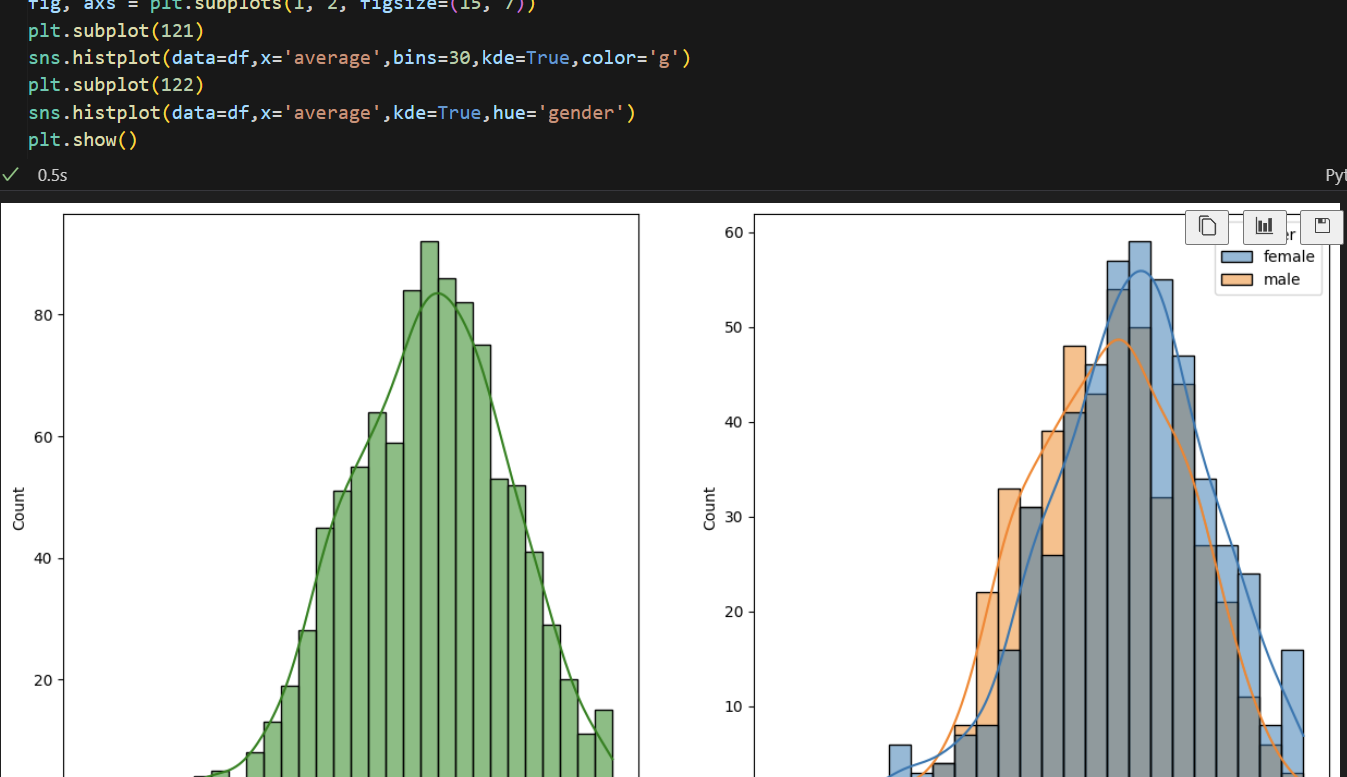
- Check statistics of data set

- Check various categories present in the different categorical column

* No missing values found.
* To check duplicate row we can use *data.duplicated()* this will return true for all duplicate rows. We have a parameter *keep* whose value can be “first” , “last” and “false” . On keeping keep = “first” all the duplicate rows except the first appearance will the returned as true.

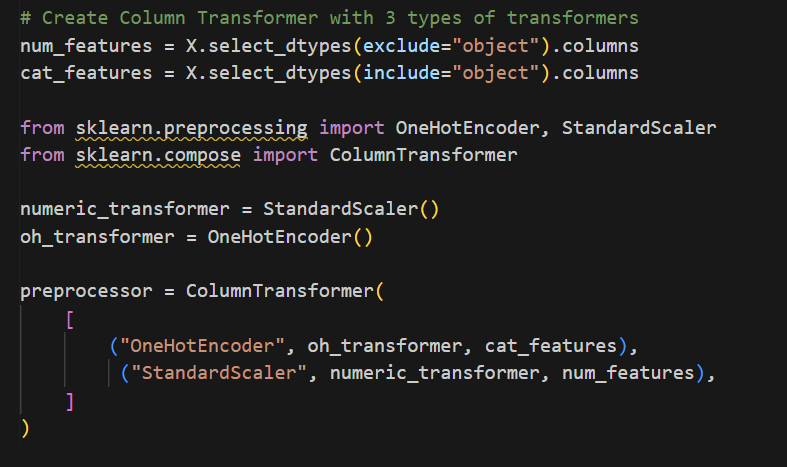
***No Duplicate Value found***

* Feature Engineering Steps
* Formed 2 different sets one for numerical features and other for categorical features(dtype == ‘O)
* Then created 2 different features Total marks and Average . This columns will be my Output
* Created a sns plot for Total score and Average and then created one giving hue = ‘g’
* Indicated females have better total score and average than males

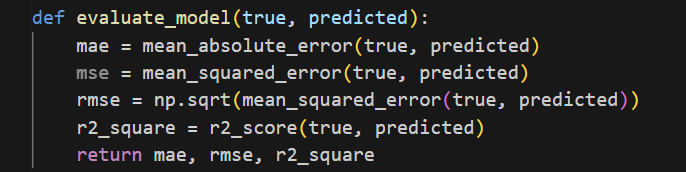


**Model Training**

* Initially divide features in two types , numerical and categorical.
* Then apply onehotencoding on categorical features and stardization on numerical features.



* Here **column transformer** has created a pipeline which is applying OneHotEncoding on Categorical feature while StandardScaler on numerical Features.
* Now we can fit any dataset in preprocessor and it will apply OHE and Stand. Based on the datatypes of columns.
* Created an evaluation function



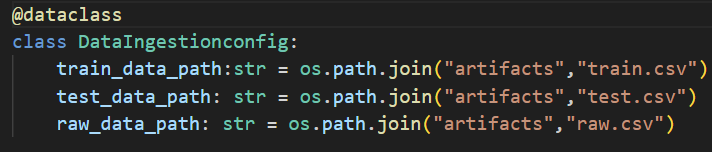
* Then model training is performed using different models



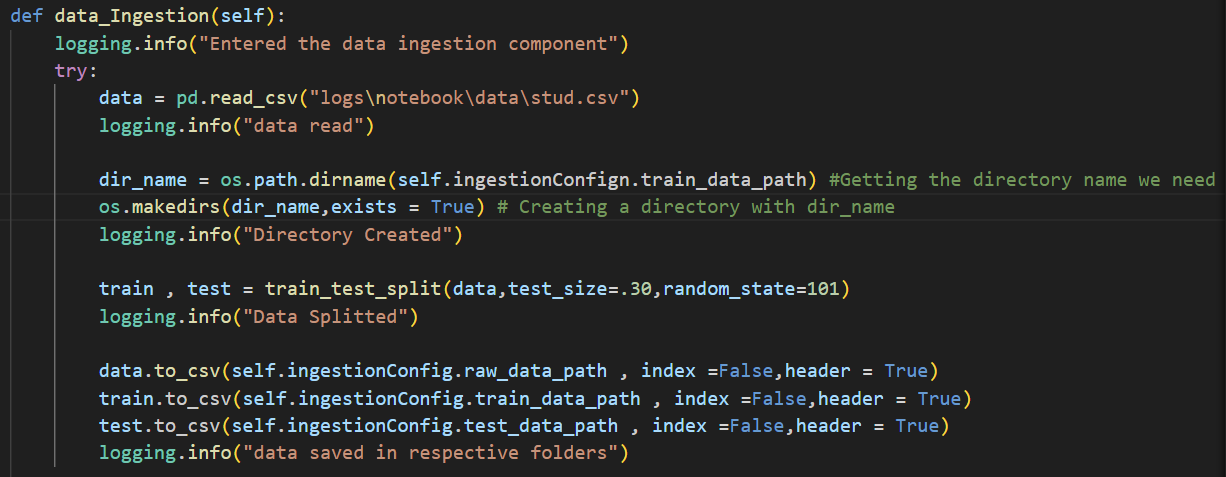
**The above 2 EDA and model training was a naive approach but now we will convert the ml and specially model training part into a modular programming framework.**

We start by creating data ingestion file. In data ingestion file firstly we create a class “DataIngestionConfig” which will be used to get any input we require such as path where we want to store test and train data.

We will use @dataclass decorator which will allow us to define variables in DataIngestionConfig class without needing to define \_\_init\_\_ or any other method. We will create path variable for train,test,raw data indicating where all these things will be stored. We will save then in an “artifacts” folder.

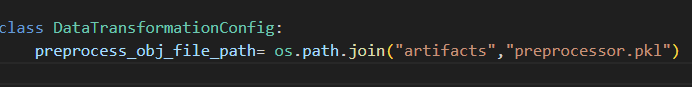


Now we define a class “DataIngestion” to perform actual data ingestion. We will create a variable ingestionConfig to hold all the paths. Then define a method “data\_ingestion” which we can modify based on the source of data like if it’s a Database like MongoDB or a local machine file.



**Data Transformation**

In the provided code, the preprocess\_obj\_file\_path is a file path where a preprocessor object will be saved. This object is typically used to transform raw data into a format suitable for machine learning models.



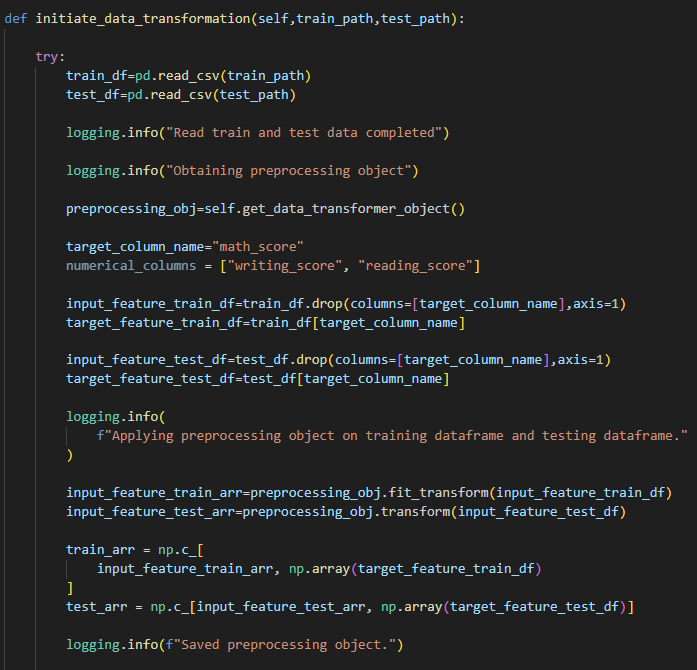
****

We created a function that will transform numerical and categorical features into the form useful for model training. For each type i.e. numerical and categorical we created a pipeline to execute. Column\_Transformer combines both the pipeline and execute numerical pipeline on numerical features and categorical pipeline on categorical features.

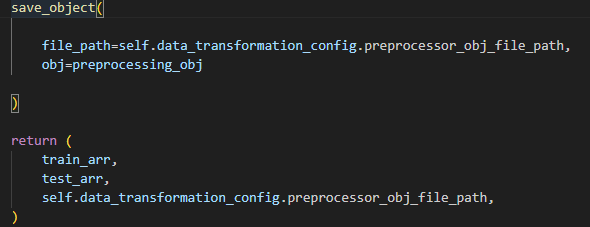
We created a class DataTransformationConfig: which will store the path where we want to store our preprocessor object.

We then created a DataTransformation class which will contain functions on how to transform the data and a function containing the steps to read and then transform the data.





We then saved the preprocessor object on the defined path and returned the preprocessed arrays.

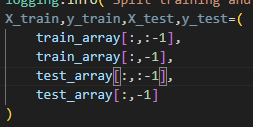


Created a function save\_object in utils.py to save the Preprocessor\_ object.\

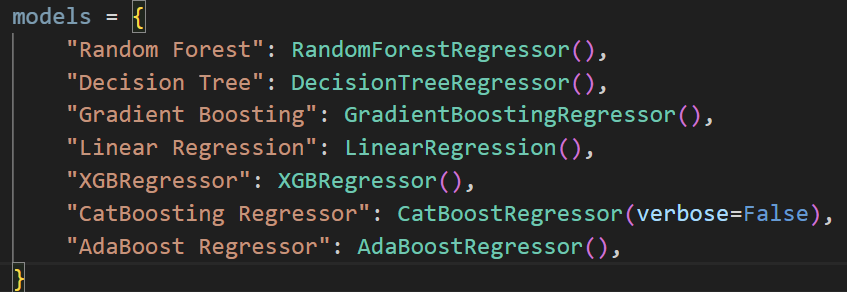
**Model Training**

Initially create ModelTrainingConfig class to define path to save model.

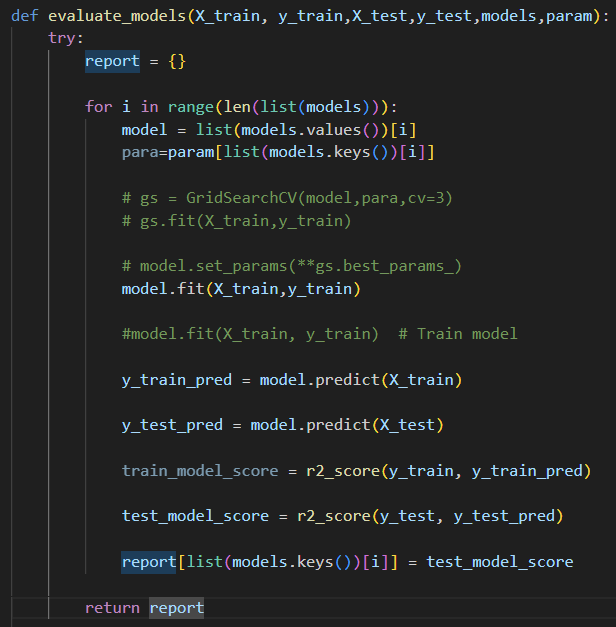
Data Transformation is returning train array , test array but with target feature(not transformed but attached with transformed data ) so we will first detach the target column from train, test transformed array. X\_test and Y\_test contains target column while other 2 doesn’t.



We created a dictionary of all the models we are going to use



Then we will create a evaluateModel function which will find r2 score for each of the models and return a dictionary containing model name and r2 score of that model.



Once we found the best score we will save the model