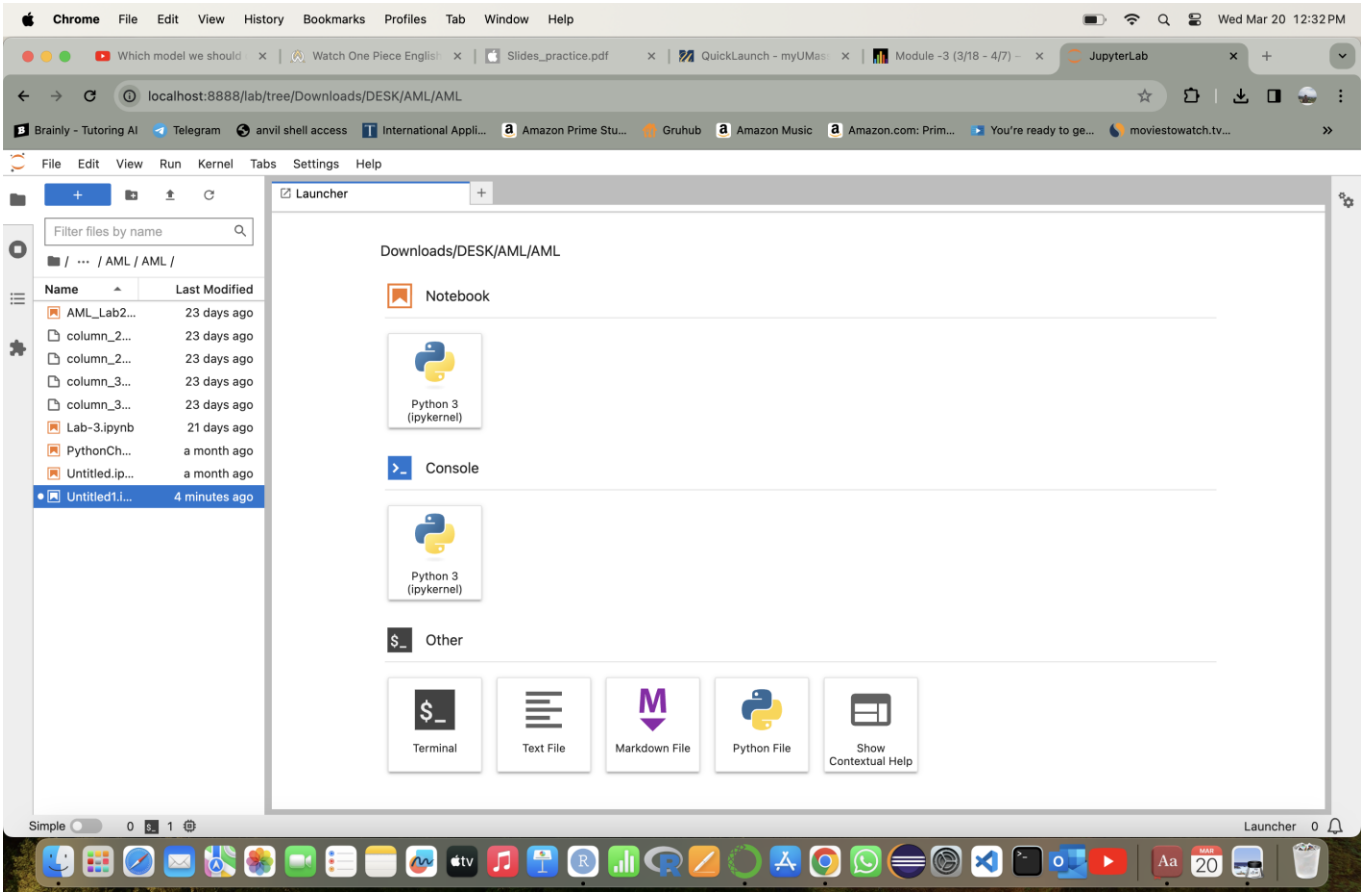
Jeevan Kumar Banoth - 02105145

Advanced machine learning

Homework – 4:

➢ Creating Jupyter Notebook with Anaconda navigator:

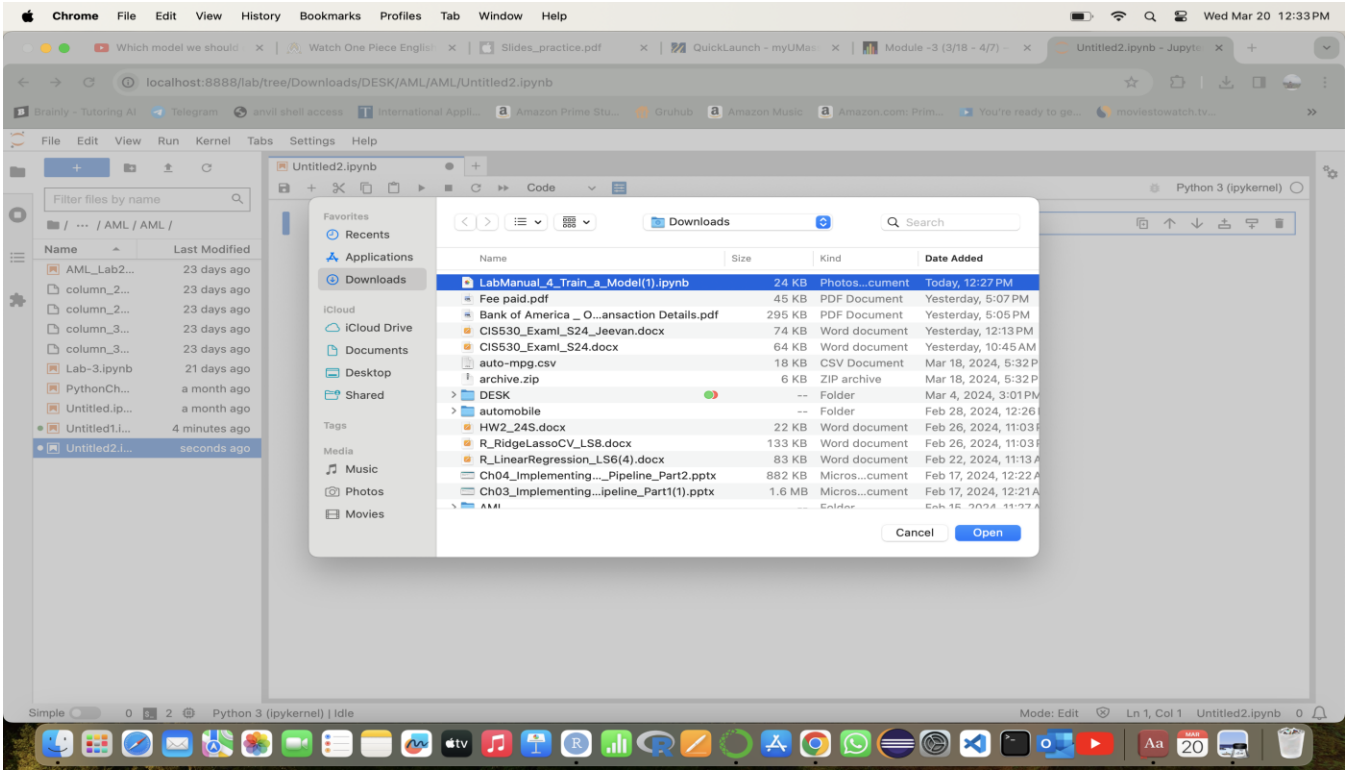


→ By using Anaconda navigator, first I opened Jupyter lab, and redirected it to my working folder.

→ In the left side of the notebook, we can see that there are files on which we have previously worked.

→ Created a notebook for this task by selecting File > New, Notebook, and then conda_python3 in the kernel dialog window.

→ But, for this lab there is no need to work on the new notebook so we will start it by uploading.



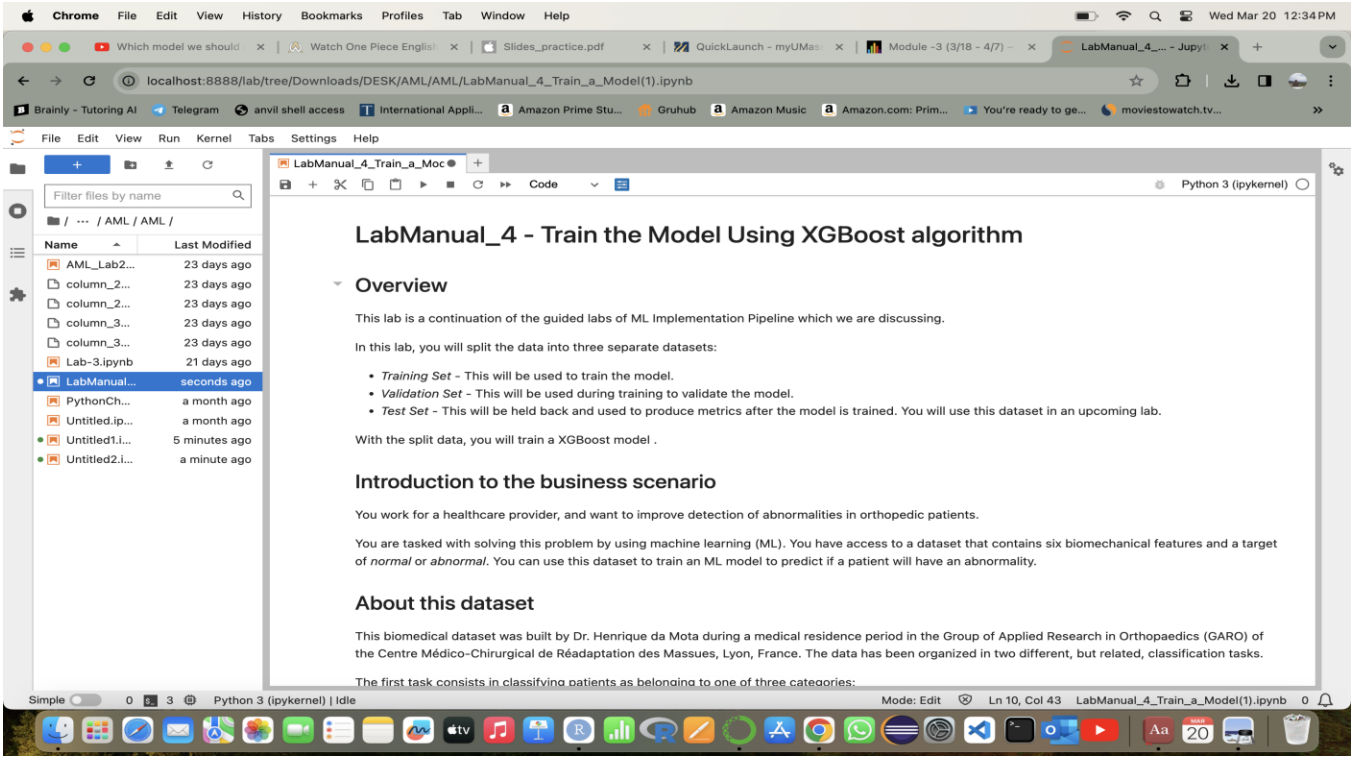
→ Using the Upload button from left side corner of the jupyter lab, we need to upload the file which we have downloaded from the MyCourses site.

→ By using the upload button, I have uploaded the ipynb file from my download section of my laptop.

→ After that we can see that the file has some code and instructions about the code on which data it is and all.

→ The file is about Training the model using XGBoost algorithm.

→ In this lab, we have to just run all the cells and check the output and understand the data.



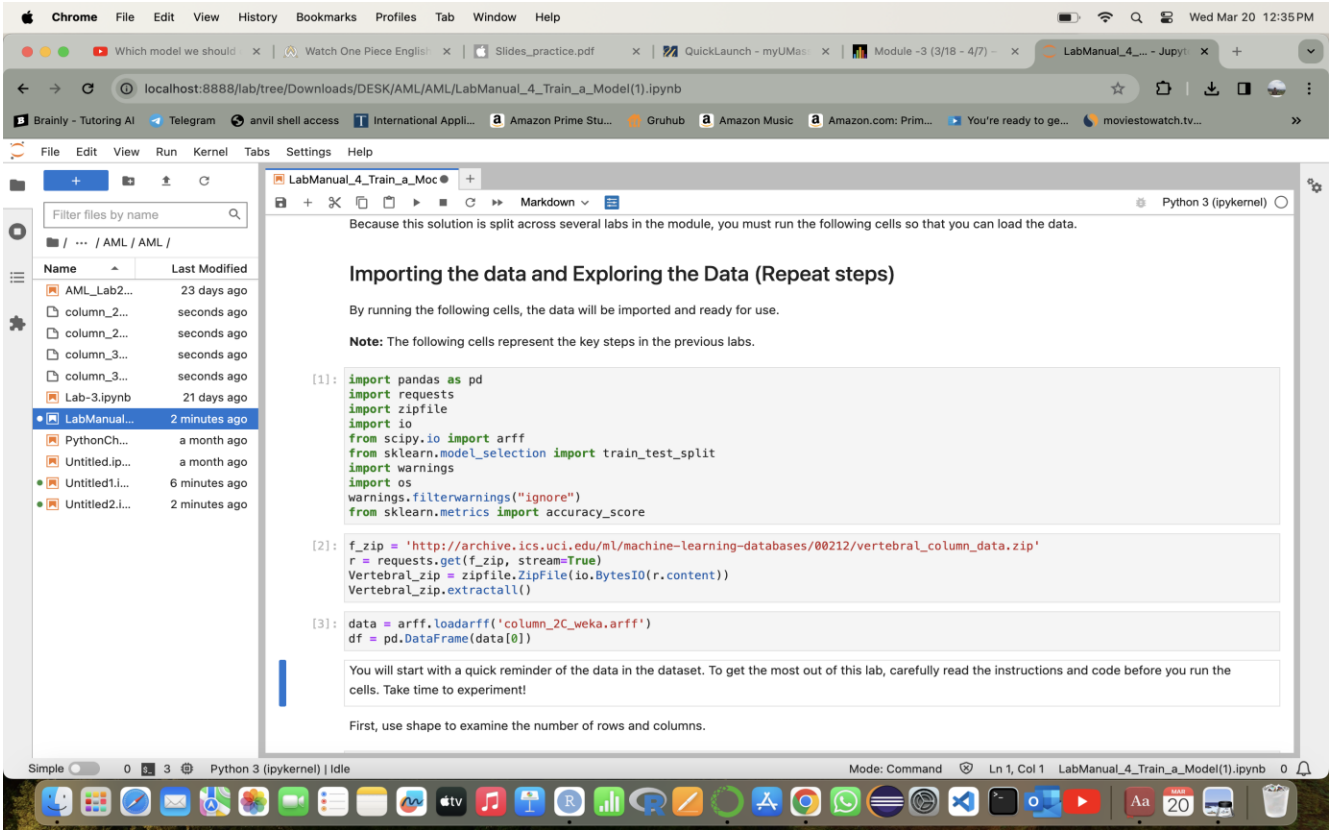
→ As we can see, above is the preview of the ipynb file after we uploaded, now we are just running the code whatever is present in that file and checking the output.

→ In this lab, we work with a health care provider, and we want to improve detection of abnormalities in orthopedic patients.

→ We have a dataset which contains six biomedical features and targets the normal and abnormal.

→ We must train and predict the dataset if a patient has an abnormality.

→ This bio medical data was built by Dr. Henrique da Mota during their medical residence period.



→ Now we are importing and exploring the dataset, and Training the Model Using XGBoost algorithm

→ This lab continues the guided labs of ML Implementation Pipeline we are discussing.

→ In this lab, we will split the data into three separate datasets.

→ *Training Set* - This will be used to train the model.

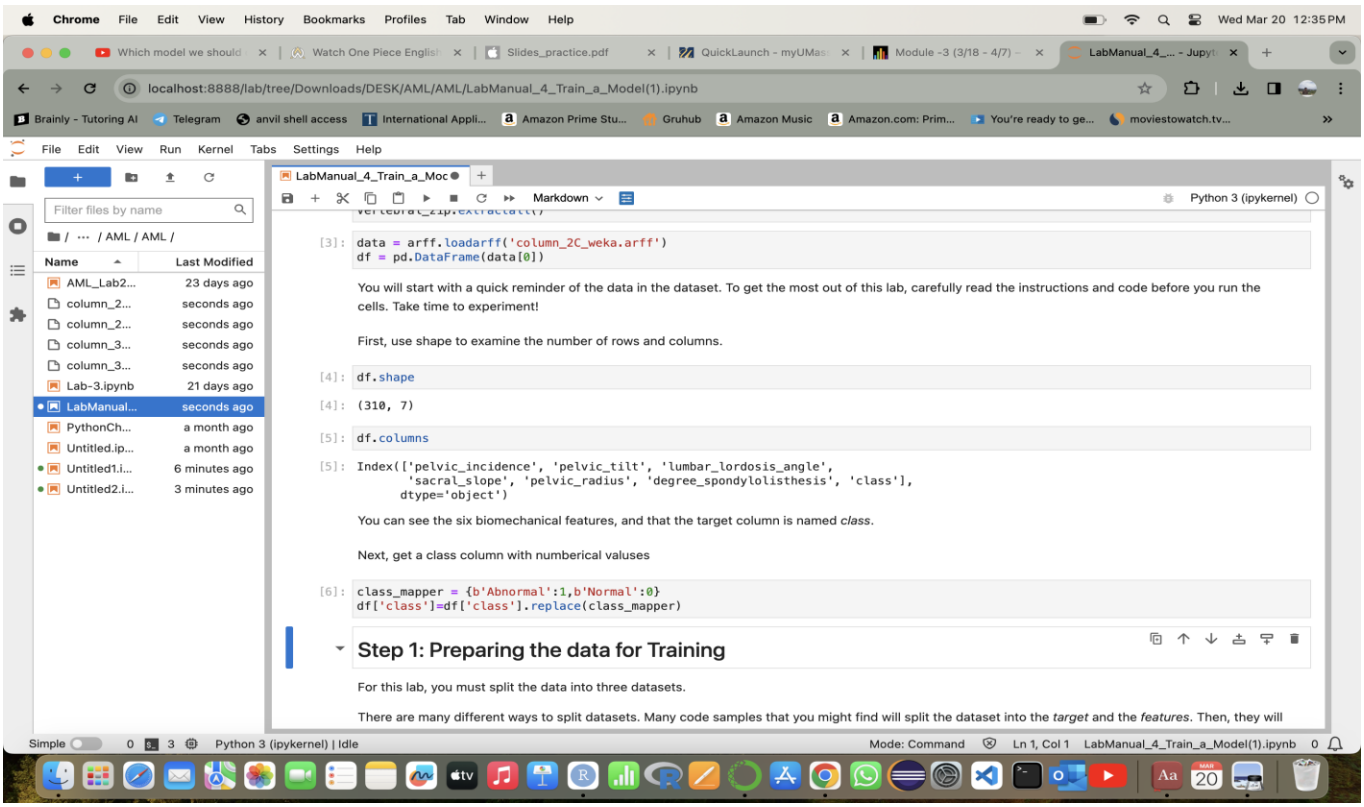
→ *Validation Set* - This will be used during training to validate the model.

→ *Test Set* - This will be held back and used to produce metrics after the model is trained.

→ We are now downloading the data by using f_zip from uscis website.

→ We have unzipped it by using extractall() function.

→ Here, the dataset was given the name with df, so wherever we use the dataset, we call it by using df.



→ df.shape and df.columns command used in the next step for checking dimensions of the DataFrame and column labels of the DataFrame respectively.

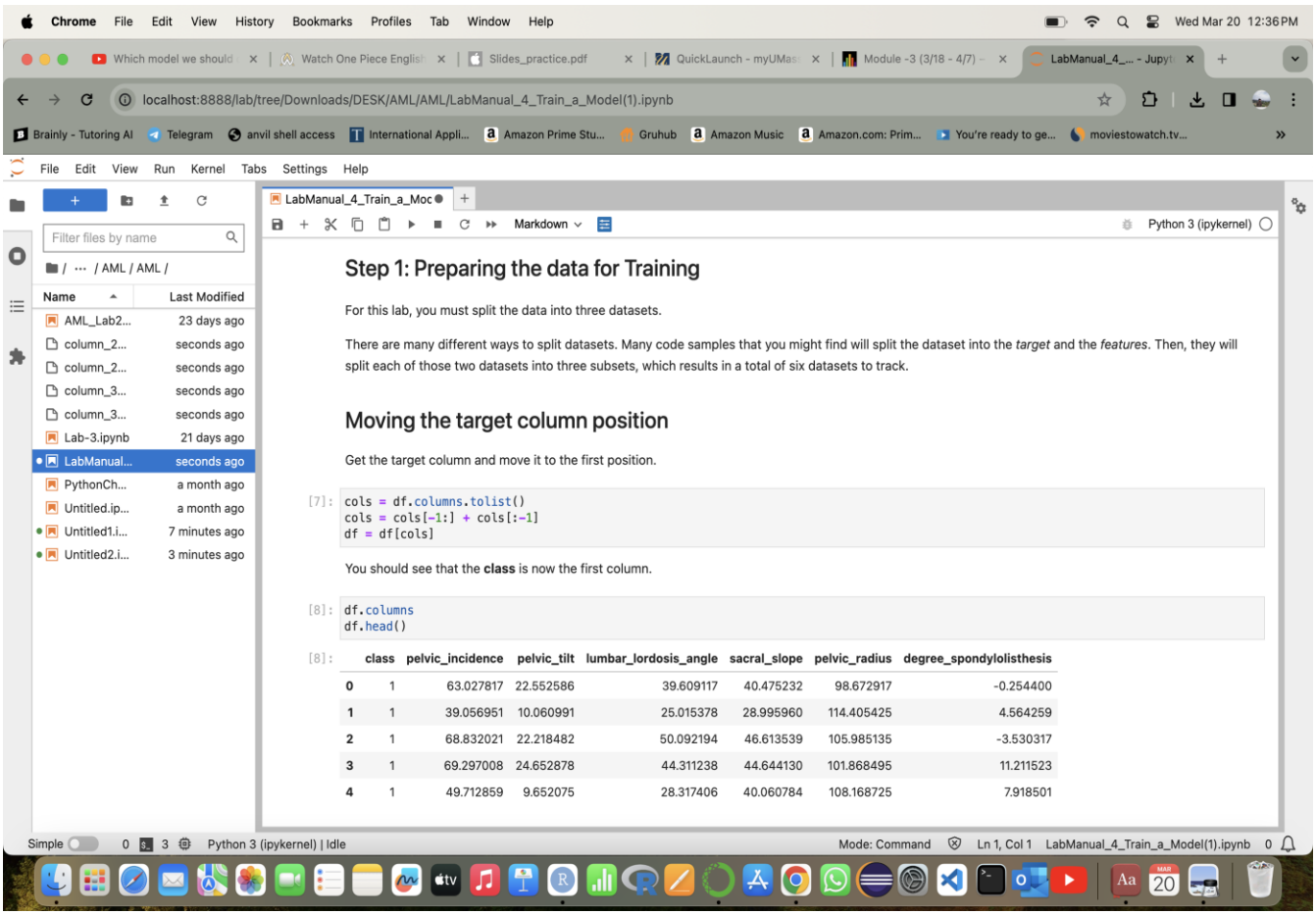
→ Next, I have used class_mapper DataFrame for transforming the values in “class” column of pandas DataFrame from byte string labels to String labels.

→ In the columns, we can see there are six biomechanical features, and that the target named class.

→ We use here column class with numerical values.

→ Then, we are preparing the Dataset for Training. For this lab, we split the data into three datasets.

→ There are many ways to split the datasets.



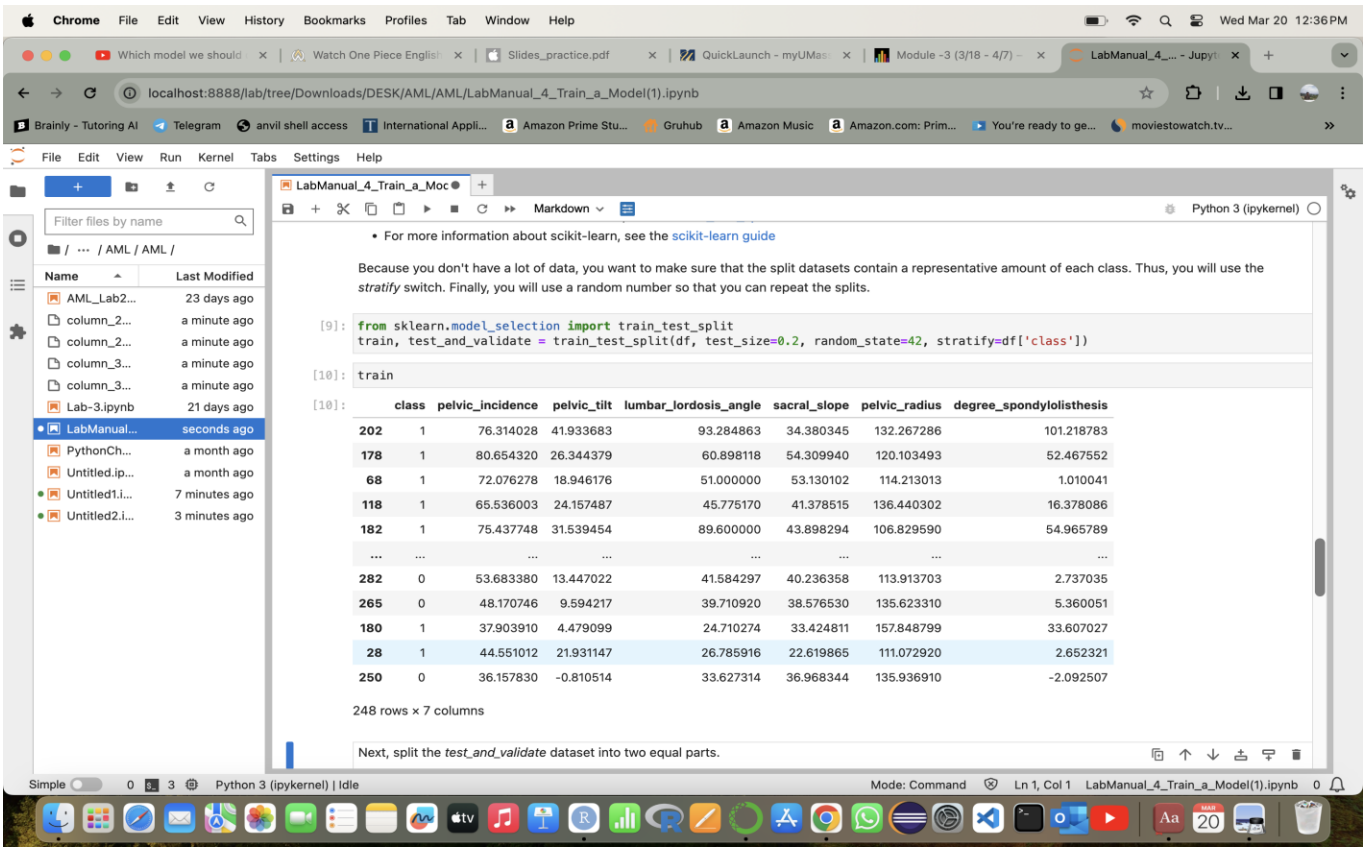
→ As we are splitting the data into three datasets, in the first part we are preparing it for training.

→ We are moving the target column position to the first place in the dataset as you can see that in the above image.

→ This code snippet rearranges the columns of the DataFrame **df** so that the last column becomes the first column, and then it displays the new column order and the first few rows of the updated DataFrame.

→ df.columns shows all columns in DataFrame after rearranging them and df.head() displays the first five rows of the DataFrame after column rearrangement.

→ As we don't have much data, we must ensure that the split data contains their respective columns in the dataset.



→ Next, we can see the split data shape and values.

→ And we will use here the random numbers, so that we can repeat the steps for splitting the dataset.

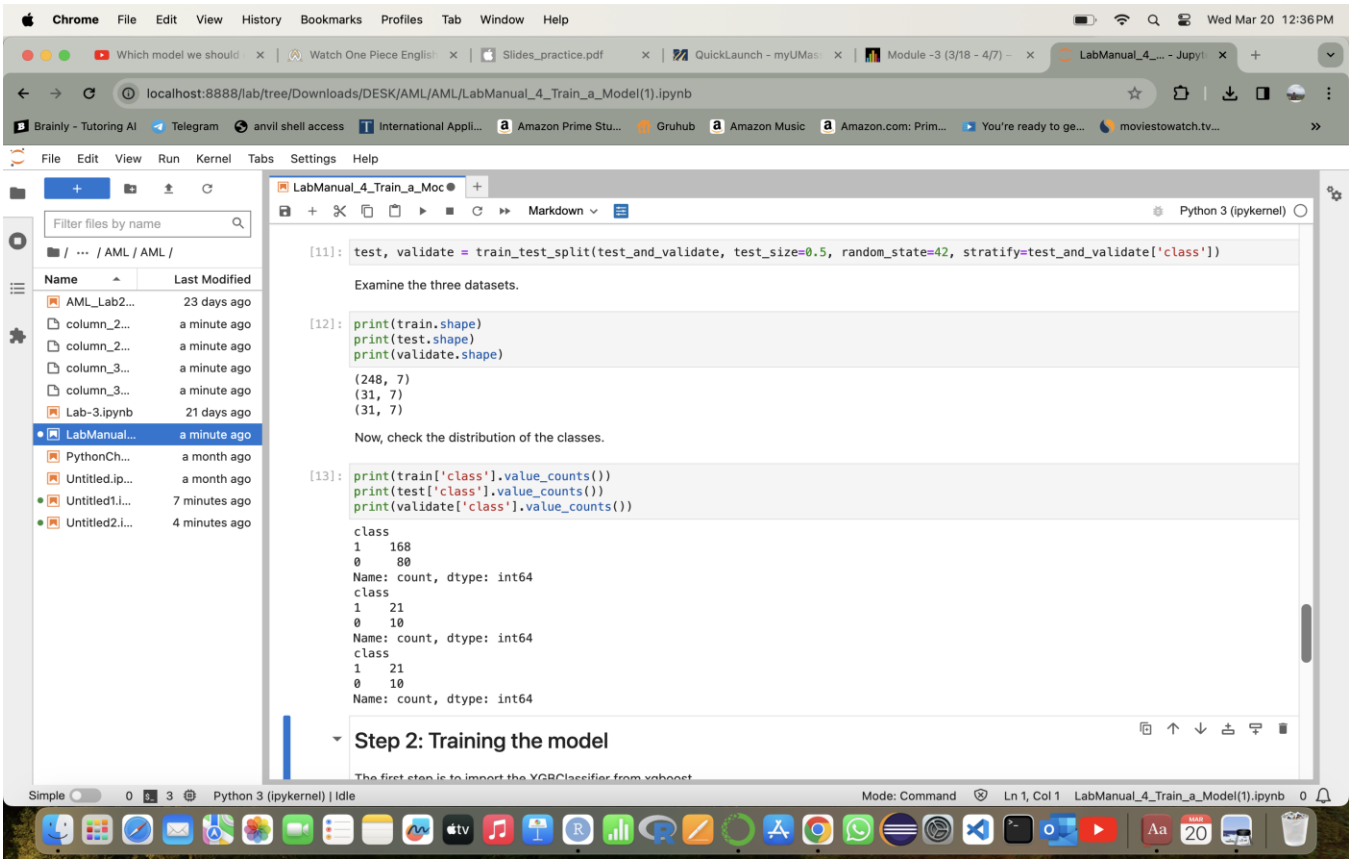
→ Here we are using sklearn for training the dataset.

→ We are not just training the dataset, but also testing and validating it by using test and validate function.

→ Separating a dataset into training and evaluation parts is crucial in machine learning. This allows you to train your model on one set of data and evaluate its performance on a different set.

→ By doing this, you can ensure that your model can generalize well, meaning it can perform well on data it has not been trained on.

→ We are using the 'train' function for training the dataset.



→ First, we need to examine the three datasets by printing them.

→ Then we must check the distribution of classes by using value_counts() function for each function

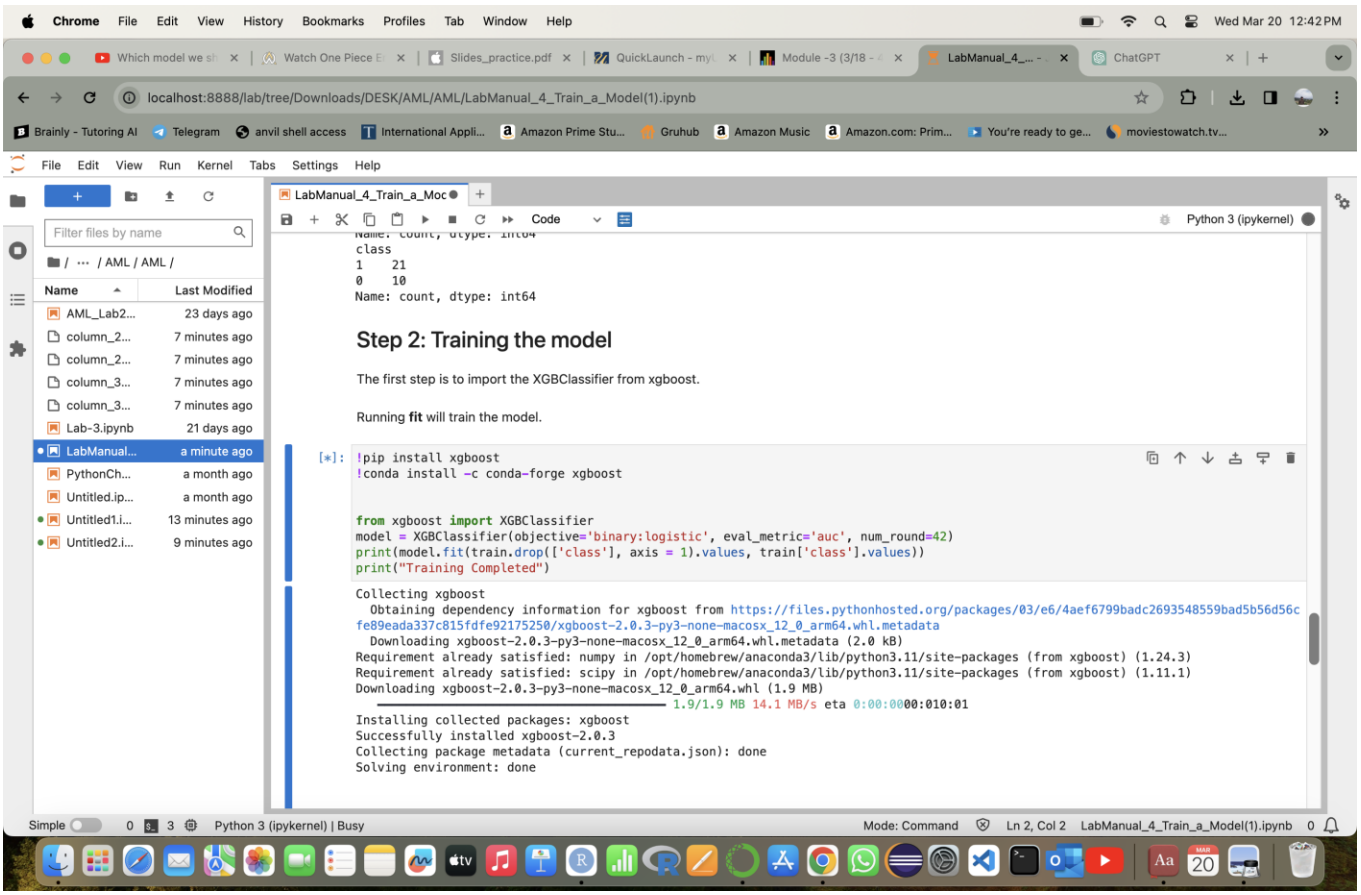
→ The test and validation sets are divided into two equal parts, called test and validation, each containing 10% of the original data.

→ The shape (size and dimensions) and distribution of classes in each of these subsets is then printed to provide an overview of data segmentation.

→ This process ensures that the subsets used for training, testing, and validation have similar proportions of different class labels, making the data distribution balanced.

→ Next, we will train the model by using XGB classifier from the XGBoost.

→ The final step will be running the model.



→ While executing the final training model, I saw an error as there is not xgboost package, so I installed it.

- This code aims to install the XGBoost library using two methods: pip and Conda.
- Once installed, it creates an XGBoost classifier from the training data.
- The classifier uses a binary logistic objective to predict the value of the 'class' column.
- To evaluate the classifier's performance, it employs the Area Under the Curve (AUC) metric.
- After the successful completion of training the datasets, we will be printing the "Training completed" as the sign of it is executed without any errors.

Conclusion:

- This Python code helps with machine learning and data analysis. It begins by loading essential libraries like pandas and sklearn.
- Next, it gets a zip file from the UCI repository that has data on vertebral columns. It then extracts the ARFF file from the zip archive and transfers it to a pandas DataFrame.
- Finally, it adjusts the Data Frame's columns by shifting the 'class,' or target variable, to the front.
- The data is divided into training and testing/validation sets with the `train_test_split` function.
- The testing/validation set is further split into test and validation subsets, keeping the class distribution consistent across all three sets.
- The script installs the XGBoost library and creates an `XGBClassifier` object with specific parameters for the objective, metrics, and number of iterations.
- The `XGBClassifier` is then trained using the training data, with the feature columns as input and the 'class' column as the target.
- The model is trained and then printed.
- We discovered the significance of using 'df.columns' to verify that our primary data column ('target') is located correctly in our dataset.
- To make modeling easier, the DataFrame columns are rearranged to have the target variable ('class') at the start.
- The dataset is divided into separate training, validation, and testing groups using a stratified sampling technique.
- To assess the performance of your XGBoost model, utilize metrics such as accuracy, precision, recall, and the AUC-ROC curve on the unseen testing set.
- Optimize model performance by fine-tuning hyperparameters using the validation set.
- Experiment with various machine learning algorithms and compare their outcomes.
- This code serves as a strong basis for constructing and evaluating an XGBoost model for classifying data related to the vertebral column.

- To customize the model, explore the data's various characteristics and modify the model to meet your unique criteria.
- This helps to keep the target variable's distribution consistent across these subgroups. This step is essential for assessing the model's performance.
- Handling data properly Dividing data into sets with similar class distributions (stratified splits) to ensure training and testing representativeness.
- Employing an advanced modeling approach (XGBoost) for making predictions.
- Once the model is trained on the chosen data (excluding the "class" variable), it is considered complete and ready for assessment.
- The message "Training Completed" signifies the conclusion of the training phase.
- The model is now prepared to be evaluated based on its performance on the test and validation datasets.

-------X------