UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Plagiarism Detector

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Good job! Your model works well, and questions answered with good thoughts.

Random Forest does help a lot in this problem, while when there are more training data, we can consider neural networks too.

Congrats on completing the program!

All Required Files and Tests

The submission includes complete notebook files as `.ipynb` : "2_Plagiarism_Feature_Engineering" and "3_Training_a_Model". And the test and helper files are included: "problem_unittests.py", "helpers.py". The submission also includes a training directory `source_sklearn` OR `source_pytorch` .

All the unit tests in project have passed.

Notebook 2: DataFrame Pre-Processing

The function `numerical_dataframe` should be complete, reading in the original file_information.csv file and returning a DataFrame of information with a numerical `Category` column and new, `Class` column.

There is no code requirement here, just make sure you run all required cells to create a `complete_df` that holds pre-processed file text data and `Datatype` information.

Notebook 2: Features Created

The function `calculate_containment` should be complete, taking in the necessary information and returning a single, normalized containment value for a given answer file.

Provide an answer to the question about containment feature calculation.

The function `lcs_norm_word` should be complete, taking in two texts and returning a single, normalized LCS value.

Define an n-gram range to calculate multiple containment features. Run the code to calculate one LCS feature, and create a DataFrame that holds all of these feature calculations.

Notebook 2: Train and Test Files Created

Complete the function `train_test_data`. This should return only a *selection* of training and test features, and corresponding class labels.

Select a few features to use in your final training and test data.

Provide an answer that describes why you chose your final features.

Implement the `make_csv` function. The class labels for train/test data should be in the first column of the csv file; selected features in the rest of the columns. Run the rest of the cells to create `train.csv` and `test.csv` files.

Notebook 3: Data Upload

Upload the `train.csv` file to a specified directory in an S3 bucket.

Notebook 3: Training a Custom Model

Complete at least *one* of the `train.py` files by instantiating a model, and training it in the main if statement. If you are using a custom PyTorch model, you will have to complete the `model.py` file, as well (you do not have to do so if you choose to use an imported sklearn model).

Define a custom sklearn OR PyTorch estimator by passing in the required arguments.

Fit your estimator (from the previous rubric item) to the training data you stored in S3.

Notebook 3: Deploying and Evaluating a Model

Deploy the model and create a `predictor` by specifying a deployment instance.

Pass test data to your deployed `predictor` and evaluate its performance by comparing its predictions to the true, class labels. Your model should get at least 90% test accuracy.

Provide an answer to the two model-related questions.

Notebook 3: Cleaning up Resources

Run the code to clean up your final model resources.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START