## Meets Specifications

Congratulations on passing this project!👏  
It was really a great experience reviewing your project.😄  
Keep learning and don't stop!💪 Good luck with future submissions :)

## 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 files are correctly submitted. Great!

All the unit tests in project have passed.

You passed all the unit tests in the project. Well done!

## 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.

You can refer to the following code:

data = pd.read\_csv(csv\_file)

data.replace({'Category': {'non':0, 'heavy':1, 'light':2, 'cut':3, 'orig':-1}}, inplace=True)

data.loc[data['Category']==0, 'Class'] = 0

data.loc[data['Category']==-1, 'Class'] = -1

data['Class'].fillna(1, inplace=True)

return data

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.

Great! 😄

You can also refer to this implement:

def calculate\_containment(df, n, answer\_filename):

answer = df.query('File == @answer\_filename')

source = df.query('Task == @answer.iloc[0].Task and Category == -1')

vectorizer = CountVectorizer(ngram\_range=(n,n))

counts = vectorizer.fit\_transform([answer.iloc[0].Text, source.iloc[0].Text]).toarray()

return np.sum(np.minimum(counts[0], counts[1])) / np.sum(counts[0])

Provide an answer to the question about containment feature calculation.

Great answer!

The function lcs\_norm\_word should be complete, taking in two texts and returning a single, normalized LCS value.

Well done!

Also refer to this implement, using enumerate:

def lcs\_norm\_word(answer\_text, source\_text):

a\_words = answer\_text.split()

s\_words = source\_text.split()

matrix = np.zeros((len(s\_words)+1, len(a\_words)+1))

for i, s in enumerate(s\_words):

for j, a in enumerate(a\_words):

if a == s:

matrix[i+1,j+1] = matrix[i,j] + 1

else:

matrix[i+1,j+1] = max(matrix[i+1,j], matrix[i,j+1])

return matrix[-1, -1] / len(a\_words)

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.

2 features are too few here because features not enough may cause overfitting. I would suggest you select 3 features.

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.

Perfect! 😄

Accuracy: 0.96

Provide an answer to the two model-related questions.

Great analysis for the performance!

## Notebook 3: Cleaning up Resources

Run the code to clean up your final model resources.