**Authors:** Song Malisa Se(301233051), Matheus Teixeira (301236904), Viet Nguyen Hoang (301228010), Yi-Lin “Aaron” Lou (301226659), Yin-Siang Mao (301180968)

Group Assignment – NLP

# Introduction

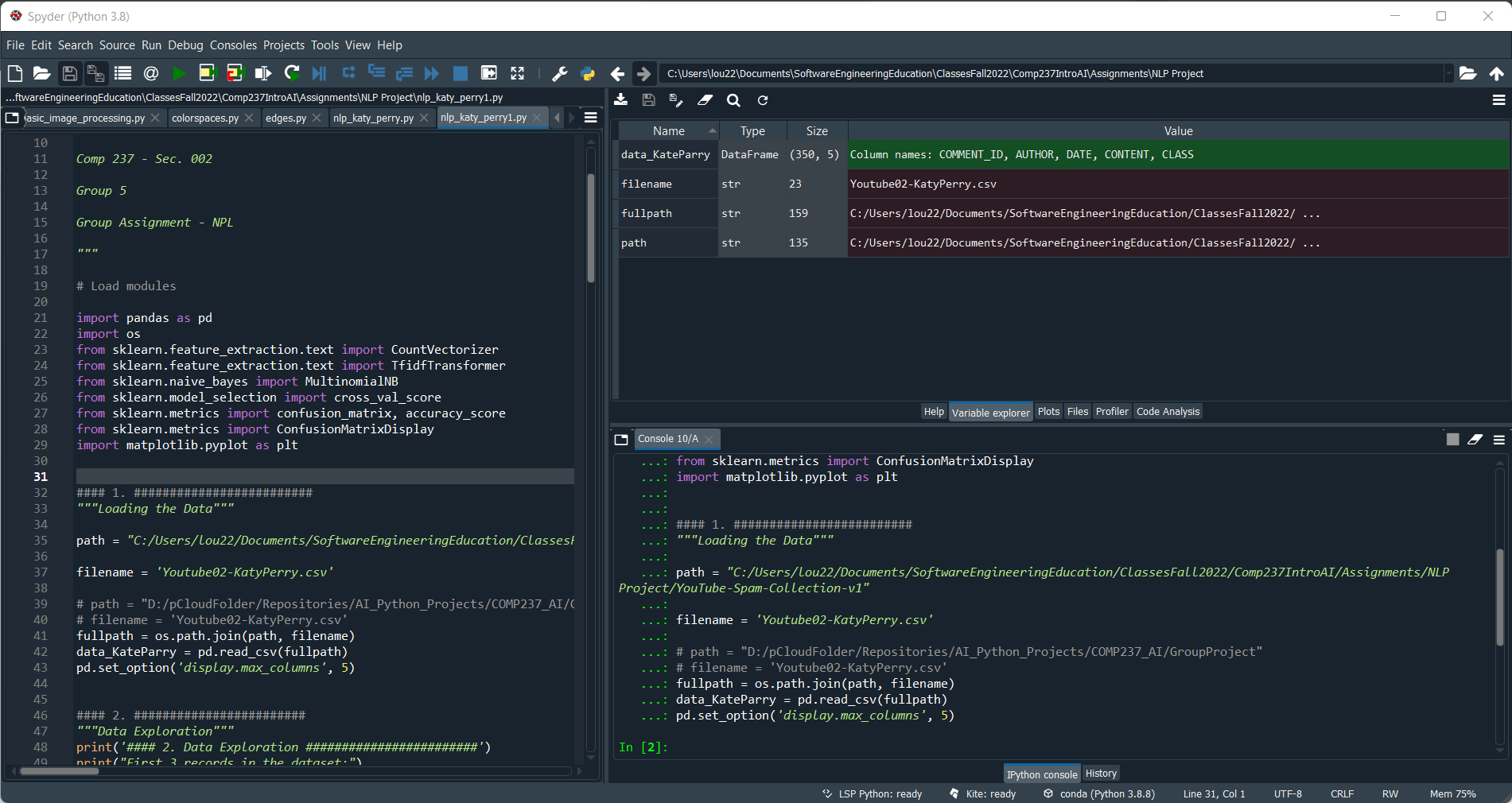
It was asked to create a Naïve-Bayes classifier to do Natural Language Processing on a YouTube Spam Collection Data Set. This collection is available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection>) maintained by the University of California Irvine. The subset used for this experiment was the Data Set related to the comments on Katy Perry's Youtube video available at (<https://www.youtube.com/watch?v=CevxZvSJLk8>). The goal is to create an artificial intelligence that can classify comments as spam or not spam.

# Methodology

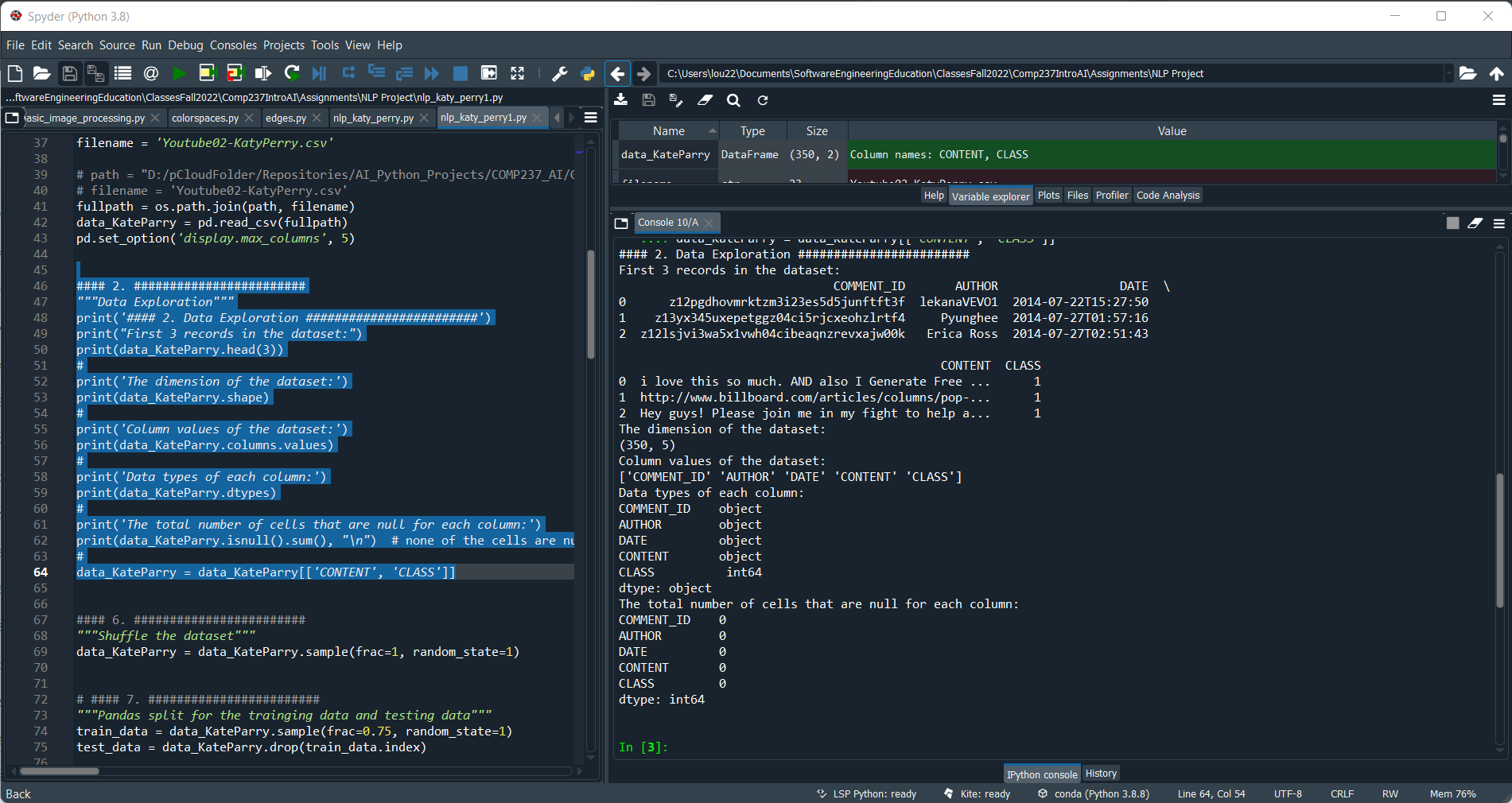
1. Download the .csv file with the comments from the UCI database.
2. Import the file to the python software.
3. Do some data exploration to understand it.
4. Pre-processing the data
   1. Extract the comments and their classifications.
   2. Shuffle data.
   3. Break it on 75% for training and 25% for testing
   4. Use the Sklearn libraries to transform the data in numbers; the Inverse Document Frequency (idf) transformation was used.
   5. Train the classifier
   6. Evaluate the classifier
   7. Test the classifier providing the test set.
   8. Come up with need data to simulate use in production

# The experiment

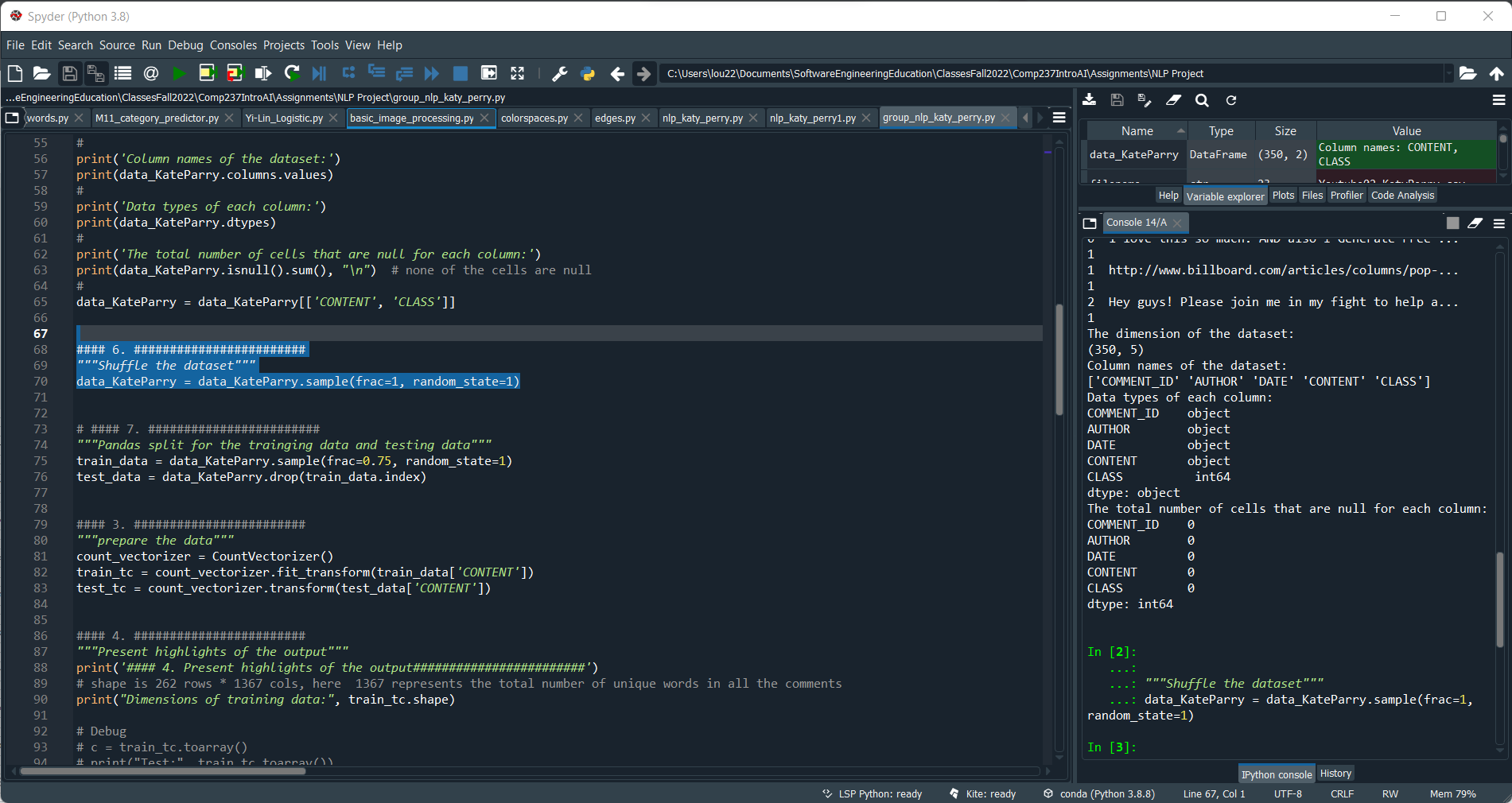
## Requirement 1 - Load the data



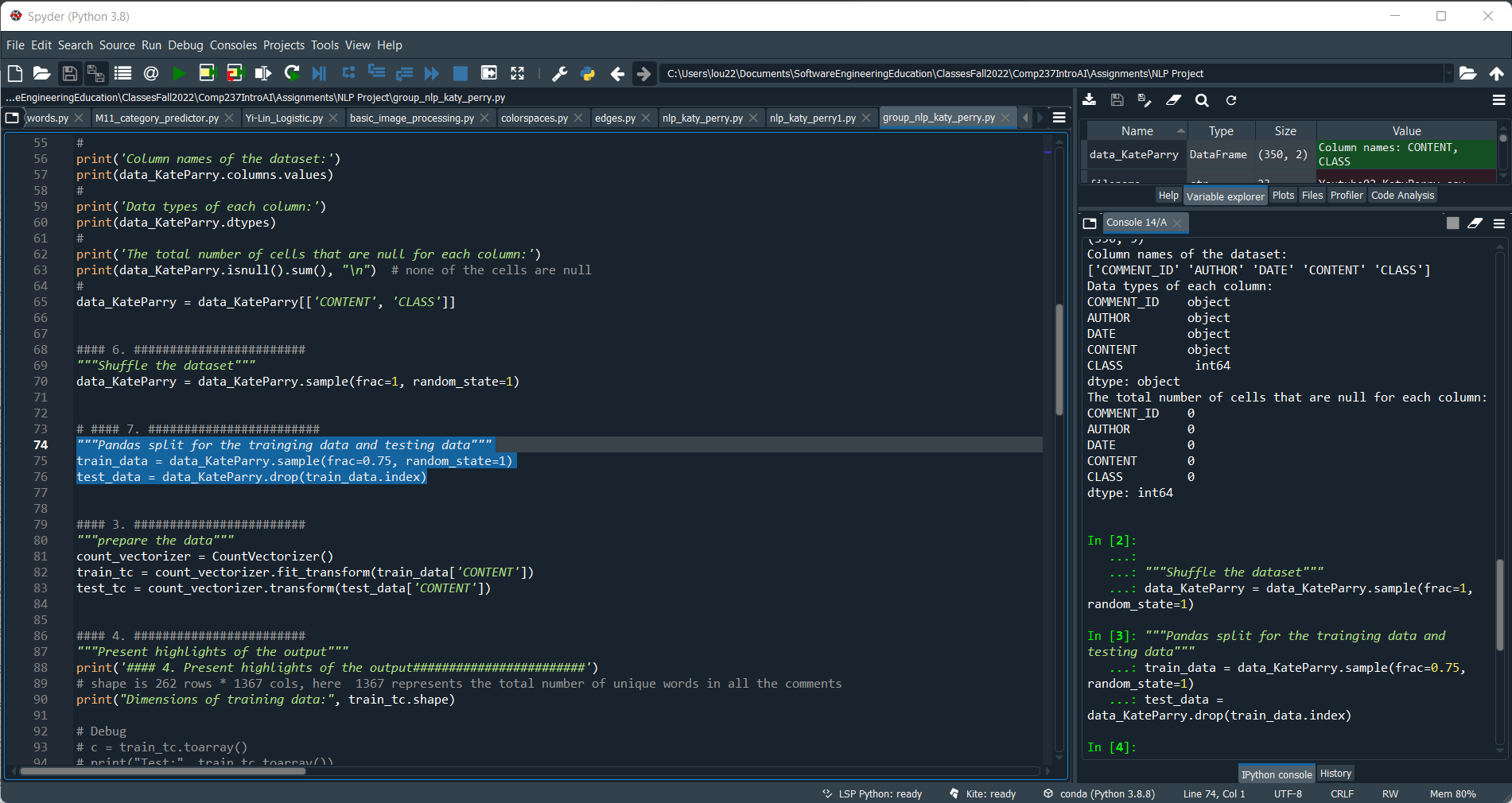
## Requirement 2 - Data explanation



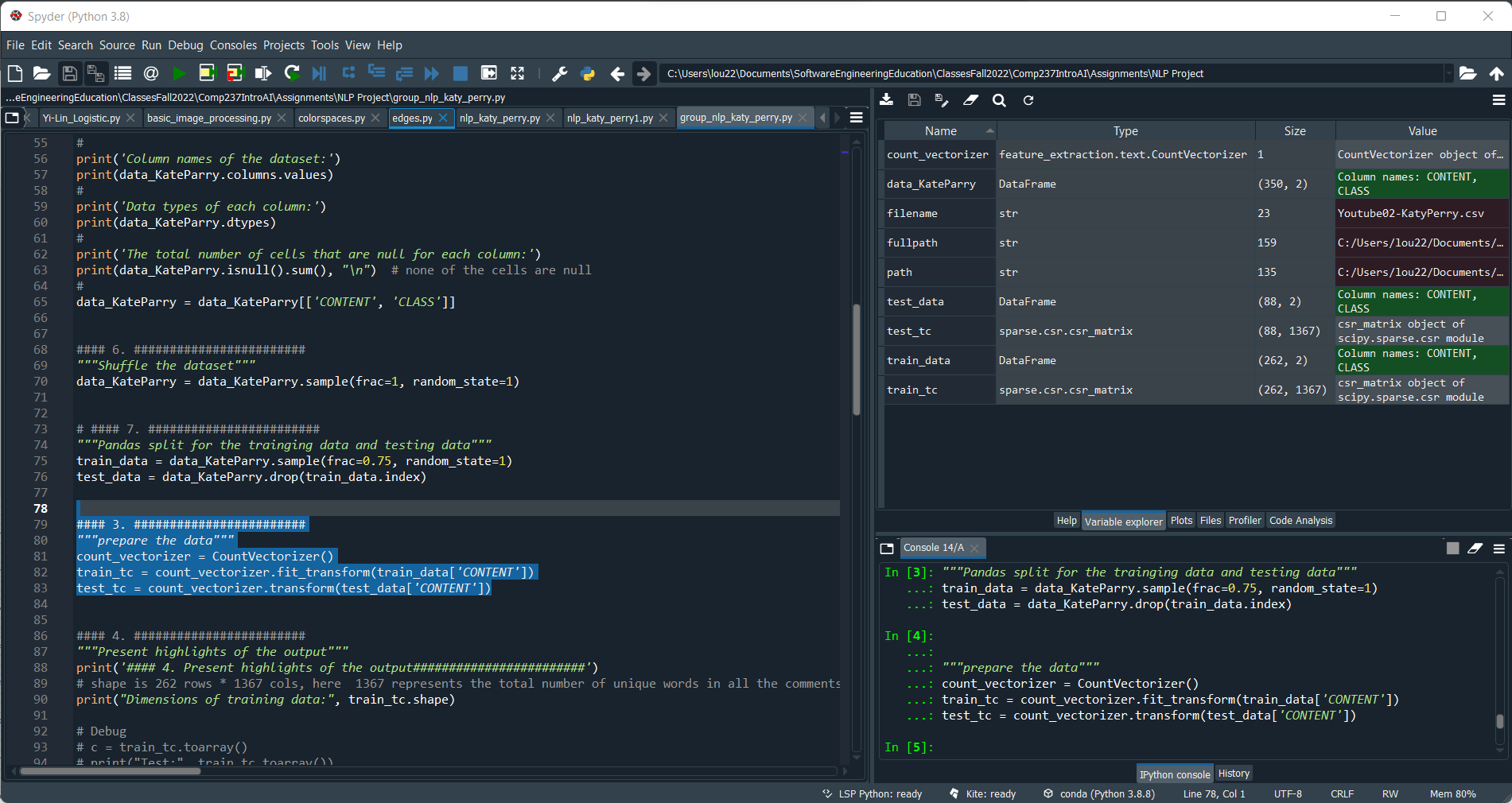
## Requirements 6 - Shuffle the dataset



## Requirements 7 - Pandas split for the trainging data and testing data

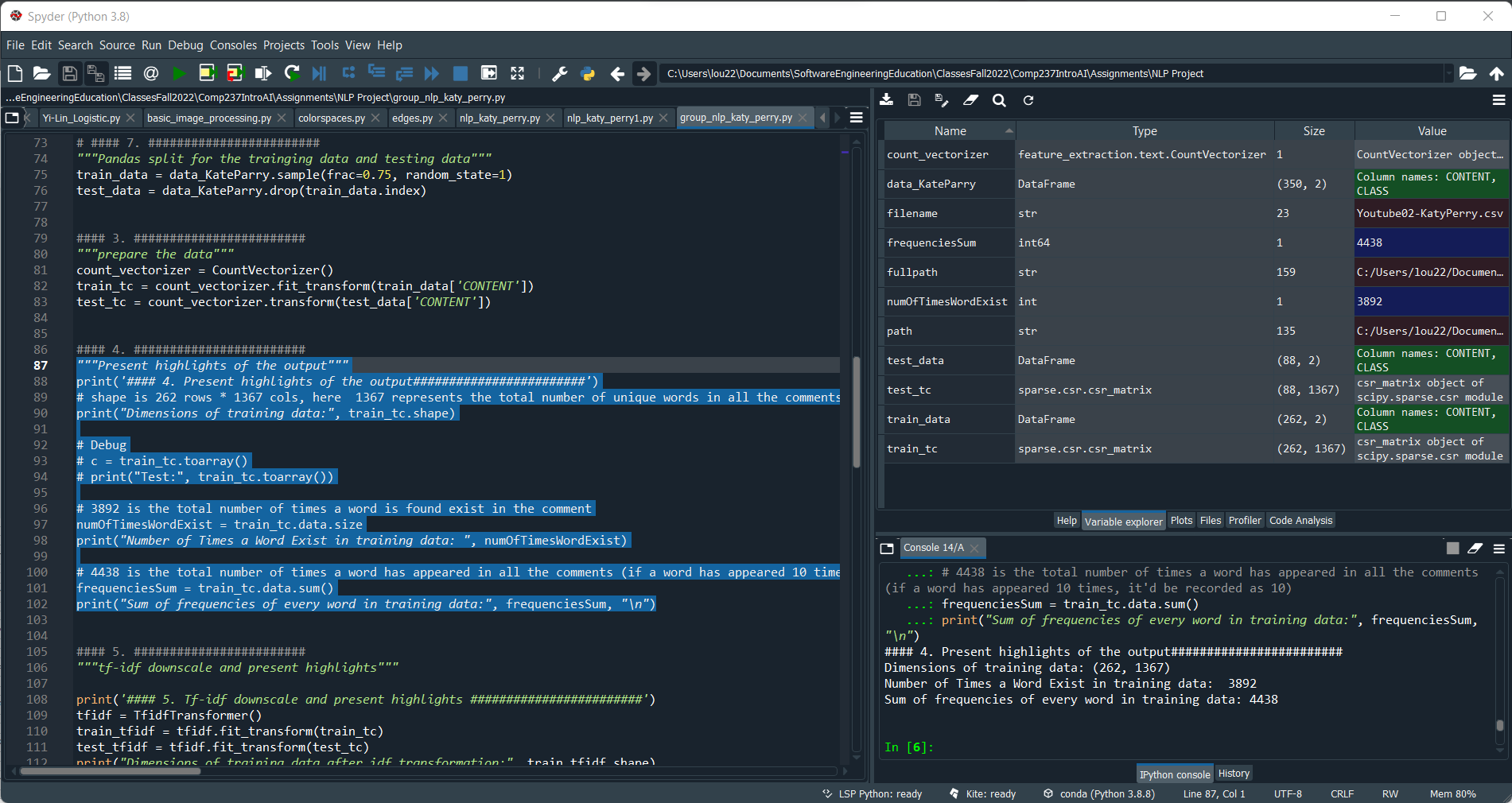


## Requirements 3 - prepare the data

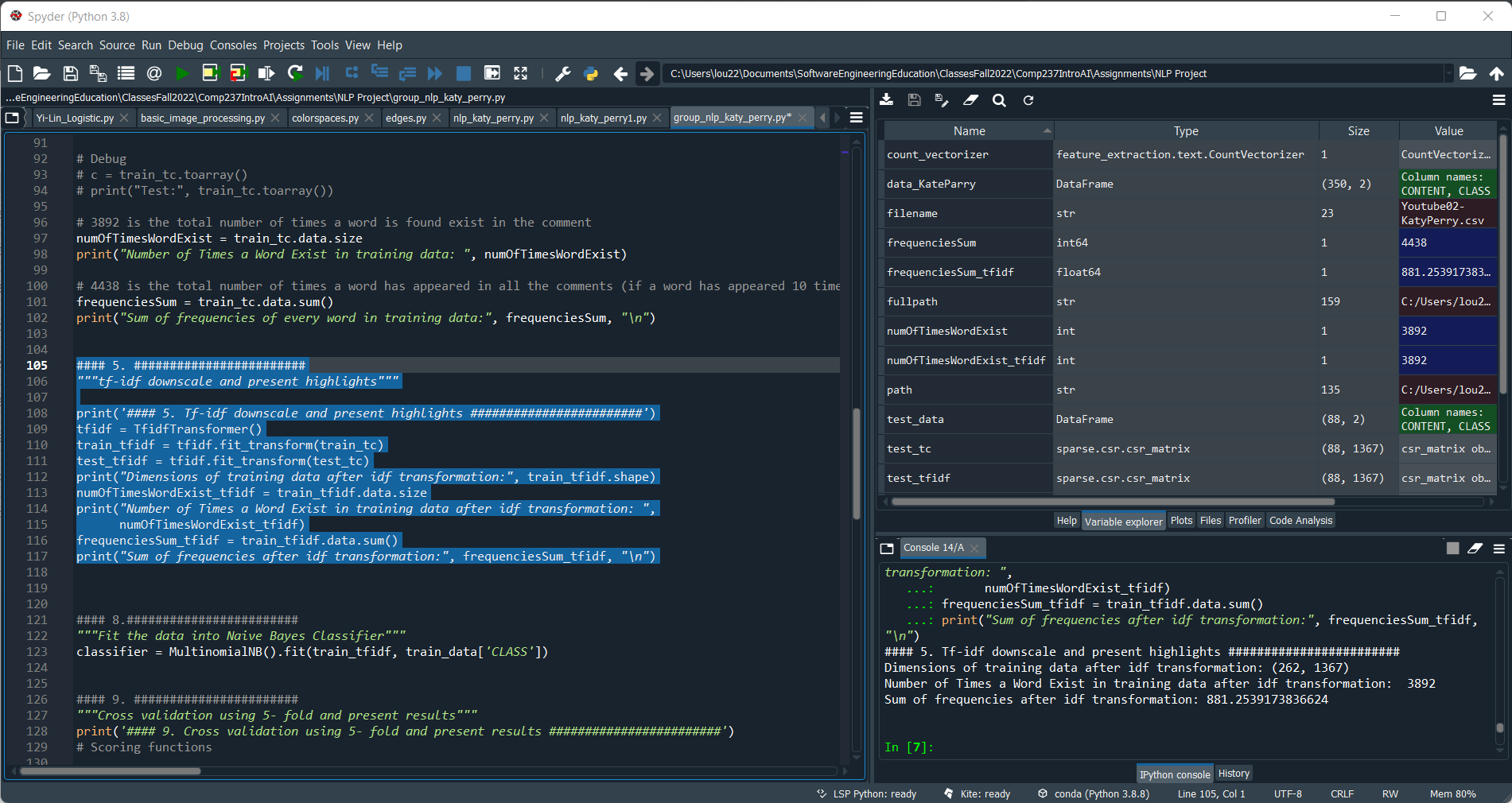


## Requirements 4 and 5 - Highlights on the transformed data

Requirement 4:



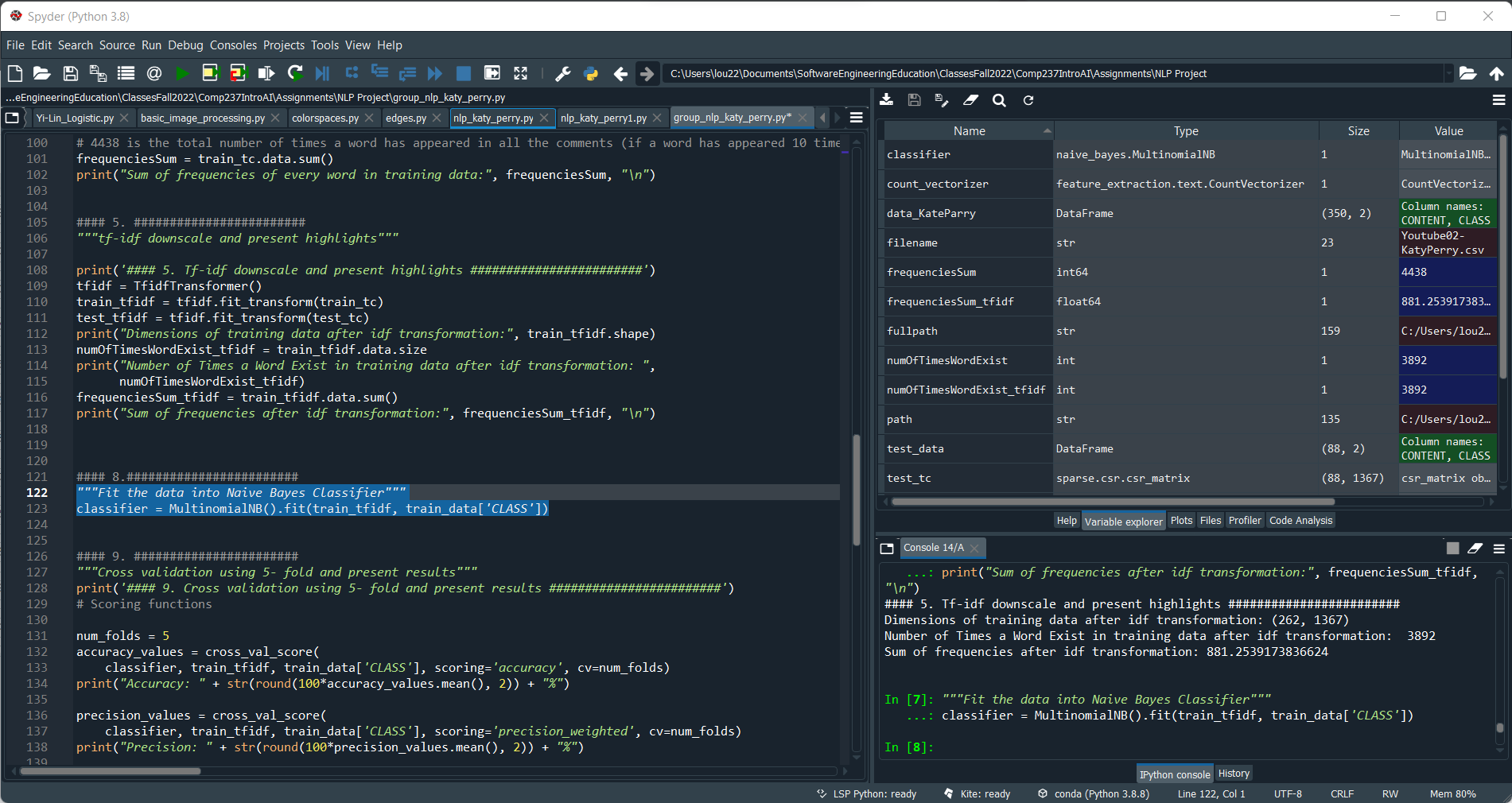
Requirement 5:



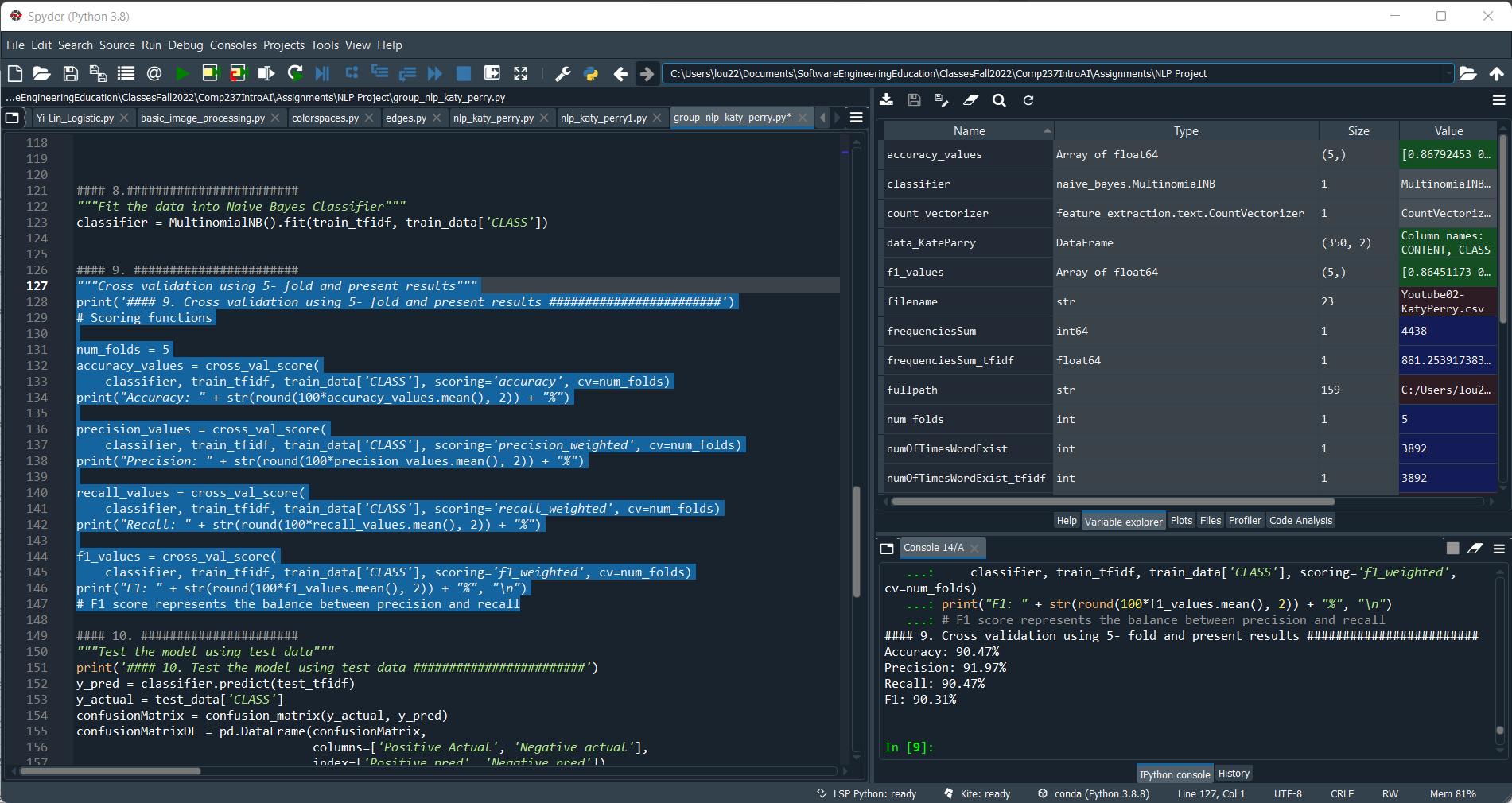
Comparison:

| **Highlight** | **Term Frequency (Req. 4)** | **Inverse Document Frequency (Req. 5)** |
| --- | --- | --- |
| # Rows of training data  (# of different documents) | 262 | 262 |
| # Columns of training data  (# of unique words) | 1367 | 1367 |
| Sum of all different words that shows up on each document | 3892 | 3892 |
| Sum of all frequencies of each unique word | 4438 | 881.25 |

## Requirement 8 - Fit the data into Naive Bayes Classifier

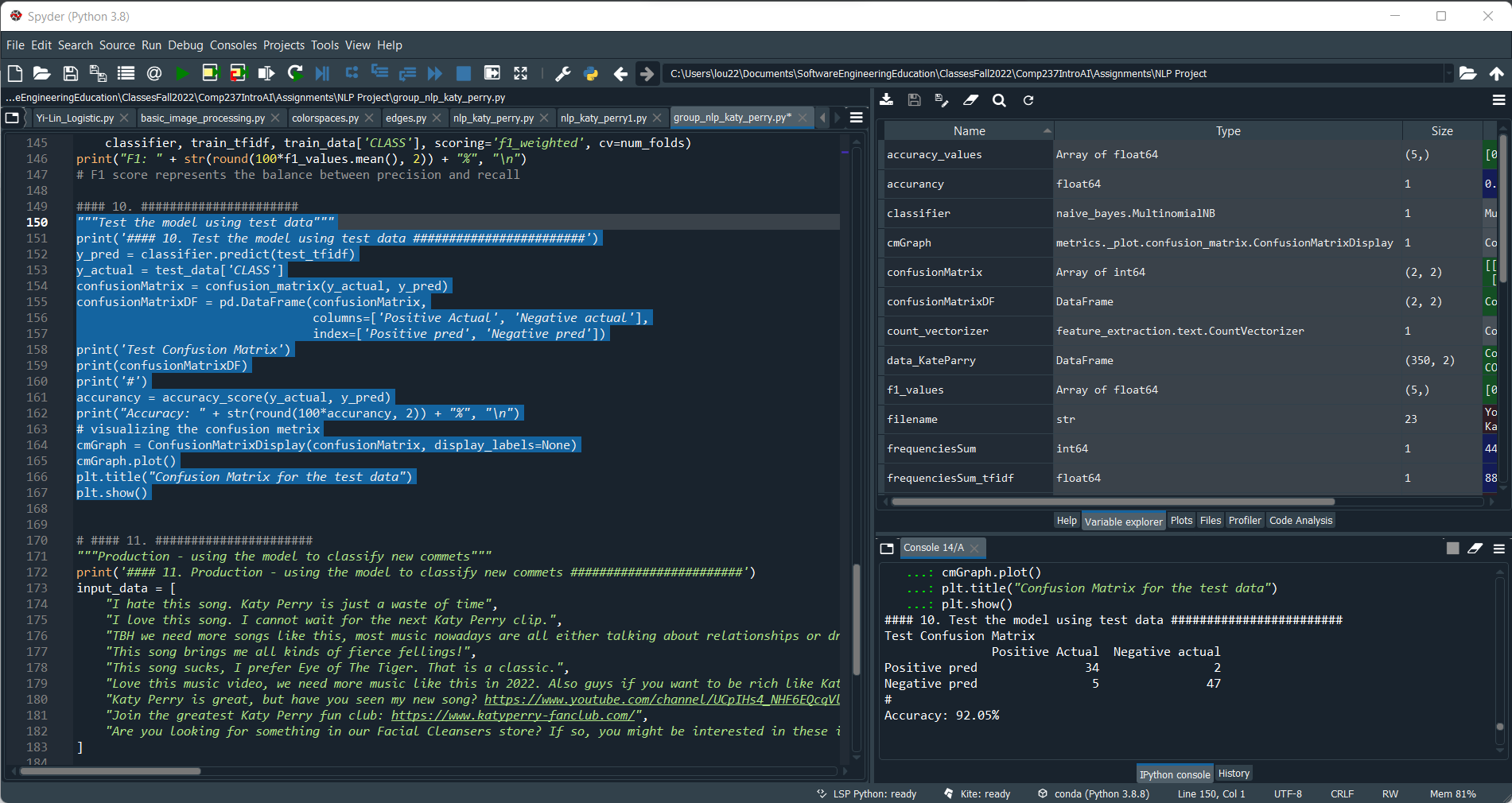


## Requirement 9 - Cross-Validation (5-fold) and scoring

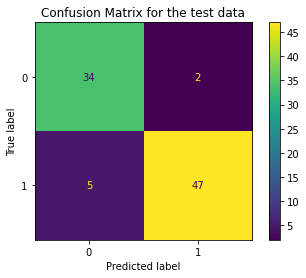


| **Score** | **Value** |
| --- | --- |
| Accuracy | 90.47% |
| Precision | 91.97% |
| Recall | 90.47% |
| F1 | 90.31% |

## Requirement 10 - Confusion matrix, using test data



Confusion Matrix visualization using matplotlib:



True positive: 34

False negative: 2

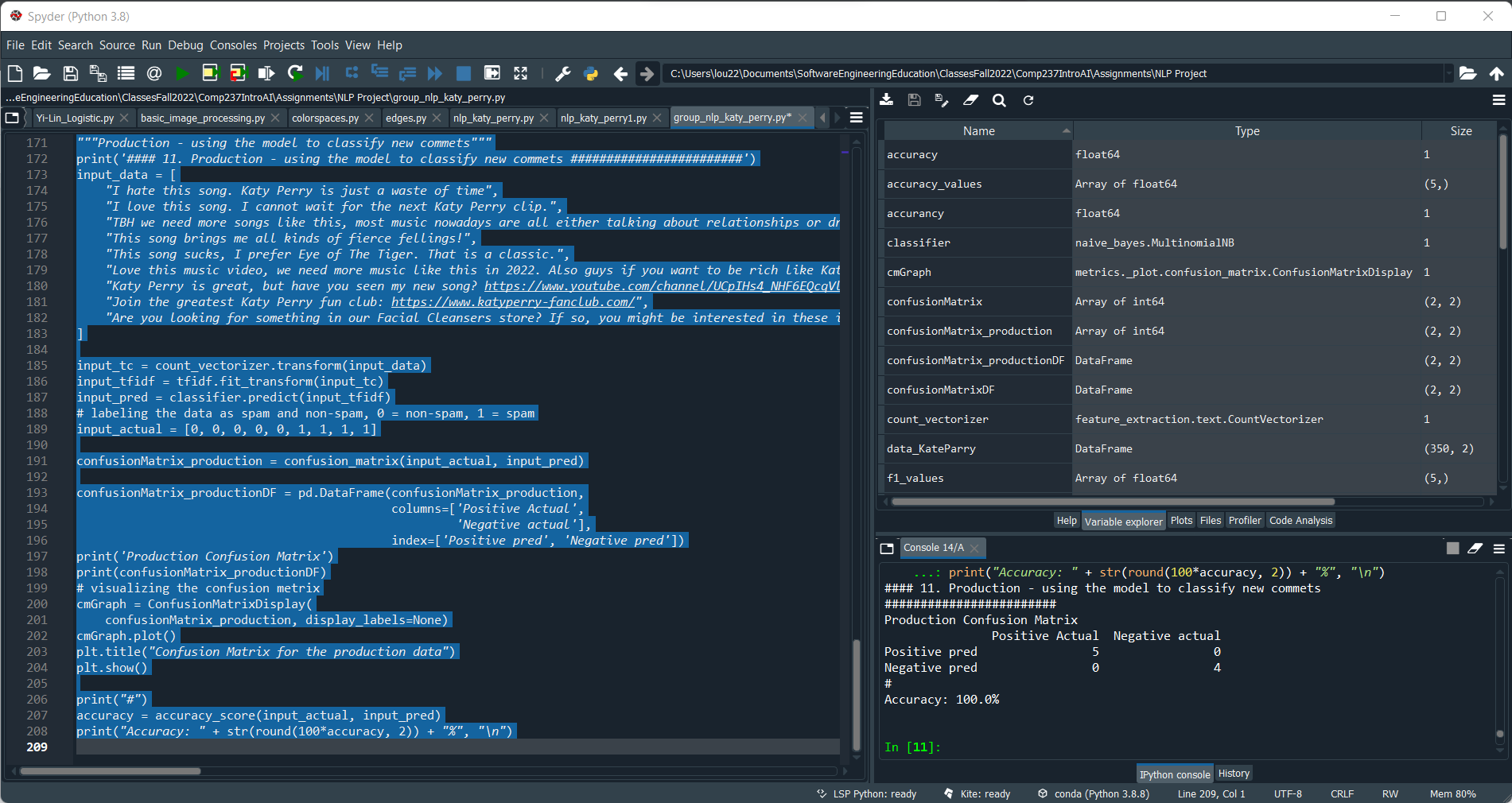
False Positive: 5

True negative: 47

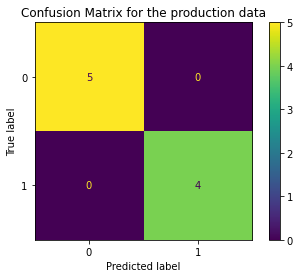
|  |  |  |
| --- | --- | --- |
|  | **Positive Actual** | **Negative Actual** |
| Positive Predicted | 34 | 2 |
| Negative Predicted | 5 | 47 |

**Accuracy:** 92.05%

## Requirement 11 - Production Test



Confusion Matrix visualization using matplotlib:



True positive: 5

False negative: 0

False Positive: 0

True negative: 4

|  |  |  |
| --- | --- | --- |
|  | **Positive Actual** | **Negative Actual** |
| Positive Predicted | 5 | 0 |
| Negative Predicted | 0 | 4 |

**Accuracy:** 100%

# Analysis

From requirements 4 and 5 we can notice that using the Term Frequency - Inverse Document Frequency does not change the number of documents, nor the number of unique words, but it downscales the data reducing the total frequency. This algorithm is also important because it takes into consideration the importance of each word on each document. This is achieved by considering the weight of TF and IDF. A high weight in TF-IDF means that the term frequency is high in the given document and a low document frequency of the given term in all the other documents. This allowed us to rule out most of the common words while obtaining high weight, unique words and increase the model’s efficiency and accuracy.

From the cross-validation test, we can notice that the classifier model is very good; all its scores were higher than 90%; in fact, for the test data the accuracy was even higher, and the confusion matrix supports the conclusion that the model is good.

For the production test, with only a few pieces of data, the model was able to classify everything correctly; however, the results could vary as number of data input increase.