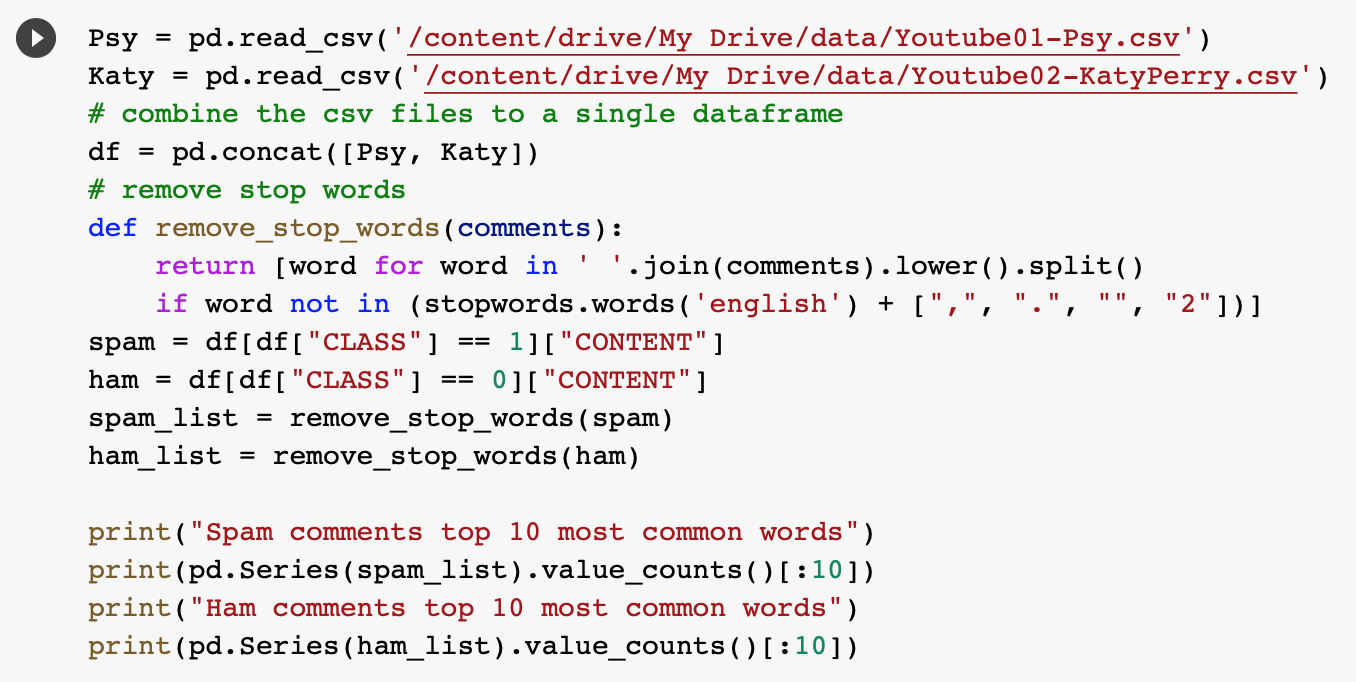
1.

The environment I used to build, train, test, evaluate my classifier is on Google Colab using python.

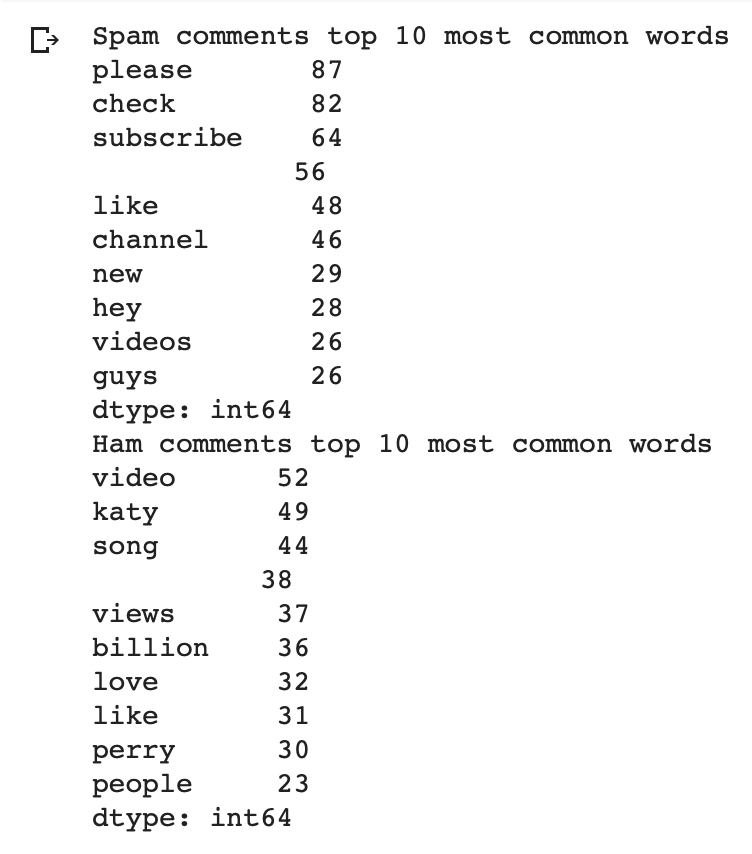
a.)

Step i.)

* I filter for the ten most frequent words of the spam and ham (non-spam) classes from the ‘CONTENT’ attribute of the two csv files; Youtube01-Psy.csv and Youtube02-KatyPerry.csv.



* This output will be useful for my derived attributes ‘spamTerms’ and ‘hamTerms’ in the next step.



Step ii.)

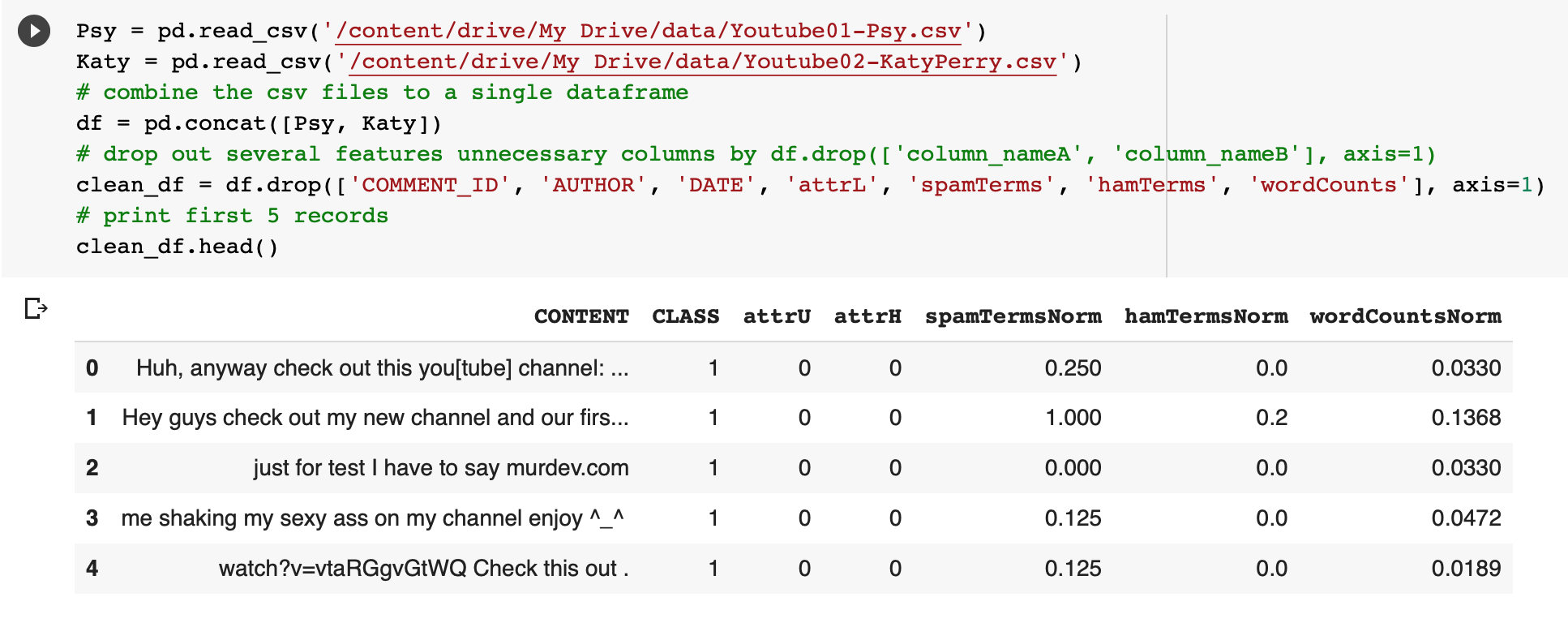
* For attribute construction, original data were transformed into new easily detected attributes for the classification algorithm to improve the predictive accuracy.
  + I used Microsoft Excel as my tool to construct the derived attributes by using the following Excel functions in blue, on records in Youtube01-Psy.csv and Youtube02-KatyPerry.csv:
* **attrL** = LEN(D2)
* **attrU** = IF(SUM(COUNTIF(D2,{"\*http://\*","\*https://\*"})),1,0)
* **attrH** = IF(SUMPRODUCT(--ISNUMBER(SEARCH({"&",";"},D2)))>0,1,0)
  + I will just filter for the symbols ‘&’ and ‘;’ which are commonly found in weblink.
* **spamTerms** = IF(ISNUMBER(SEARCH("check",D2)),1,0)+IF(ISNUMBER(SEARCH("please",D2)),1,0)+IF(ISNUMBER(SEARCH("like",D2)),1,0)+IF(ISNUMBER(SEARCH("videos",D2)),1,0)+IF(ISNUMBER(SEARCH("subscribe",D2)),1,0)+IF(ISNUMBER(SEARCH("channel",D2)),1,0)+IF(ISNUMBER(SEARCH("new",D2)),1,0)+IF(ISNUMBER(SEARCH("guys",D2)),1,0)+IF(ISNUMBER(SEARCH("hey",D2)),1,0)
  + This attribute makes use of step i.) results and count the number of the top ten most frequent spam comments’ words from the comments in ‘CONTENT’ attribute of column D. This is for the purpose of creating the ‘spamTermsNorm’ attribute below.
* **hamTerms** =

IF(ISNUMBER(SEARCH("video",D2)),1,0)+IF(ISNUMBER(SEARCH("katy",D2)),1,0)+IF(ISNUMBER(SEARCH("song",D2)),1,0)+IF(ISNUMBER(SEARCH("views",D2)),1,0)+IF(ISNUMBER(SEARCH("billion",D2)),1,0)+IF(ISNUMBER(SEARCH("love",D2)),1,0)+IF(ISNUMBER(SEARCH("like",D2)),1,0)+IF(ISNUMBER(SEARCH("perry",D2)),1,0)+IF(ISNUMBER(SEARCH("people",D2)),1,0)

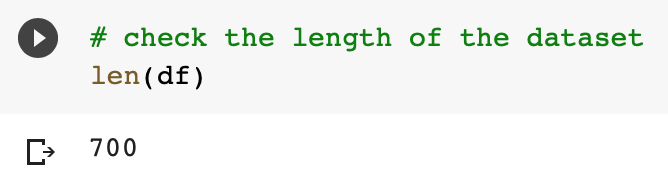
* + This attribute makes use of step i.) results and count the number of the top ten most frequent non-spam (ham) comments’ words from the comments in ‘CONTENT’ attribute of column D. This is for the purpose of creating the ‘hamTermsNorm’ attribute below.
* **wordCounts** = LEN(TRIM(D2))-LEN(SUBSTITUTE(D2," ",""))+1
  + Count the number of words in the comment. I created this attribute because I prefer to use this measure instead of attrL which counts the characters.
* **spamTermsNorm** = (I2-MIN($I$2:$I$351))/(MAX($I$2:$I$351)-MIN($I$2:$I$351))
  + I will use this min-max normalized version of the ‘spamTerms’ attribute instead.
* **hamTermsNorm** = (J2-MIN($J$2:$J$351))/(MAX($J$2:$J$351)-MIN($J$2:$J$351))
  + I will use this min-max normalized version of the ‘hamTerms’ attribute instead.
* **wordCountsNorm** = (K2-MIN($K$2:$K$351))/(MAX($K$2:$K$351)-MIN($K$2:$K$351))
  + I will use this min-max normalized version of the ‘wordCounts’ attribute instead.

Step iii.)

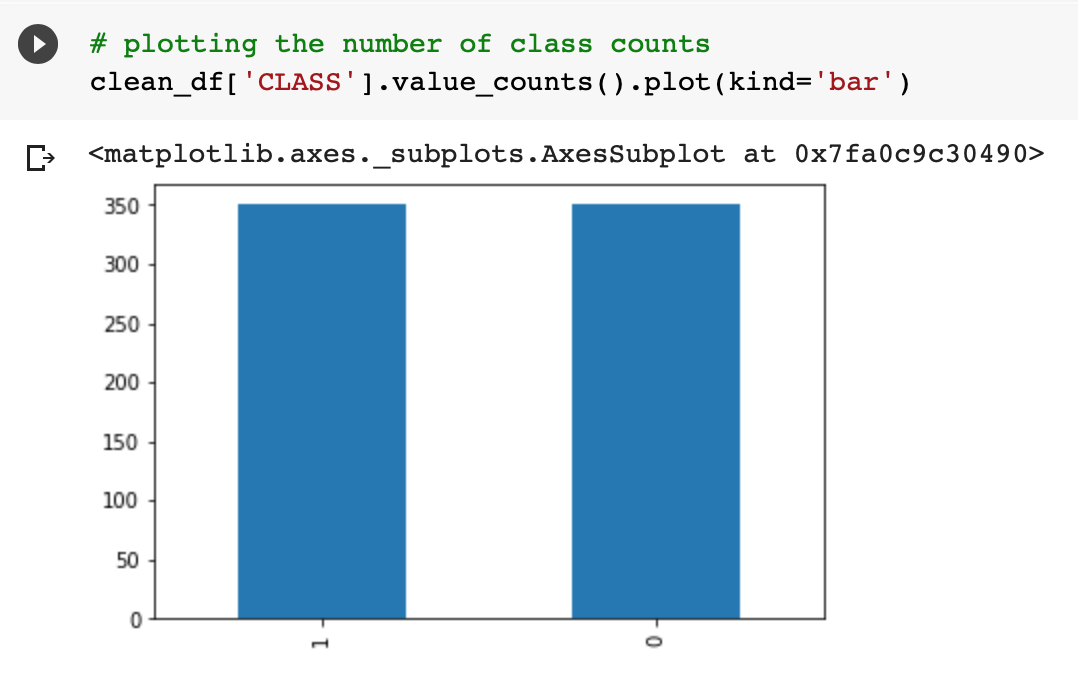
* I dropped some attributes that are not significant to my classifier as described in the previous step as a part of my data preprocessing steps.
  + ‘COMMENT\_ID’, ‘AUTHOR’, and ‘DATE’ are unnecessary columns from the dataset as they are not very informative for our case.
  + ‘attrL’ will be replaced by ‘wordCountsNorm’ attribute instead.
  + ‘spamTerms’, ‘hamTerms’, and ‘wordCounts’ will be replaced by their normalized version; ‘spamTermsNorm’, ‘hamTermsNorm’, and ‘wordCountsNorm’.



* For extra precaution, I checked the number of records of the dataset derived from the two csv.

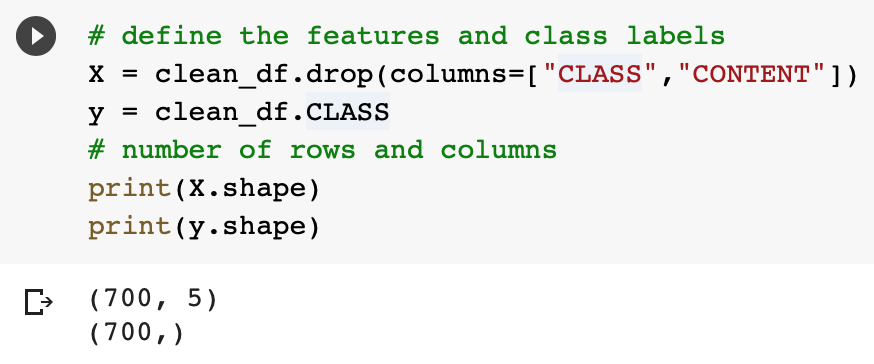


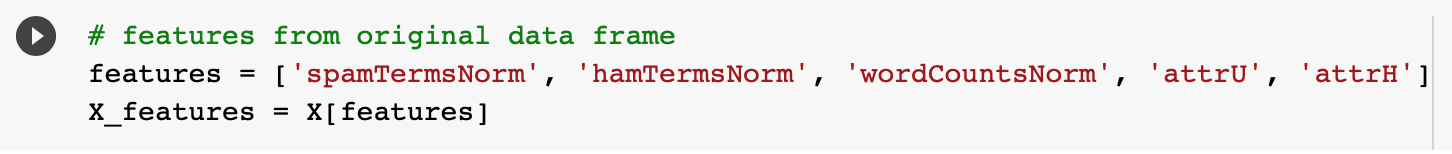
* I also checked that the dataset is in balance with the target classes; number of spam values (1) are the same as the number of ham values (0). Since the dependent attribute is dichotomous (binary), I will use Logistic regression for my classifier model since logistic regression is used more for classification, the result of the model is ​​0 if not spam and 1 if spam, thus indicating the resulting probability.



Step iv.)

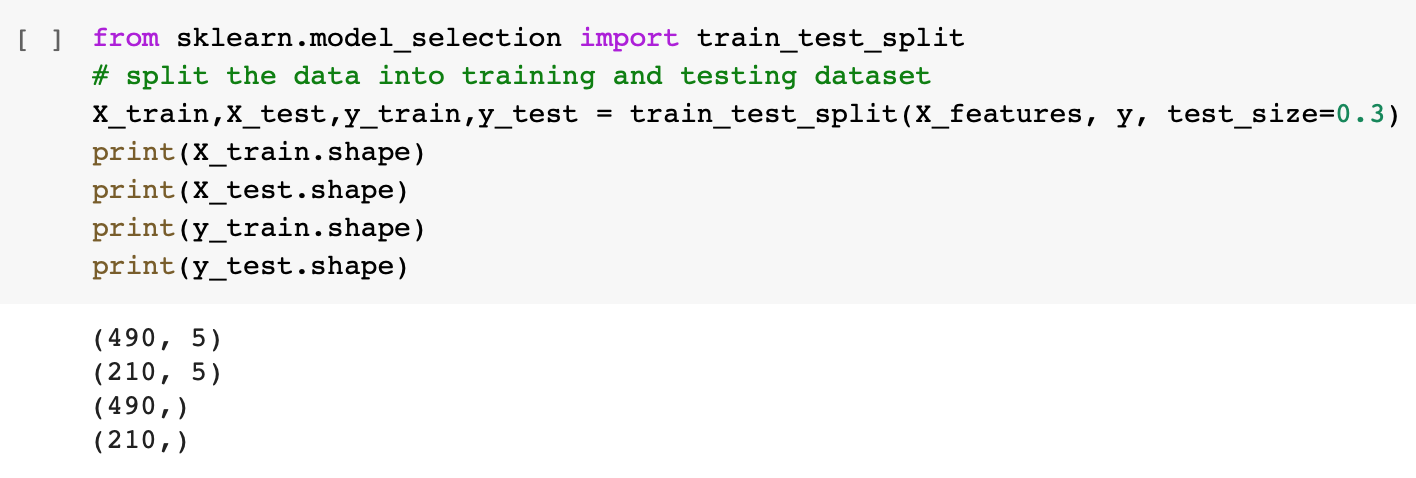
* Then I defined the classes and features from the csv files data frame, that I will be feeding to the classifier in the later steps.

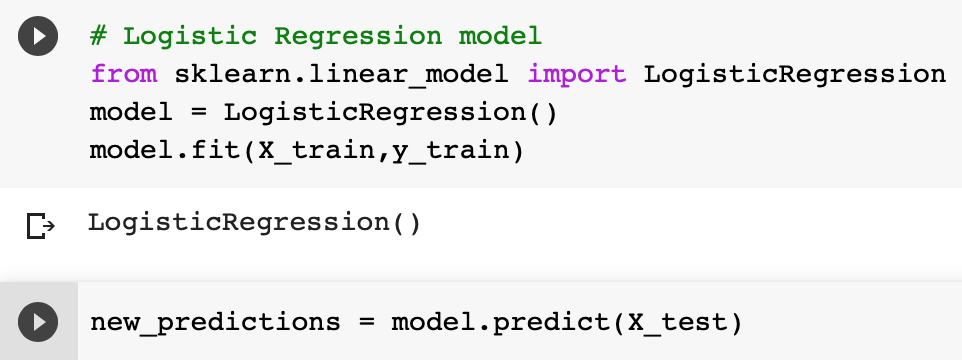




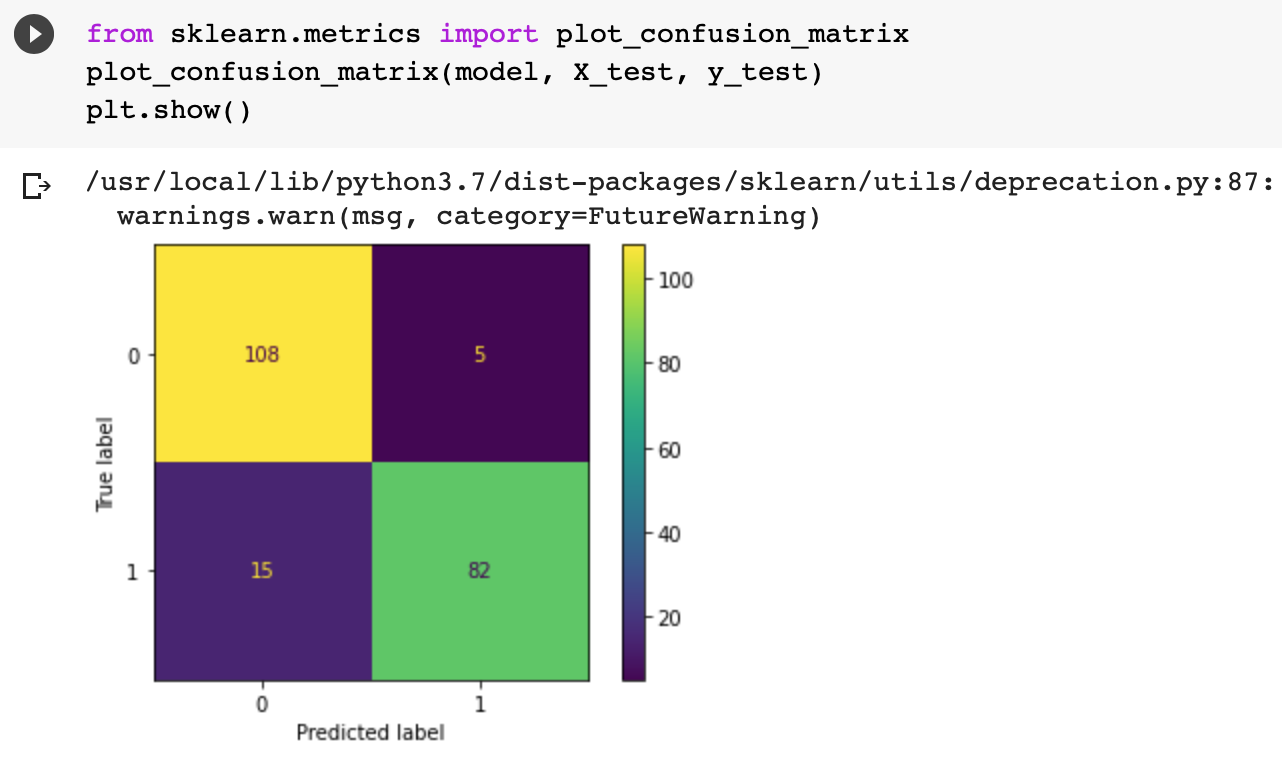
Step v.)

* Then I split the data into training data to later fit the model and testing data to later test it.
* I used logistic regression as my classifier model and train the model by fitting my training data into the model so it can learn from it.
* Then I used the model generated to make predictions with my testing data.

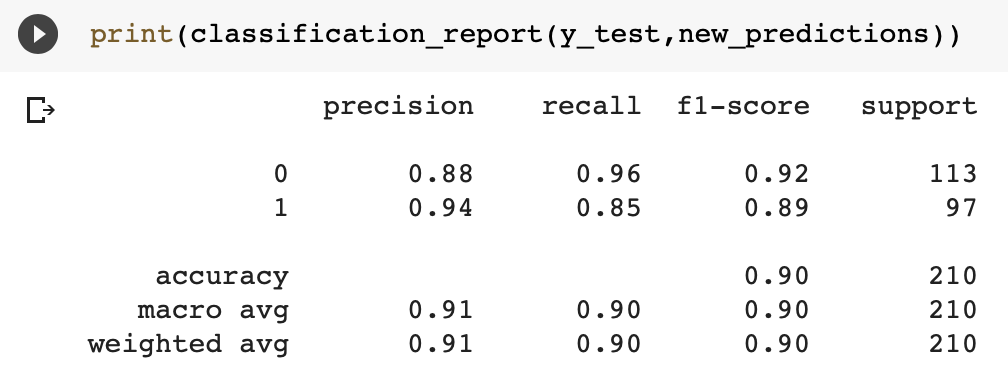




* Then print the confusion matrix, classification report and accuracy of the class predictions.

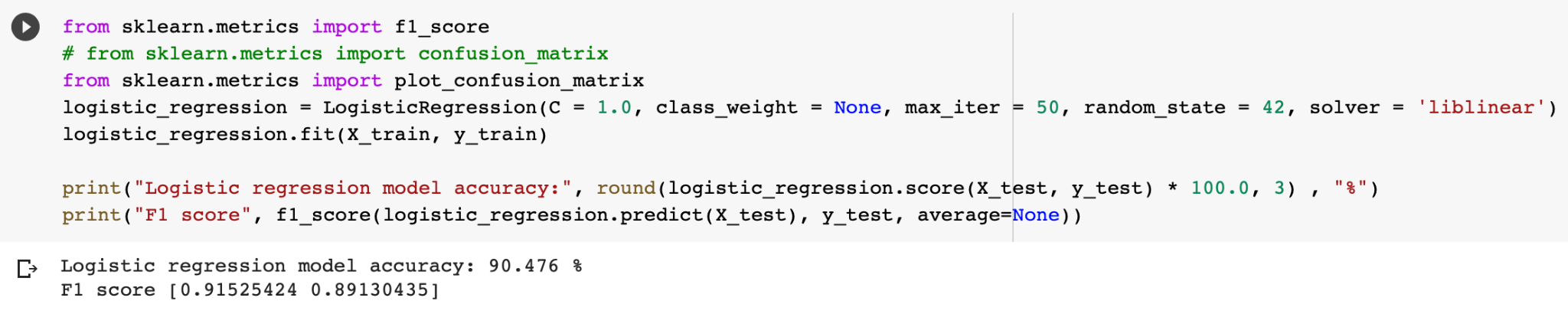
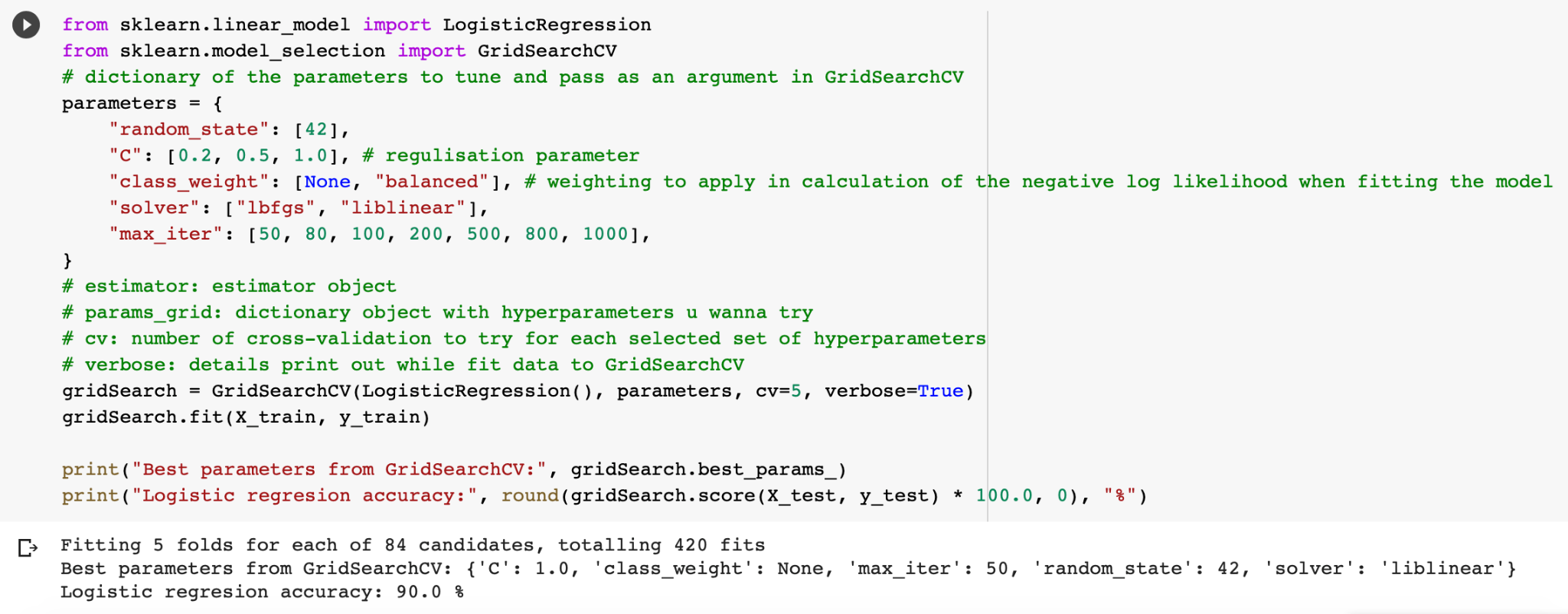


* Training and testing the model on the data lead me to the accuracy score of the classifier.
  + The accuracy score is 0.90 which is good. This classifier has an F1-score of 0.92 and 0.89 so also good. A high F1 score means both precision and recall are high too.



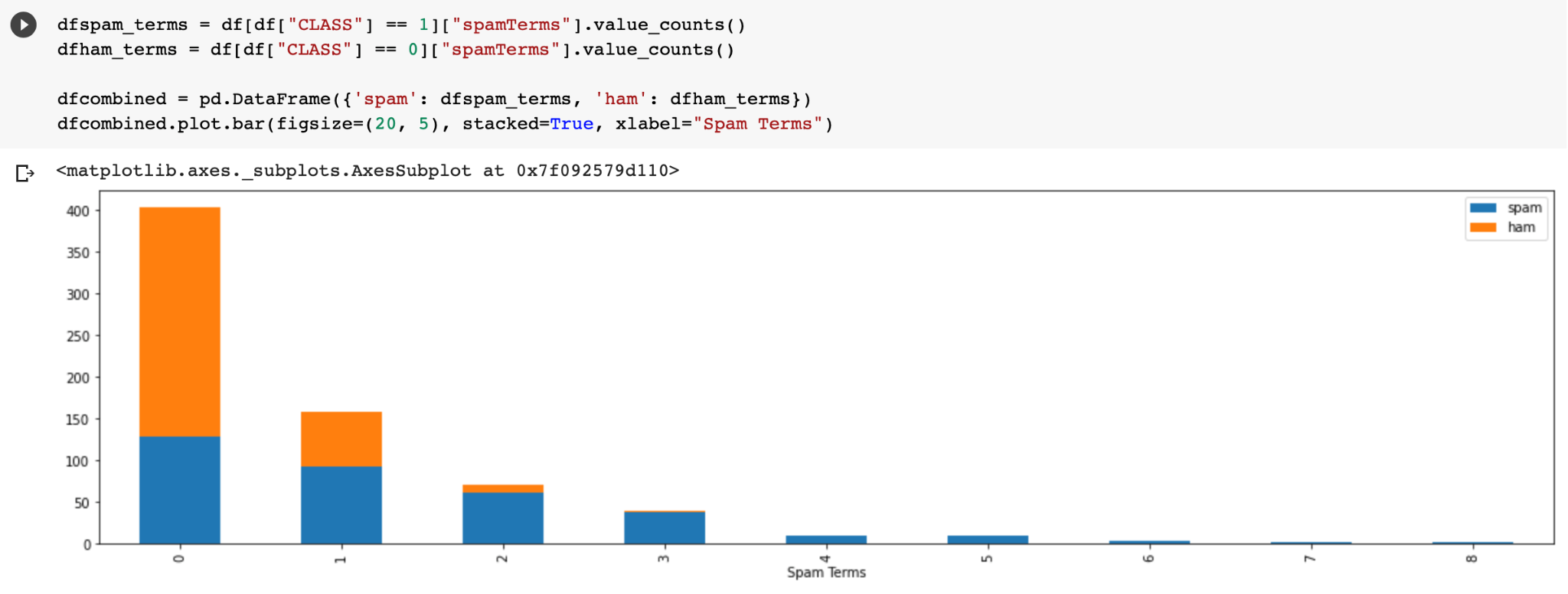
Step vi.)

* I also performed parameter tuning to find the best model training parameters, by Initializing GridSearchCV() object and fitting it with hyperparameters to search through the best parameter values. Then the predictions were made on the same data set again taking into consideration the best parameters found through the fitting process.

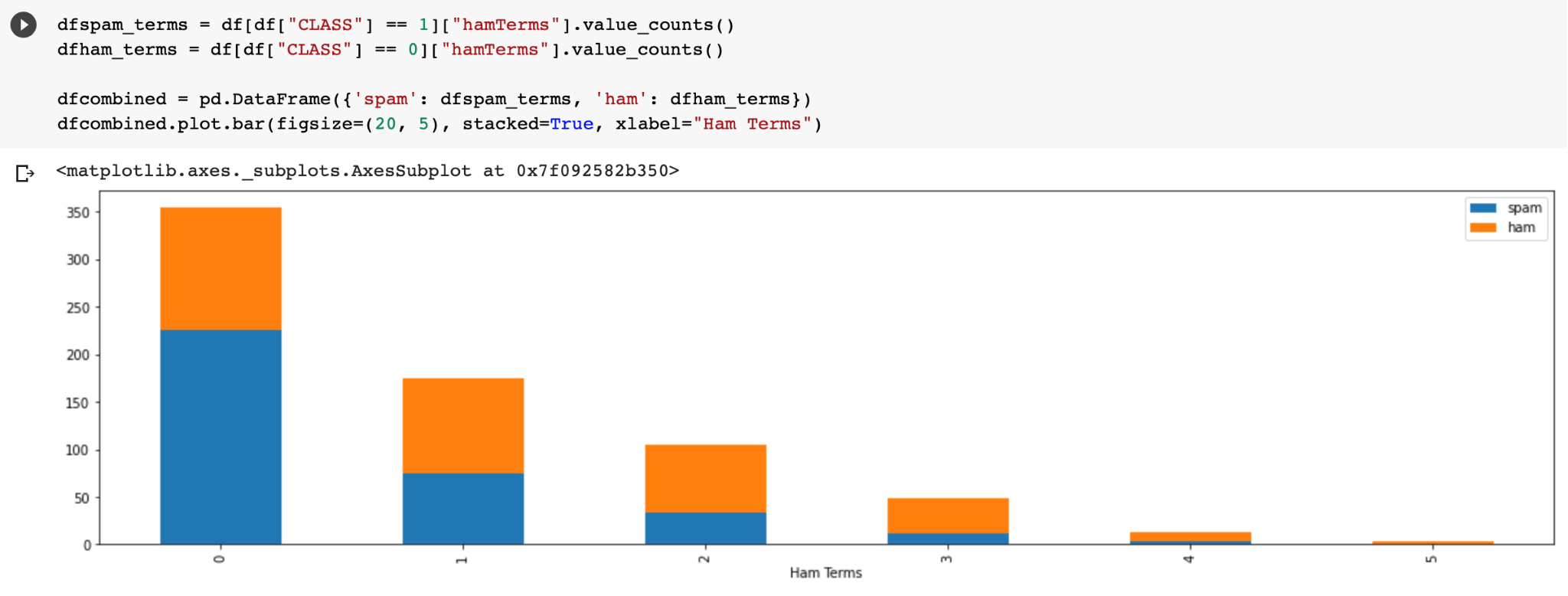


b.)

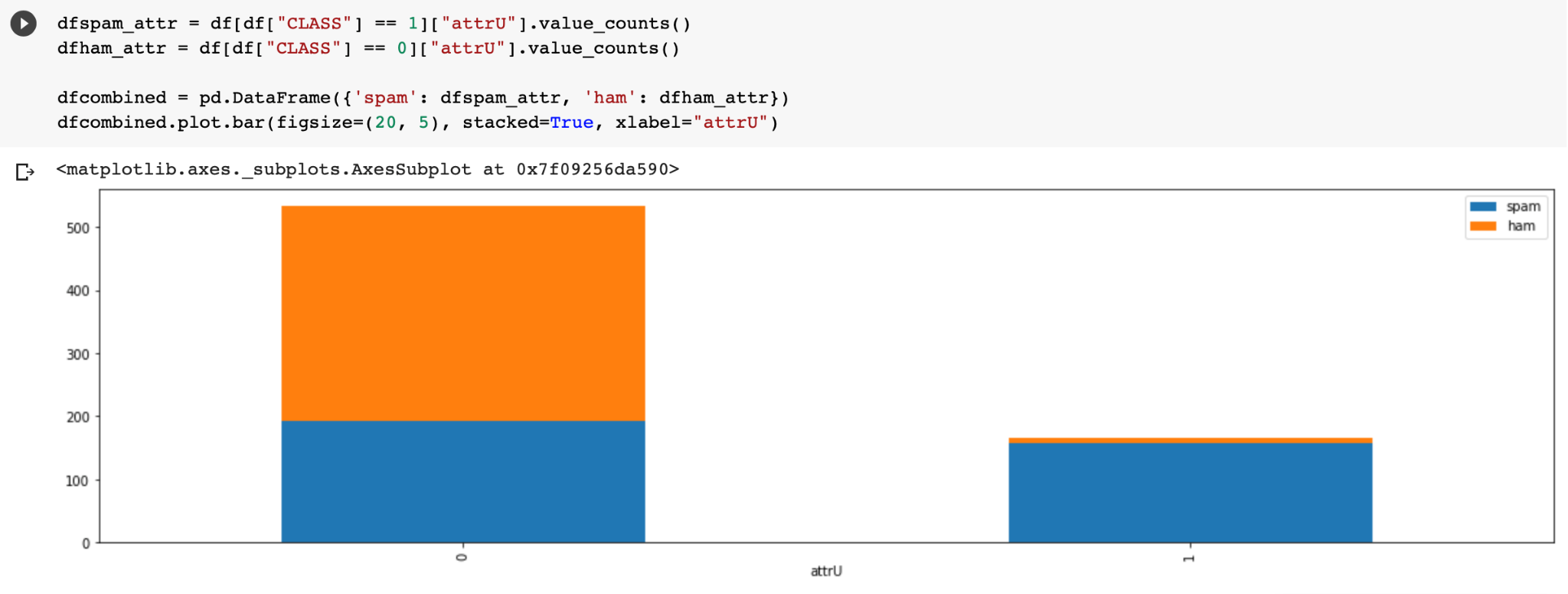
* The derived attributes spamTermsNorm, hamTermsNorm, attrU were the most useful attributes because each of these attributes were able to help the model distinguish between spam and ham classes, partly because a considerable portion of data do reflect these attributes’ properties. (spamTermsNorm and hamTermsNorm are normalized versions of spamTerms and hamTerms.)
  + Most of the data that are classified as spam had those http:// or https:// strings and contains those 10 most common spam keywords.
  + Most of the data that are classified as non-spam do contain those 10 most common hams (non-spam) keywords.
* While attrH and wordCountsNorm help in increasing the accuracy for a bit because not a large portion of the data in the spam and ham classes reflect these attributes’ properties compared to the previously discussed attributes. (wordCountsNorm are normalized versions of wordCounts.)
* Below diagrams visualized the distribution of the original classified data that reflect the chosen attributes’ properties:
  + A relatively small portion of ham comments has 1 to 3 of these top frequent spam terms, as compared to the spam comments where more than 2/3 of the spam comments have these top frequent spam terms.



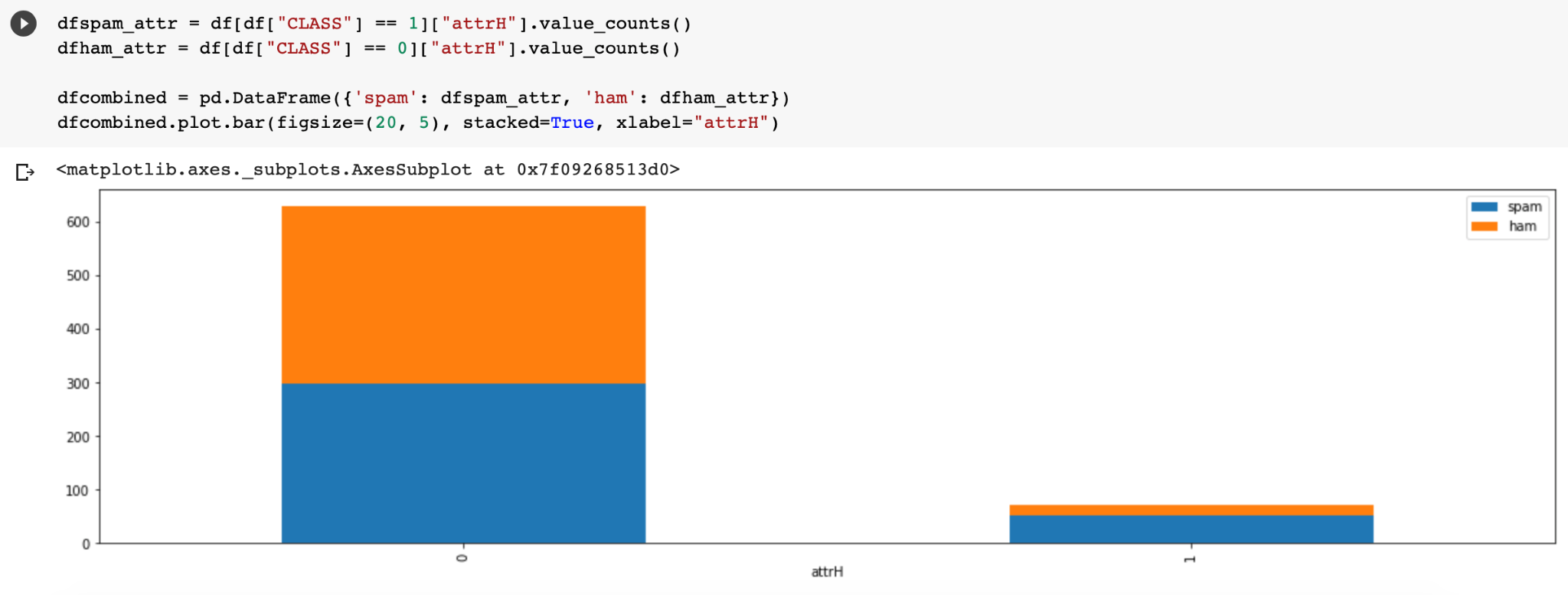
* + A relatively small portion of spam comments has 1 to 4 of these top frequent ham (non-spam) terms, as compared to the ham comments where about 2/3 of the ham comments have these top 10 frequent ham terms.



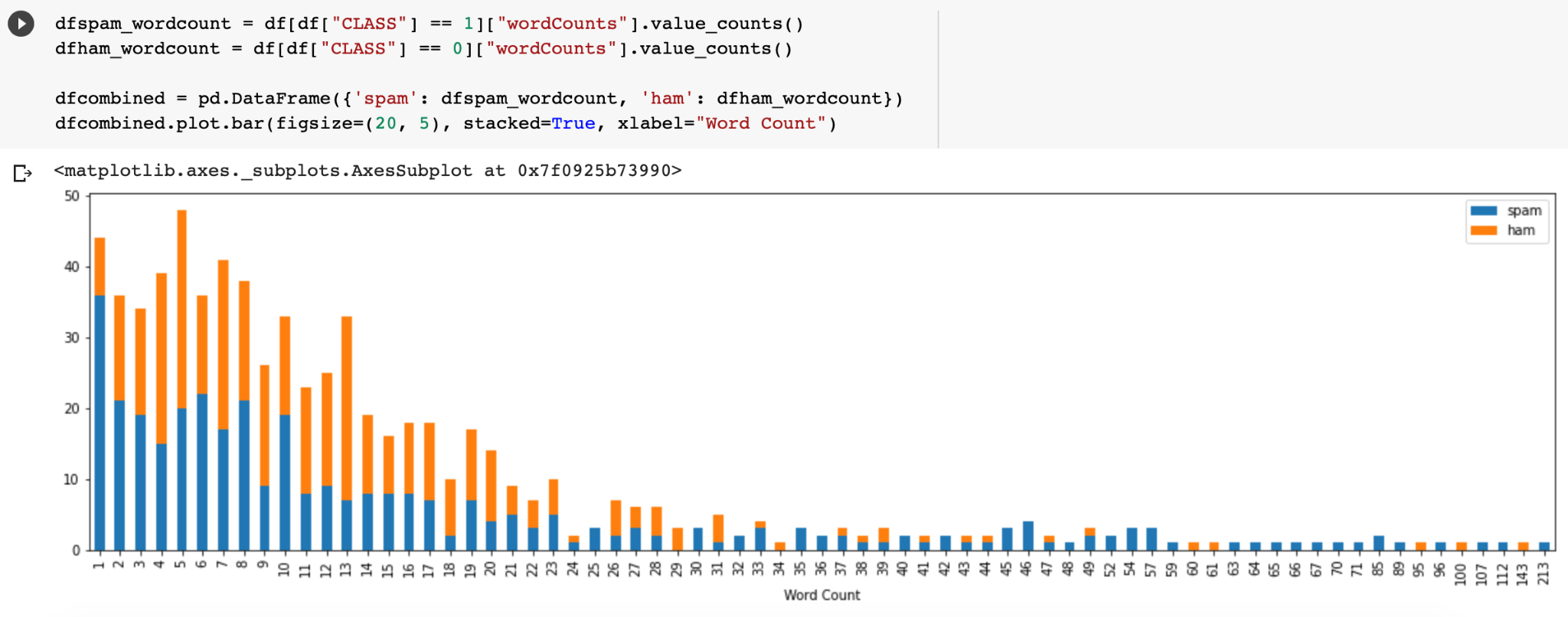
* + About half of the spam comments meet ‘attrU' attributes criteria of containing the strings ‘http://’ and ‘https://’. While an extremely small number of ham comments contain these strings.



* + Although only a small portion of the data contain the ‘& and ‘;’ symbol, but the fraction of spam comments that met these criteria is significantly more than the fraction of the ham comments.

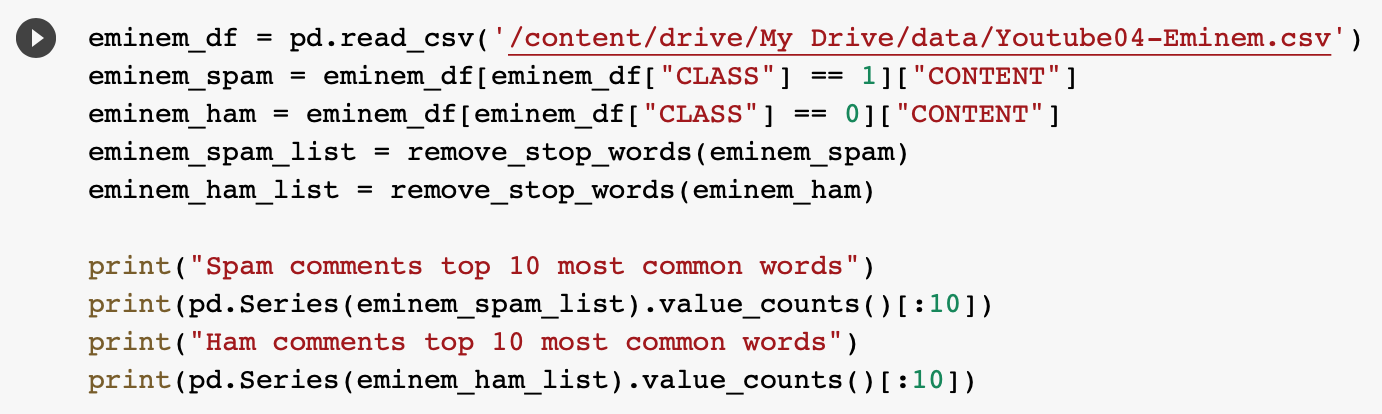


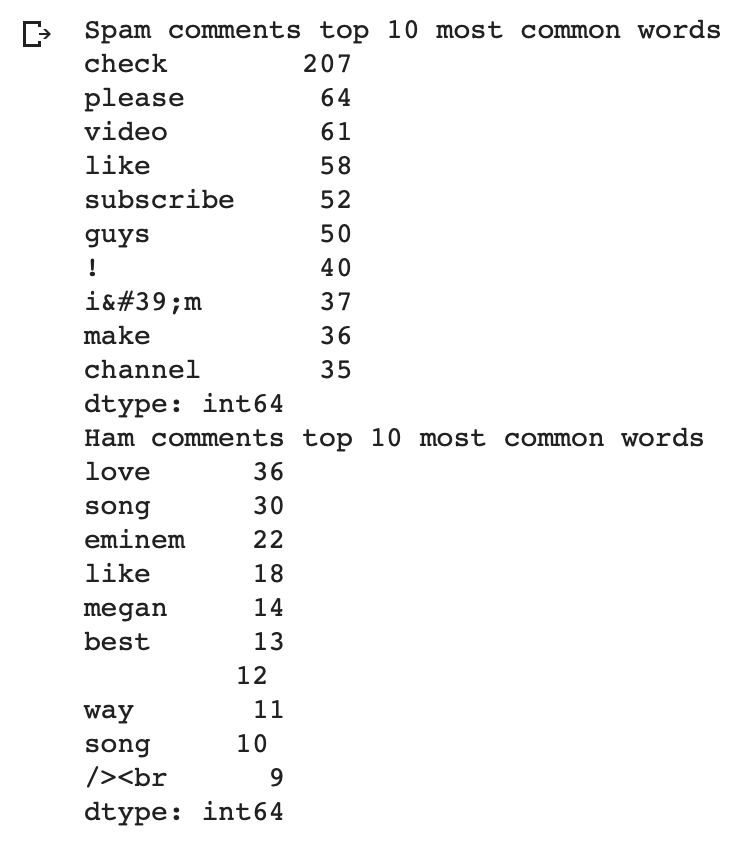
* + Although spam comments do occupy more word counts, this is not much of a significant feature.

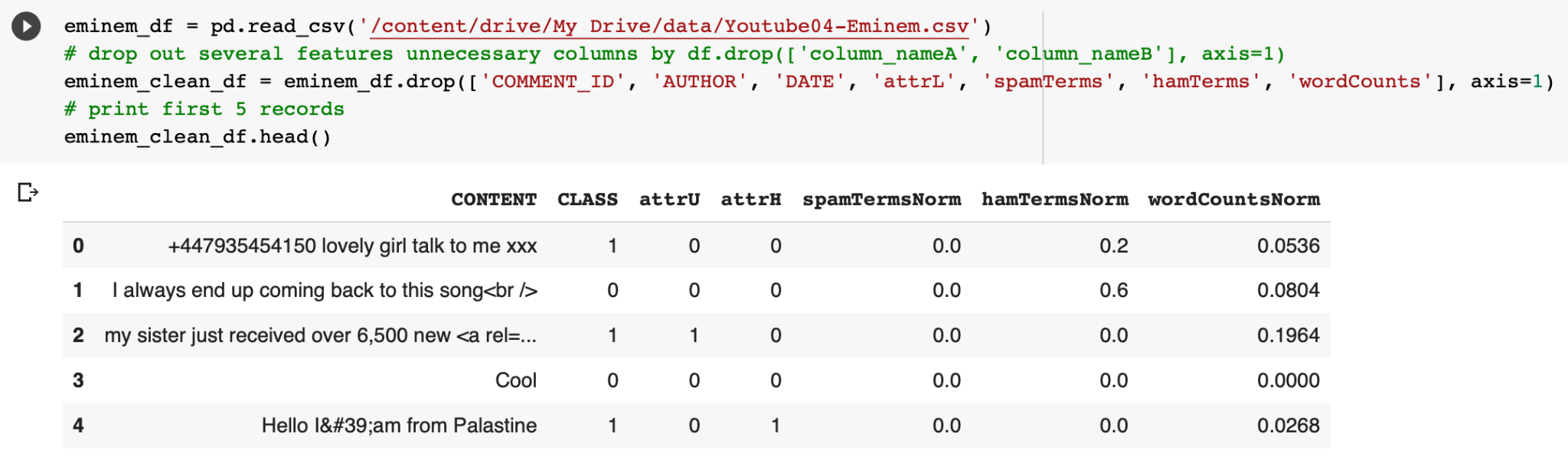


c.)

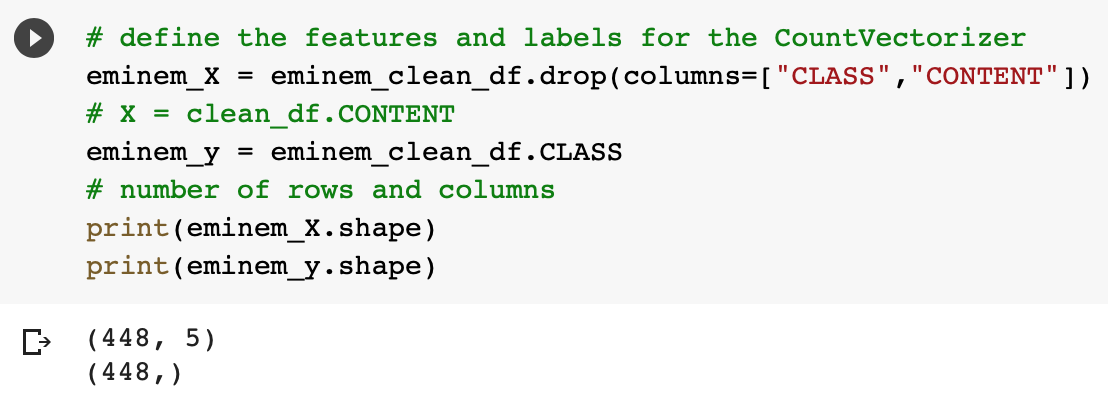
* Using the data in Youtube04-Enimem.csv, I followed the similar procedure as before as in 1a.) steps i.) to v.); finding the top 10 most frequent spam and ham words then preparing the excel with the same attributes as well as dropping the unnecessary columns, except I am using the whole dataset to test this time for the already built logistic regression model previously.

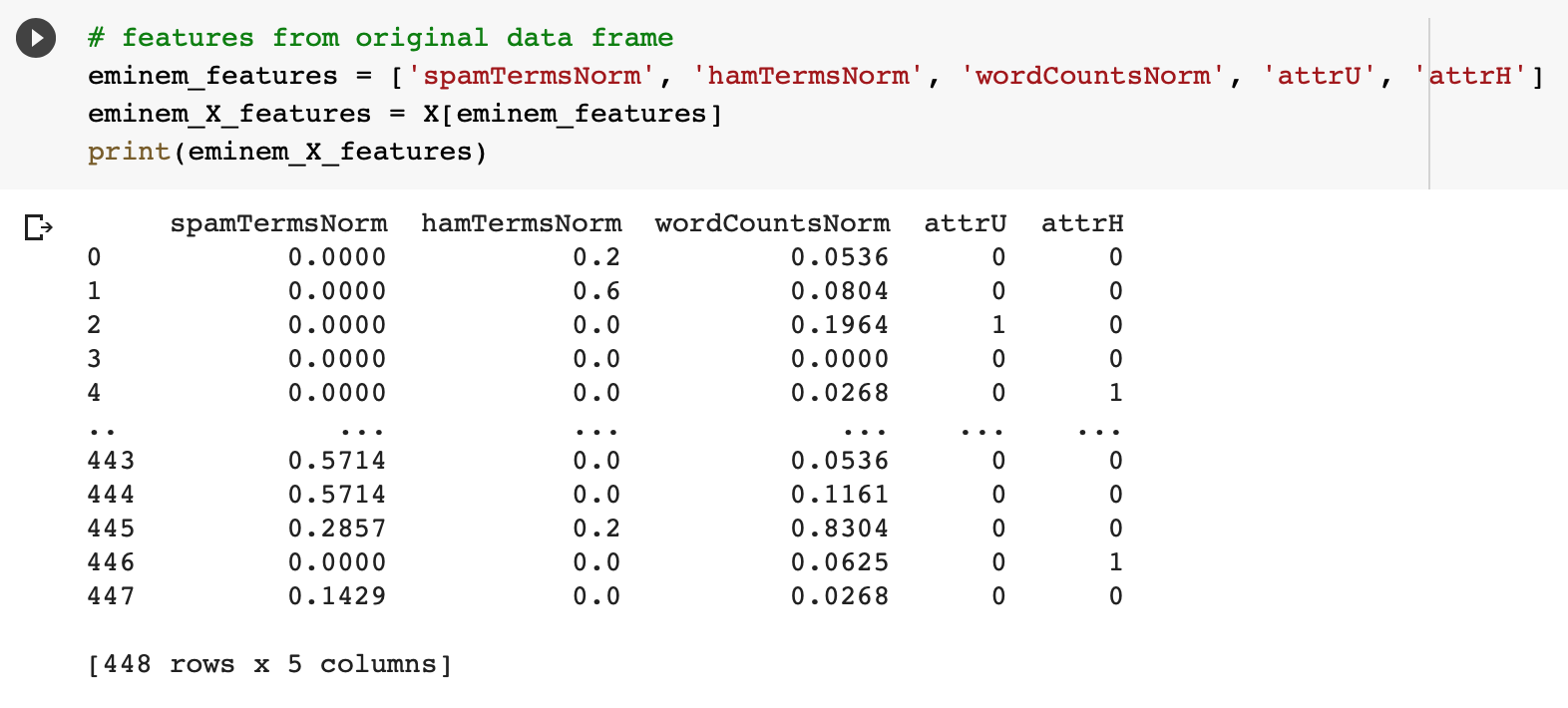




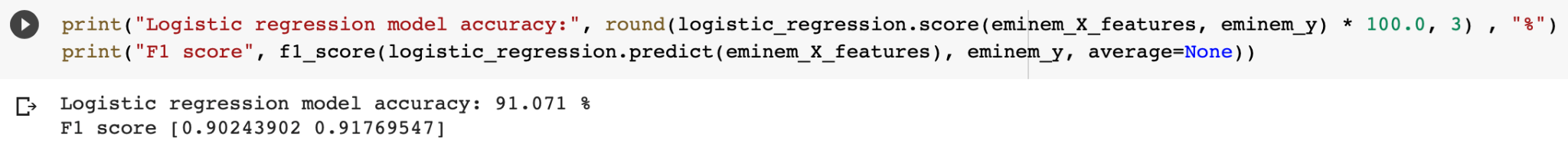


* Then I defined the classes and features from the csv files data frame, that I will be feeding to the classifier in the later steps.

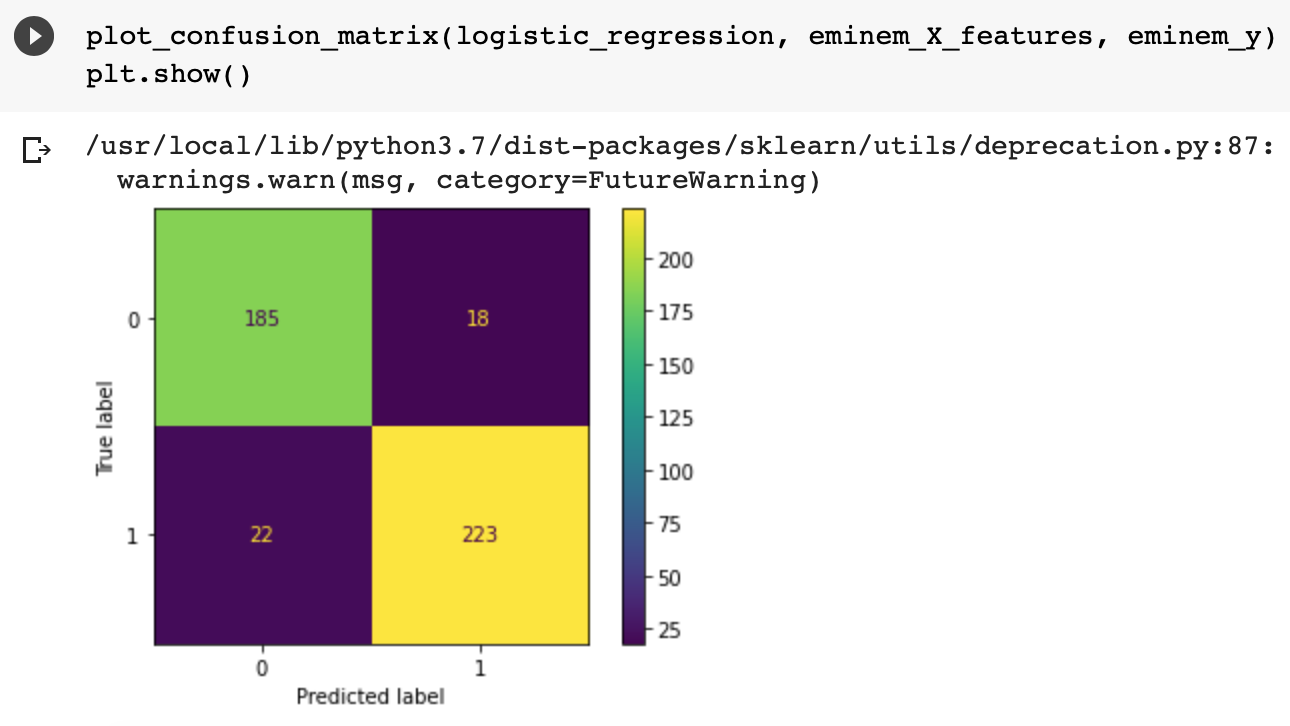




* Then I used the whole dataset to feed into the previously constructed logistic regression classifier model to get the accuracy and F1-score.
  + The accuracy score is 0.91 which is good with an F1-score of 0.90 and 0.92



* Then print the confusion matrix.



* The F1-score is the metric I prefer to measure the performance; the closer to its best value 1, the better the model is performing. F1-score is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. F1-score is useful when you have data with imbalance classes; in the case of Youtube 04-Eminem.csv, there are more spam classes than ham (non-spam) class. Although just slightly imbalanced, F1-score would still be a better metric to evaluate our model on. In contrast to the normal arithmetic mean that gives all values the same weight, the harmonic mean gives a higher weight to low values, thus taking into account how the data is distributed.

