**Assignment 1** (15%)

CSE 5120 (Section #01) – Introduction to Artificial Intelligence – Spring 2021

*Submitted to*

Department of Computer Science and Engineering  
California State University, San Bernardino, California

*by*

Alan L. Perez (004862867)

(Your collaborator in this homework (if any)): James Flora

Date: *May 9th, 2021*

*Email:*

* *004862867@coyote.csusb.edu*
* *Your collaborator’s email (if you collaborated with any): N/A (didn’t get his email, he helped me out with the professor during the professor’s office hours)*

**Assignment Report**

Brief description of your work here acknowledging your collaboration with your class fellow (or a friend from other CSE 5120 section), and the capacity at which he/she collaborated with you, followed by the algorithms you implemented.

1. **Classifier\_lingSpam.py for Ling-spam dataset**

Your brief explanation of the dataset, your code solution, and any documentation with screenshots of your code Evaluation (results from classifier\_lingSpam.py)

The dataset is 702 mail files. Out of those 702, 350 are ham files meaning non-spam and the rest are spam. They are stored in a sorted order. However, in mac you have to sort them explicitly by using emails.sort() in make dictionary. The ham files we classify or label them as 0 and the spam files we label the as 1. The index is from 351 to 701. For the test labels the first 130 are ham files and the last 130 files are spam files. We collect the text from the files using extract features. We open the files, and in each line we get all the words because we need to make a dictionary in order to create a matrix of the data for the support vector machine to understand the data. You also have to use files.sort() in extract\_features. Then we create train\_dir and pass it the path of the directory where the train-mails are. After in train\_mails we create an empty array of 702 zeros. Then we make the final half 1 because of the spam as previously stated. Afterwards we create a train matrix and equal it to extract\_features on train\_dir. This is done in order to get the information out of train\_dir to train the model. Here the model we use is LinearSVC() or linear support vector classification. Then we “fit” the train\_matrix and the train\_labels to the model and put that into the result variable. Next do a similar method with test-labels as previously stated. Create a test\_dir equated to the path of test-mails, we extract\_features from that directory and equate it to test matrix, and create an array of 260 zeros. Set the second half of that directory equal to one for the spam. And we equate results to the model.predict(test\_matrix). Lastly we print and create the confusion\_matrix with test\_labels and results as its parameters. Then we print and create the accuracy\_score with test\_labels and result as the parameters.

A screenshot of a computer

Description automatically generated

1. **Classifier\_enron.py for Enron-spam dataset**

Your brief explanation of the dataset, your code solution, and any documentation with screenshots of your code Evaluation (results from classifier\_enron.py)

In extract\_files the labels were automatically created to differentiate spam and ham. For spam files the file’s name at the end is labeled “.spam” and therefore can differentiate between the two. We look at the word at the second index and check if it is equal to spam using mail.split(“.”)[-2] == ‘spam’. We have a total of 3,3716 emails for the dataset. You then return the train labels. Then you need to do the 60/40 split and you do this with “from sklearn.model\_selection import train\_test\_split. We then set train\_dir to the path “enron-spam”, and dictionary = make\_dictionary(train\_dir). Then we equal freatures, and labels to extract\_features(train\_dir). Afterwards we create the split with “X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.40). y\_train, y\_test come from train\_labels because we provided the labels and features in train\_test\_split. Extract features gave us the features matrix and the labels. Then we use the same “LinearSVC()” model as before. Here instead of Then we “fit”[ing] the train\_matrix and the train\_labels to the model we use “(X\_train, y\_train)” . Then we equate result to model.predict(X\_test). Afterwards we print and create the “confusion\_matrix” using “(y\_test, result)” as parameters and we print and create the “accuracy\_matrix” using “(y\_test, result)” as parameters. Finally we save the model into a sav file and name it “saved\_model = emailClassifeier\_enron.sav” and we save it using “with open(saved\_model, ‘wb’) as file: pickle.dump(model, file)”.

A picture containing text, screenshot, monitor, indoor

Description automatically generated

1. **Evaluation (evaluation.py) for your model performance evaluation - optional**

You can also provide brief description of your code written in evaluation.py to load the saved model that can be readily used on test dataset for the staff. This section is optional, and you can skip it

N/A