Cole Ellison, Connor Levesque, and Henry Swaffield
Machine Learning HW4 Approach

To more effectively address the problem of email spam classification, our group’s high-level approach to maximize our cross-validation accuracy and F1 score was to create a much larger feature set and pick the best feature subsets as well as hyper-parameter arrangements from the possible combinations. The features and hyper-parameters considered were not only those preexisting; we also created new variables to tweak. To achieve the goal of identifying the optimal hyper-parameters and variable subsets, we prioritized building a modular system to automate the workings of our pipeline, such that one call to spamsvm.py performs the search through our multidimensional hyper-parameter-space, calling all necessary sub-steps. Once again, our efforts can be split into categories: expanding the feature set to encompass more information and choosing which features to use, also known as variable subset selection.

The first and most obvious change we made was to dramatically expand the set of words considered, broadening it to include all tokens returnable by a call to Python’s str.split() that are seen in the test set. We then honed these down with hyperparameters discussed later. Even before any other changes were made, this change brought substantial increases in accuracy. Another feature-set expansion strategy was to separate the subject line from the rest of the email and flag those as separate features, which also yielded surprisingly good increases in accuracy. We automated this process so that the index of a word corresponds to its feature. That is, if “dog” has index 3 and appears in email 235, the third entry of the feature array for email 235 would be true (1 in our case). We then doubled the length of the feature array and added specific flags for word that appeared in the subject line of the email. To return to the example above, if “dog” appeared in the subject line in email 235 and our word list was 4083 words long, then the 4086th entry in the feature array would be true in addition to the entry mentioned above. By looking at the subject line independently, we were able to raise our accuracy nearly a full percentage point on the cross validation data.

Most of our thinking, and automated processes, revolved around selecting the ideal hyperparameters and variable subsets. After reading Guyon and Elisseeff’s 2003 paper, *An Introduction to Variable and Feature Selection*, we decided that variable filtering would be the best approach to employ for selecting better and better variable subsets. At a high-level, we defined correlational criteria between features and output labels, and using these criteria we trained using varying numbers of the seemingly-most-predictive features. As the authors discuss in their paper, mutual information is a powerful force, meaning that two seemingly useless variables in isolation, could yield powerful insights,
when considered together, but despite that they say that filtering (which considers a single variable separate from the context of others) still performs mostly as well as other more sophisticated approaches, such as wrapper methods or embedded methods (we considered wrapper methods, which treat the learning machine as a black box, specifically we backward elimination and forward selection of features to include).

Our filtering criteria removed words which occurred less than F—our frequency hyper-parameter. We also ignored words whose presence were not sufficiently correlated with an email either being spam or not. This was determined by averaging the labels (-1 or 1) associated with each instance of a word and taking only those whose average absolute value was above a set threshold. Thus T, for threshold, became our second hyper-parameter. With these two additional hyperparameters established, we wrote a loop to sweep through the possible values of F, T, and C to find an optimal combination. We began by comparing broad ranges of values and slowly narrowed our focus on the combinations which maximized our accuracy and F1 score. We also compared alternative types of kernels and kernel specific parameters before settling on the linear kernel. This search resulted in F = 35, T = 0.1650, and C = 0.0167, with an accuracy of 98.75% and an F1 score of 0.9863 on our cross validation set.

Works Cited