We developed a machine learning-based predictive model to detect spam emails as part of this work. Spam detection is a binary classification problem. Hence, we selected “class” as the target variable and used the other variables present in the data set as predictors.

One issue with the given data set is high class imbalance, with around 61% of records marked as non-spam. We initially tried randomized oversampling of the minority class (1). However, randomized oversampling introduce duplicates, and hence, the model evaluation will be incorrect. Other oversampling techniques are interpolation dependent and may introduce additional noise in the training data.

Another potential problem was the data dimensionality (57 predictors in total). We tried incorporating Principal Component Analysis but did not find it helpful, which could be attributed to the non-linearity in the data set.

Therefore, we thought of using machine learning algorithms that support class weights as parameters and provide feature selection. Moreover, we only have 3320 observations, so going for deep learning models is overkill. Our choices were Random Forests (RF) and Gradient Boosting Machine (GBM), available through the scikit-learn and LightGBM packages.

Although these ensemble machine learning models are somewhat invariant to feature scaling, we thought of analyzing the effects of using different scalers available in the scikit-learn package. We found that MinMaxScaler() gives the best result.

Next, to train the model, we initially used the scikit-learn GridSearchCV with RepeatedStratitfiedKFold having five-folds and three repeats. We chose the stratified version because this is a classification problem. The model evaluation was based on the similarity of the training and validation ROC AUC scores to reduce overfitting. We then checked the AUC for the test data. For additional interpretability, we also showed F1, precision, recall, and balanced accuracy. Note that ROC AUC was the metric that was optimized.

Running the scikit-learn GridSearchCV with different hyperparameters of RF and GBM on an Alienware M17 R1 (personal PC) took around 3 hrs in each case. Hence, we switched to the Missouri S&T Foundry High-Performance Computing cluster and used Dask to train our models in a distributed computing environment. This resulted in a significant reduction of computing time wherein we could tune several parameters in a very short time. For example, the GBM model training took around 3 hrs on the Alienware and took only 10 minutes using Dask.

Finally, we found that the GBM with a test AUC score of 0.961 slightly outperforms RF (AUC = 0.954).