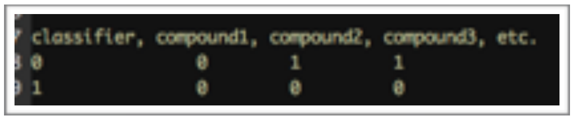
Rob Jarvis (GeorgeCostanza)
Current Data miner score - .57 self f1 score = 0.646153846154 ranked 21st

For this homework assignment I decided to follow a similar prototype approach as I did in Homework 1. I separated the preprocessing and classification into two separate python files and wrote the cleaned up data files to disk and read from them when performing the classification. After discussing approaches with classmates, I thought a good first step would be to use the pandas library for the data frame feature. I attempted to run a few quick tests and see how the classification model would work straight up. I continuously got errors due to the difference in length between the training and testing array being passed into the sklearn library. After an incredibly frustrating half hour, I remembered that it was mentioned in class that there may be different number of features in the testing and the training dataset. With that in mind, I combined all records from the testing and the training dataset into one pandas data frame. I also created a dictionary that was counting the number of times compounds were appearing. I would use this dictionary later to reduce the total number of compounds in hopes of having a more accurate classifier. Once all of the compounds were added into the dictionary, I iterated through each element in the dictionary. I set a threshold for the compounds to keep based on the number of times they appeared in the dataset. This number ranged from anywhere to 25-51. My reasoning behind this was that compounds that appear in the dataset more often are going to have a greater affect on the classifier (0 or 1) than compounds that are appearing infrequently. Once the unnecessary compounds were weeded out, I created a new structure for the data that looks like the screenshot below.



```
classifier, compound1, compound2, compound3, etc.
0            0          1          1
1            0          0          0
```

The screenshot is just from my notes in vim, but shows what the new datasets will look like. The first column will hold the classifier value (only for the training dataset, the test dataset will not have a classifier included.) Then the compounds are included across the rest of the columns. The 0 and 1 for the compound represents whether or not the compound was included for this particular record. I wrote the necessary data to disk files and moved on to the next step.

In the prediction portion of this assignment, I messed around with a few different models. I stuck to the sklearn library, but attempted to use KNN, RandomForestClassifier, and NaiveBayes. My best results have come from using RandomForest. I encountered multiple issues when using the SKLearn NaiveBayes model. I was forced to convert data types of my array due to some issue with Numpy.

I messed around with a few different of the feature selection/reduction libraries that SKLearn offers. I used the SKLearn SelectKBest and chi2 library to select the best features based on the chi2 statistical test. Chi2 was used because we are working with a binary classification problem. With the SelectKBest library you can choose the K number of best classifiers to keep in your data. I usually experimented with ranges from 25-50.

For my F1 Score I again used the sklearn library. I noticed that my F1 score was generally pretty close to my accuracy score shown on the data mining website. But the values would vary each run, sometimes as much as .1 (I even sometimes received an undefined metric score). The f1 score is a way to measure the test's accuracy but it considers the precision and recall of the score (a weighted average of precision and recall). I imagine that as more folds are added in to the validation, the more accurate the f1 score would be. My current accuracy score on the data miner site is .57, even though I received an F1 score of .62.

I often get precision and recall confused. Precision is looking at the correct predictions along with all possible values. Meaning that we're looking at the right predictions made along with the total number of correct and incorrect possibilities. Recall is only concerned with the correctly labeled items. Recall is looking at the number of correctly labeled features but is also looking at the total number of correct labels that could have been made. Here are two simplified formulas for recall and precision. Precision: Correctly labeled items / correctly labeled items + incorrectly labeled items. Recall: Correctly labeled items/ total number of possibly labeled items.

In the prediction model script I created new random forest objects for testing the F1 score. In these tests I would fit the RandomForest model to my reduced feature classes and the known prediction results. I would use my initial prediction to create a baseline RandomForest model

Even though I made many different changes to the parameters I usually found myself consistently hitting 57% on the accuracy score from the DataMiner site (even with Fscore a bit higher around 60%). A few times I would dip a bit lower (into the 30's) but never much higher. I played around with the KNeighborsClassifier library for KNN but did not see better results from my F1 score. The F1 score for the KNN was usually around 45.

I believe that one issue I have with my implementation is in the feature reduction. My current implementation is using the SelectKBest feature on the train dataset and then on the test dataset. I think this is incorrect and I should either be running this feature as a part of preprocessing, or only on the train dataset. Once I have the features reduced on the train dataset, then I should be trimming the testing dataset based on the features used in the training dataset. Since my first step in the preprocessing pipeline was to combine all the features (from both training and testing) together, I don't believe I would run into any sort of issues with imbalanced or missing data.

In order to better handle the imbalanced data, a support vector machine may yield better results. Using the F1 score was a good first step along with the other additional steps taken. But I think that possibly giving some additional weight to the positive labels would result in better performance on my model.

I am also curious to use the pickle feature that is included with sklearn. Maybe this would give me better results in efficiency/speed than writing CSV files to disk. If I could write/read objects to disk, that would help my performance.