Email Classification By Machine Learning - GROUP 2

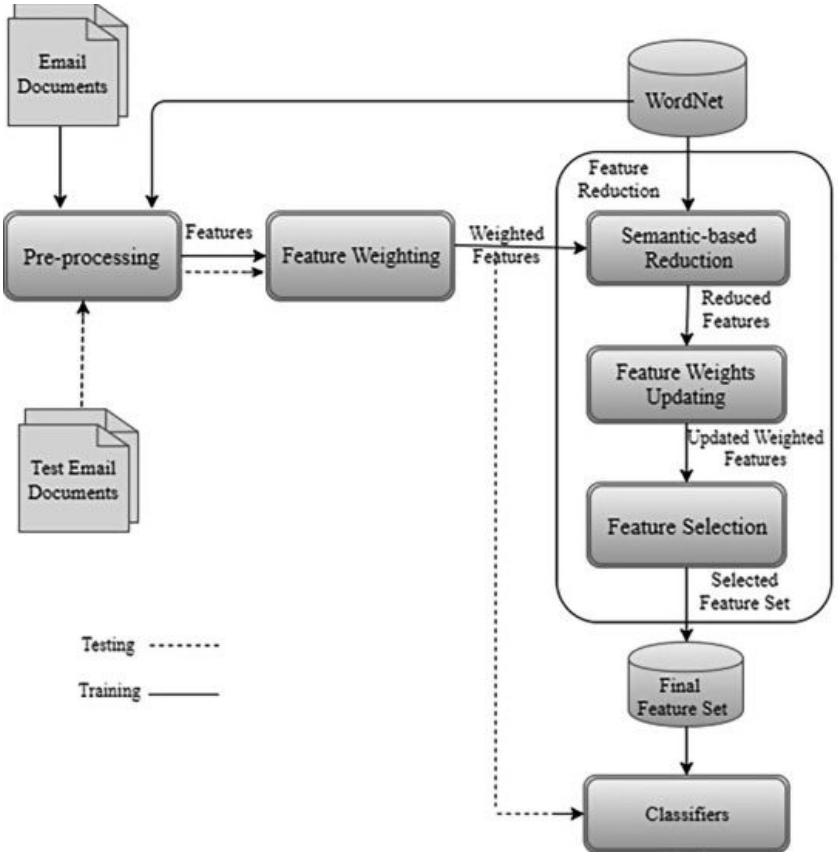
Will Smith
Melika Shekarriz
Adam Ismaeili

Introduction

What is classification?
Classification is a technique to categorize our data into a desired and distinct number of classes where we can assign a label to each class. One of the applications of classification is email classification. We can use different types of machine learning algorithms such as SVM, KNN, Decision Tree and Random Forest(RF). For this task the goal of the algorithm was to determine whether a given email was of type 'personal' or 'promotional'. The methodology with which we found the best results was a Decision Tree.

Methodology

Decision Tree
Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data. Decision Tree, as it name says, makes decision with tree-like model. It splits the sample into two or more homogeneous sets (leaves) based on the most significant differentiators in your input variables. To choose a differentiator (predictor), the algorithm considers all features and does a binary split on them (for categorical data, split by cat; for continuous, pick a cut-off threshold). It will then choose the one with the least cost (i.e. highest accuracy), and repeats recursively, until it successfully splits the data in all leaves (or reaches the maximum depth).



Data Exploration

For this assignment, two datasets were provided to our team; the first for training and the second for testing. Each dataset was composed of thirteen columns, made up of an ID field, three categorical fields and nine quantitative fields. The training data had 14,063 rows while the test data had 6,029.

Process

Step One

The first thing our group did was to establish a basic testing framework for testing a predictive algorithm on a part of the training data. In this step we established a method of cross validation, and being supplied with a sample solution ran this algorithm through the cross validation testing method. In order to confirm that the process worked, we then submitted the solution generated by the provided sample algorithm. The solution was not good, as can be seen in the results table below (0.23423). However, this result was similar to the testing result generated by our testing process which was a relief. In this testing process we take 40% of the training data away for testing, and train on the remaining data. Then we generate a prediction of the labels on the 40% test data and compare that to the real labels. We perform this process ten times with a random split of 40/60% of the data each time, and take the average of the

accuracy each time to determine the accuracy of our algorithm. The accuracy is logged as in the below screenshot, beginning with 'Average accuracy score over 10 attempts ...".

```
weighted avg       0.98      0.98      0.98      5626

/Users/elbershayz/Documents/CentraleSupelec/task/tree.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.
  dataSet[MAIL_TYPE_FIELD] = dataSet[MAIL_TYPE_FIELD].apply(lambda value: value.lower())
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1903
           1       0.98      0.98      0.98      3723

    accuracy                           0.98      5626
   macro avg       0.97      0.97      0.97      5626
weighted avg       0.98      0.98      0.98      5626

Average accuracy score over 10 attempts =  0.9754354781372202
/Users/elbershayz/Documents/CentraleSupelec/task/tree.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.
  dataSet[MAIL_TYPE_FIELD] = dataSet[MAIL_TYPE_FIELD].apply(lambda value: value.lower())
-----------COMPLETE-----------
```

Step Two

It was now time to generate our own prediction algorithm. Being the simplest type, we selected the decision tree algorithm to try first. In order to just get a solution of our own out quickly, we elected to remove all categorical fields and the date field from the dataset rather than go through the encoding process. This way it was quite quick to get a solution working. Quickly it seemed like our results were looking quite good from the testing process described above, so we ran the algorithm on the real test data and submitted our solution on Kaggle to get a score of 0.98029, a drastic improvement over the sample code.

Step Three

With such sudden success using this reduced dataset and a decision tree, we decided to see just how small our dataset could get while still maintaining good results. It was thought that if we could see such success with only 75% of the data, maybe we could reduce our dataset down to 10% and still see some good results (the thought being that, for example, the algorithm would

```
Average accuracy score over 10 attempts =  0.8585673658016353
-----------COMPLETE-----------

mean accuracy: 0.9098316639133435
max accuracy: 0.9746000710984715
    - org   ✗
    - tld   ✗
    - ccs   ✓
    - bcced   ✓
    - mail_type   ✗
    - images   ✓
    - urls   ✓
    - salutations   ✓
    - designation   ✗
    - chars_in_subject   ✓
    - chars_in_body   ✓
    - label   ✓
(centrale) elbershayz (master *=) task $
```

be able to completely determine the label of an email based on the images field). To determine which fields to use, we used a powerset of all possible combinations of the remaining fields. We ran each combination through our testing process (256 combinations for 10 iterations each) which took almost 10 minutes. The test determined that using all of the remaining categories except for 'designations' would generate the best result, as can be seen in the above screenshot of the terminal output. When submitted to Kaggle, this result ended up being worse than our previous result, achieving a score of 0.97855.

Step Four
In order to get the best results we thought we should bring back the full dataset rather than continuing with only 75% of the fields. To do this we used a OneHotEncoder to split the categorical values 'org', 'tld' and 'mail_type' into columns for each possible value. We also identified that the values for 'mail_type' were often duplicated but with different capitalisations, so we converted each value to lowercase. Still the date field was deleted, but this predictor found good testing results so we submitted a solution to Kaggle and received our best results yet; 0.99146. This would prove to be our best result.

Step Five
Having found such good results with a decision tree, the group was skeptical of achieving such results with another algorithm (especially with minimal time remaining to fine-tune such an algorithm). Nevertheless we tried using a logistic regression algorithm on the same set as data as before (used for achieving the best tree results) and scored 0.93292 on Kaggle. Because of the setup performed in step 1 trying this new algorithm took a matter of minutes.

Step Six
The final algorithm used was a MultiLayerPerceptron. We tried using the perceptron with two hidden layers of sizes 5 and 2, and obviously the output with 2 layers (being personal and promotional). Through testing this algorithm seemed quite inaccurate, and a submission to Kaggle received an accuracy of 0.78412, affirming the tests. It was believed that the predictions could certainly be improved through refinement of the algorithm parameters, but a few attempts at increasing the amount of layers and size of each layer found less successful tests so the project was abandoned under the notion of time constraints.

Future optimizations

With more time, it's strongly believed that a better solution could be developed. As a starting point, it's most likely that a random forest could achieve better results than our individual tree because of the increased depth and complexity with which it can make its predictions. It's also highly feasible that some kind of neural network, with enough tuning and computation power, could generate a better result. Lastly, it would likely help the result to include the date column, however work must be done to decide on the best way to quantify this data (this could based on time of day, day of week or time in history).

Submission #	Kaggle Accuracy
1	0.23423
2	0.98029
3	0.97855
4	0.99146
5	0.93292
6	0.78412

Conclusion

In conclusion, the optimal result was found using the Decision Tree Classifier rather than the Logistic Regression and Neural Network solutions. It's believed that a lot of this success is based on the ease with which one can setup a tree, and the lack of understanding required. It's highly likely that given more time, or if we were machine learning experts, we would have been able to use a more complex algorithm to generate better results than our 99.146%. In saying that, sometimes the simplest approach is the best and perhaps this is the case here.