CS5691: Pattern Recognition and Machine Learning
Assignment 3

Lalit Jayanti, ME21B096

Question 1: SPAM or HAM

Part (i): Data-set Choice

For this assignment the Enron Spam Dataset (`https://github.com/MWiechmann/enron_spam_data`) is chosen. It consists of 33716 mails, (17171 Spam mails, 16545 Ham mails). Table 1 provides a description of the data available in this dataset.

Column	Description
Subject	The subject line of the e-mail
Message	Email Content as plain text
Spam/Ham	"Spam" or "Ham" classification
Date	Date the email arrived in YYYY-MM-DD

Table 1: Dataset Description

Part (ii): Feature Extraction

An exploratory study is conducted on the dataset to find the total number of distinct words and their distribution over the 2 labels (Ham, Spam) to gain a better idea of the dataset.

Firstly, each email is stripped of all punctuation and numbers using regex. Then all the remaining characters are converted to lowercase, finally the email is split into separate words.

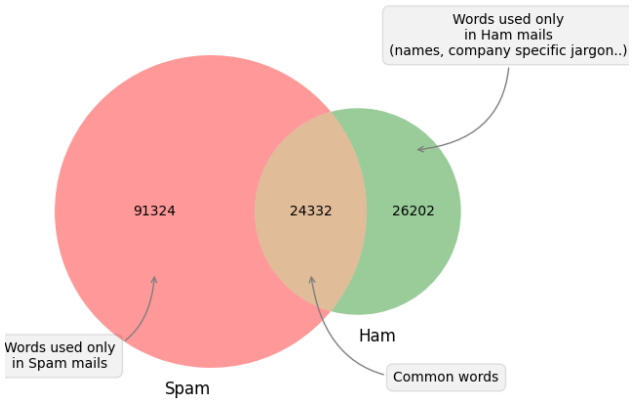


Figure 1: Venn Diagram of Distribution of distinct words

It is observed that the among the most occurring words in all mails are words such as "enron", "houston", "company", "hou". These words are highly domain specific and occur frequently in the "ham" mails of the dataset, however these domain specific words may not occur in general "ham" mail. Thus any classifier built on top this directly may perform well on tests belonging to this dataset however it may not perform well in classifying general mails. However, if only words occuring in spam mails are considered, it is observed that the most occuring words are now "please", "free", "money", "special", "stock", "limited", "investment".
Based on this analysis 2 Feature sets are extracted. Firstly, the dataset is split in 80% for training and 20% for testing (ensuring that relative percent of spam/ham is kept same for both).

Feature set 1: The top K (K = 500) most occuring words in all "Spam" emails are found. Now for every email (in the training set and later on in the test set). The extracted dataset is of the size (Number of emails = N) $\times$ K. Where the $(i,j)^{\text{th}}$ entry is the number of times the j^{th} most occurring word (overall Spam) occurs in the i^{th} email. The set of K (K=500) most occuring words is additionally stored, this is to aid in encoding and decoding the features later on.

Feature set 2: The top K (K = 500) most occuring words in all emails are found. Now for every email (in the training set and later on in the test set). The extracted dataset is of the size (Number of emails = N) $\times$ K. Where the $(i,j)^{\text{th}}$ entry is the number of times the j^{th} most occurring word (overall) occurs in the i^{th} email. The set of K (K=500) most occuring words is additionally stored, this is to aid in encoding and decoding the features later on.

Note: The **Feature set 1** contains words that occur in only spam emails, therefore it contains very few domain specific words, hence it may be more general and classifiers trained on this may show more robust performance in a general setting.
Implementation:

- Program for Feature Extraction: `scripts/feature_extraction.py - featureExtractor()`
- Program of splitting and building dataset: `scripts/build_dataset.ipynb`
- Input Data: `data/enron_spam_data.csv`
- Feature set 1: `data/enron_train_encode_f1.csv, data/enron_train_encode_dict_f1.json`
- Feature set 2: `data/enron_train_encode_f2.csv, data/enron_train_encode_dict_f2.json`

Part (iii): Algorithms and Procedure

Four Binary classification algorithms are tested on the extracted datasets for various hyper parameters and kernels where applicable. The trained models are then tested on the 20% test data.

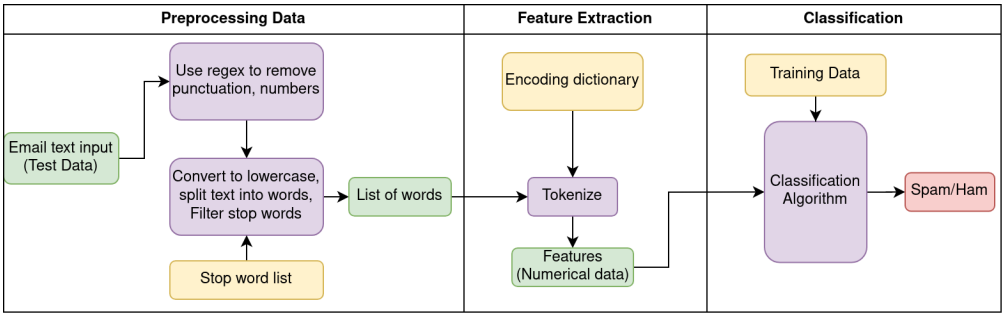


Figure 2: Procedure for Training and Prediction

The Figure 2 shows the high level procedure followed to train the 4 classifiers and predict for given data. A detailed explanation of the training procedure for each classifier is presented in the following section.

(A) : Naive Bayes

The Naive Bayes classifier is trained over this dataset and tested. Here the underlying distribution is assumed to be a Bernoulli distribution for each word (occurrence or the non-occurrence of the word).

Accordingly, the extracted dataset is modified, in each data point, if a particular word count is greater than 0 it is made 1 (word has occurred), else it is kept 0 (word has not occurred).

Subsequently, the linear decision boundary is computed for the model and used to make predictions for the test data.The test errors are tabulated in Table 2.

Feature Set 1	Feature Set 2
86.30%	92.811%

Table 2: Naive Bayes test accuracy

Implementation:

- Program for Naive Bayes: `scripts/naive_bayes.py - naiveBayes()`
- Program for Testing: `scripts/spam_ham_classify.ipynb`
- Feature set 1: `data/enron_train_encode_f1.csv, data/enron_train_encode_dict_f1.json`
- Feature set 2: `data/enron_train_encode_f2.csv, data/enron_train_encode_dict_f2.json`
- Test data: `test/email1.txt...email6621.txt`

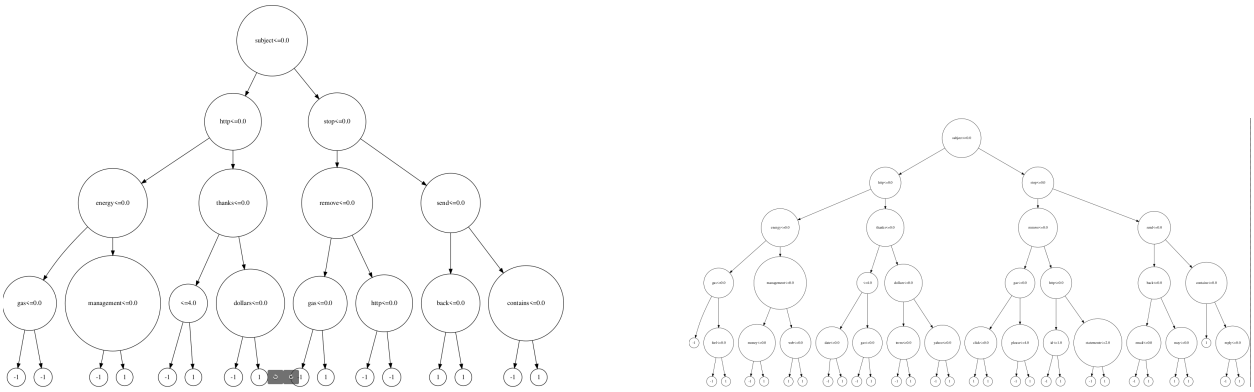
(B) : Decision Trees

The binary classifier, decision tree is implemented and tested on the chosen dataset. The testing is done for trees of different heights. The test accuracy is tabulated in Table 3.

Max Height	Feature Set 1	Feature Set 2
3	73.46%	77.25%
4	75.58%	82.16%
5	79.23%	85.94%
6	79.67%	87.40%

Table 3: Decision Tree test accuracy

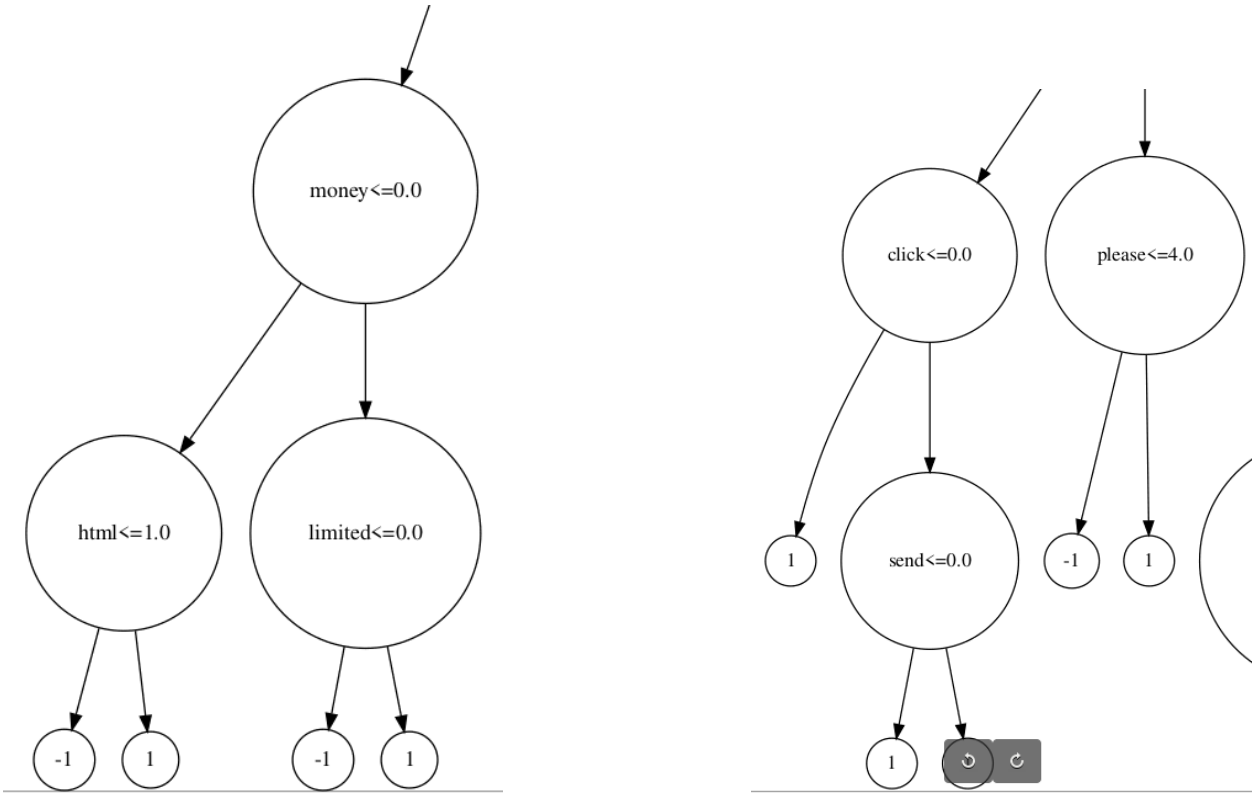
It is observed that as height of the decision tree increases, the test accuracy improves. Figure 5 shows the visual representation of the decision trees obtained. The Figure 4 show some segments of the trees, for example as observed in Figure 4b, if the word "please" occurs more than once it is most likely to be spam, similarly the co-occurrence of words "money", "limited" may be indicative of spam as seen in Figure 4b.



(a) Decision tree with max height 4

(b) Decision tree with max height 5

Figure 3: Decision trees visualized



(a) Decision tree segment

(b) Decision tree segment

Figure 4: Example Segments of the Trees

Implementation:

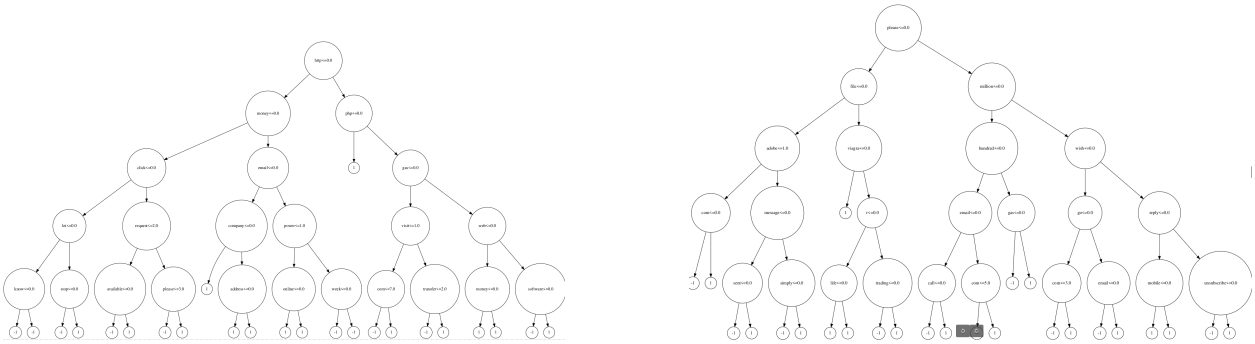
- Program for Decision Trees: `scripts/decision_tree.py - decisionTree()`
- Program for Testing: `scripts/spam_ham_classify.ipynb`
- Feature set 1: `data/enron_train_encode_f1.csv, data/enron_train_encode_dict_f1.json`
- Feature set 2: `data/enron_train_encode_f2.csv, data/enron_train_encode_dict_f2.json`
- Test data: `test/email1.txt...email6621.txt`

(C) : ADA Boost

The meta classifier, ADA boost is implemented and tested on the chosen dataset. This classifier builds an ensemble of decision tree stumps. The testing is done for ensemble of trees of different heights, and different number of trees. The test accuracy is tabulated in Table 4. It is expected that this classifier will outperform the decision tree classifier.

Total Trees	Max Height	Feature Set 1	Feature Set 2
3	10	85.14%	82.63%
3	30	88.14%	88.97%
4	30	89.32%	92.96%
5	10	87.6%	92.85%
5	30	91.2%	94.88%

Table 4: ADA Boost test accuracy



(a) Example decision stump from ensemble

(b) Example decision stump from ensemble

Figure 5: ADA Boost visualized

Implementation:

- Program for ADA Boost: `scripts/ada_boost.py - adaBoost()`
- Program for Testing: `scripts/spam_ham_classify.ipynb`
- Feature set 1: `data/enron_train_encode_f1.csv, data/enron_train_encode_dict_f1.json`
- Feature set 2: `data/enron_train_encode_f2.csv, data/enron_train_encode_dict_f2.json`
- Test data: `test/email1.txt...email6621.txt`

(D) : Support Vector Machine (SVM)

The Soft Margin Support Vector Machine is tested on this dataset. Since it is not known of the data has a linear structure, 3 kernels are tested (linear, radial basis function (rbf), polynomial (poly)). Furthur, the hyper parameter "C" is cross validated for the model this is done using K-Fold cross validation with K=2. The results are tabulated in Table 5. The classifiers are ranked according to the mean validation over the folds.

Kernel	C	Mean validation Score	Rank
linear	0.010	0.908	5
rbf	0.010	0.751	10
poly	0.010	0.524	15
linear	0.100	0.913	4
rbf	0.100	0.830	9
poly	0.100	0.555	14
linear	1.000	0.902	7
rbf	1.000	0.907	6
poly	1.000	0.700	12
linear	10.000	0.902	8
rbf	10.000	0.915	3
poly	10.000	0.646	13
linear	100.000	0.915	2
rbf	100.000	0.916	1
poly	100.000	0.724	11

Table 5: Cross Validation for SVM

Kernel Selection: Based on the results in Table 6, the "rbf" kernel is chosen based on its better validation accuracy. In hindsight, it is logical that this kernel will perform better, since 2 emails which have similar content (spam/ham) will be "closer".
Cross validation: K-Fold (K=2) cross validation is performed on this dataset, which revealed that C=100 is a suitable value for the hyperparameter.
A close 2nd preference will be the one with a linear kernel and C=100.

Kernel	C	Feature Set 1	Feature Set 2
rbf	100	94.39%	95.19%
linear	100	94.39%	96.81%

Table 6: SVM test accuracy

Implementation:

- Program for SVM: pre-existing `sklearn svm.SVC()` has been utilized
- Program for Testing: `scripts/spam_ham_classify.ipynb`
- Feature set 1: `data/enron_train_encode_f1.csv, data/enron_train_encode_dict_f1.json`
- Feature set 2: `data/enron_train_encode_f2.csv, data/enron_train_encode_dict_f2.json`
- Test data: `test/email1.txt...email6621.txt`

Part (iv): Performance

Based on the study conducted in the previous section, the best performing classifiers are presented for further evaluation in Table 7.

Classifier	Feature Set 1 (Top 500 of all spam words)	Feature Set 2 (Top 500 of all words)
Naive Bayes	86.30%	92.811%
Decision Tree (height: 6)	79.67%	87.40%
ADA Boost (trees: 30, height: 5)	**91.2**%	**94.88**%
SVM (kernel: linear, C: 100)	**94.39**%	**96.81**%
SVM (kernel: rbf, C: 100)	**94.39**%	95.19%

Table 7: Top Performing Classifiers

For **Feature set 1**, Ada boost and SVM outperform Naive Bayes and Decision trees.

For **Feature set 2**, Ada boost SVM and Naive Bayes all produce accuracy above 90% indicating that they are good classifiers for this dataset.
In particular it is pointed out that classifiers return better results over **Feature set 2** than over **Feature set 1** however this is obtained at a certain loss in the generalization of the classifier. Since training on **Feature set 2** results in learning relations based on domain specific data from the ham mails.

	Unnamed: 0	Given Label	Naive Bayes	Decision Tree (height: 6)	ADA Boost (trees: 30, height: 5)	SVM (kernel: linear, C: 100)	SVM (kernel: rbf, C: 100)
0	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1
2	2	1	1	1	0	1	0
3	3	1	1	1	1	1	1
4	4	1	1	1	0	1	0
...	...	...	...	...	...	...	...
6617	6617	0	0	0	0	0	0
6618	6618	0	0	0	0	0	0
6619	6619	0	1	0	0	0	0
6620	6620	0	0	1	0	0	0
6621	6621	0	0	0	0	0	0

6622 rows × 7 columns

Figure 6: Predictions with different classifiers tabulated

Implementation:

- Procedure for training and predicting: `scripts/spam_ham_classify.ipynb`
  - To run this script, use the "Restart Kernel and Run all cells" option in Kernel
  - All the predictions for the chosen classifiers will we found in:
  - `outputs/predictions_f1.csv`
  - `outputs/predictions_f2.csv`
- Test data: `test/email1.txt...email6621.txt`
  - The folder test can be modified by adding emails to it for testing.
- Final Predictions: `outputs/predictions_f1.csv, outputs/predictions_f2.csv`

Part (v): Conclusion

Four different classifiers were tested for spam/ham classification.

Naive Bayes was tested with the expectation that the underlying distribution for words appearing or not appearing in mails could be modelled as a Bernoulli distribution, this yielded good results on **Feature set 2** but not in **Feature set 1**.

Decision trees were tested with the expectation that mails could be classified based on number of occurrences of a word in a mail and their co-occurrences. However, this yielded low test accuracy.

Subsequently ADA Boost was implemented to boost the weak learning Decision tree stumps. This gave a good result (high test accuracy) in both feature sets. Finally, the soft margin SVM was tested with different kernels and was cross validated. This yielded even better test accuracy.

Finally it is concluded that for email Spam-Ham classification, **ADA Boost (30 Trees, height 5) and SVM (rbf kernel)** are good classifiers and the method used to generate **Feature set 1 (top K words in Spam)** yields generalized classifiers which can potentially perform well in a generalized setting.