Assignment 4

Rahul Sangole

The Modeling Problem

The objective of this assignment is to develop classification models which can analyze emails and categorize them as spam or not.

Data

The emails are provided as individual ascii files. There are a total of 9,362 files totaling roughly 114 MB on disk. The data in the files is not clean - neither is the data in a long or short dataframe format, nor is it formatted in a cleaned and consistent key-value dictionary pair format. The following shows the first 10 lines for one of the emails. We can see that these data are a mixture lines without key-value pair formatting (line 1), and some approximate key-value pair ":" separated formatting.

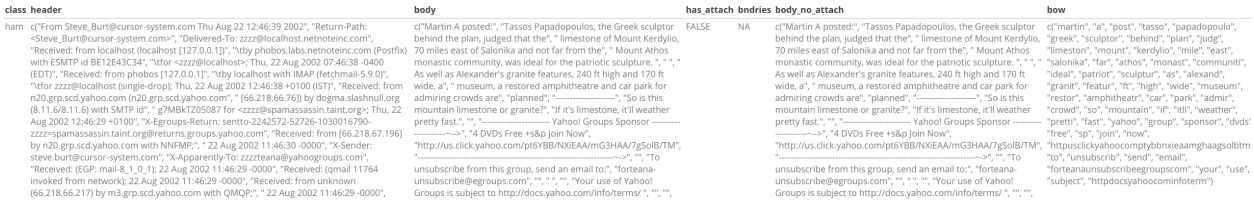
```
From ilug-admin@linux.ie  Mon Sep  2 13:13:50 2002
Return-Path: <ilug-admin@linux.ie>
Delivered-To: zzzz@localhost.netnoteinc.com
Received: from localhost (localhost [127.0.0.1])
    by phobos.labs.netnoteinc.com (Postfix) with ESMTP id 6A0CD47C7C
    for <zzzz@localhost>; Mon,  2 Sep 2002 07:43:01 -0400 (EDT)
Received: from phobos [127.0.0.1]
```

Each file is marked `spam` or `ham`, the two classes to predict. The preparation of the data for modeling differed between the Naive Bayes approach - which requires a word frequency based approach for the two classes - and the classification algorithms like randomForest, Support Vector Machine, decision trees, and logistic regression - which require features to be extracted from the email corpus.

Data Preparation

The data is processed entirely using a `tidyverse` approach using data preparation verbs from `dplyr`, `readr` and `tidyr`. Fundamentally, each file is stored "within" a dataframe as an object [this is using the *list-columns* feature of `tibbles`]. Thus, we start with a dataframe which is 9,362 rows by 2 columns [class and raw data]. For each data operation performed, a new column is added which appends a column holding the transformed data objects. To prepare the data for the NB model, the following steps are taken.

- All 9,362 files are read into a tibble. Each file is read in line by line, and stored as a character vector of the length of each respective file.
- Each email is split into header and body sections
- For each email, we check if attachments are present
- If present, the body if the email is extracted sans-attachments
- The "dirty" header is then processed to a cleaned named-character vector which is used downstream in the latter modeling approaches
- The body of the emails are extracted into a bag of words after the following clean up activities:
  - conversion of text to UTF-8 encoding
  - removal of punctuation
  - removal of special characters
  - case conversion to lowercase
  - removal of numbers



class	header	body	has_attach	bndries	body_no_attach	bow
ham	c("From Steve_Burt@cursor-system.com Thu Aug 22 12:46:39 2002", "Return-Path: <Steve_Burt@cursor-system.com>", "Delivered-To: zzzz@localhost.netnoteinc.com", "Received: from localhost (localhost [127.0.0.1])", "\tby phobos.labs.netnoteinc.com (Postfix) with ESMTP id BE12E43C34", "\tfor <zzzz@localhost>; Thu, 22 Aug 2002 07:46:38 -0400 (EDT)", "Received: from phobos [127.0.0.1]", "\tby localhost with IMAP (fetchmail-5.9.0)", "\tfor zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:46:38 +0100 (IST)", "Received: from n20.grp.scd.yahoo.com (n20.grp.scd.yahoo.com", " [66.218.66.76]) by dogma.slashnull.org (8.11.6/8.11.6) with SMTP id", " g7MBkTZ05087 for <zzzz@spamassassin.taint.org>; Thu, 22 Aug 2002 12:46:29 +0100", "X-Egroups-Return: sentto-2242572-52726-1030016790-zzzz=spamassassin.taint.org@returns.groups.yahoo.com", "Received: from [66.218.67.196] by n20.grp.scd.yahoo.com with NNFMP;", " 22 Aug 2002 11:46:30 -0000", "X-Sender: steve.burt@cursor-system.com", "X-Apparently-To: zzzzteana@yahoogroups.com", "Received: (EGP: mail-8_1_0_1); 22 Aug 2002 11:46:29 -0000", "Received: (qmail 11764 invoked from network); 22 Aug 2002 11:46:29 -0000", "Received: from unknown (66.218.66.217) by m3.grp.scd.yahoo.com with QMQP;", " 22 Aug 2002 11:46:29 -0000",	c("Martin A posted:", "Tassos Papadopoulos, the Greek sculptor behind the plan, judged that the", " limestone of Mount Kerdylio, 70 miles east of Salonika and not far from the", " Mount Athos monastic community, was ideal for the patriotic sculpture. ", " ", " As well as Alexander's granite features, 240 ft high and 170 ft wide, a", " museum, a restored amphitheatre and car park for admiring crowds are", "planned", "----------", "So is this mountain limestone or granite?", "If it's limestone, it'll weather pretty fast.", "", "----------", "Yahoo! Groups Sponsor ----------~-->", "4 DVDs Free +s&p Join Now", "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM", "----------~->", "", "To unsubscribe from this group, send an email to:", "forteana-unsubscribe@egroups.com", "", " ", "", "Your use of Yahoo! Groups is subject to http://docs.yahoo.com/info/terms/ ", "", "",	FALSE	NA	c("Martin A posted:", "Tassos Papadopoulos, the Greek sculptor behind the plan, judged that the", " limestone of Mount Kerdylio, 70 miles east of Salonika and not far from the", " Mount Athos monastic community, was ideal for the patriotic sculpture. ", " ", " As well as Alexander's granite features, 240 ft high and 170 ft wide, a", " museum, a restored amphitheatre and car park for admiring crowds are", "planned", "----------", "So is this mountain limestone or granite?", "If it's limestone, it'll weather pretty fast.", "", "----------", "Yahoo! Groups Sponsor ----------~-->", "4 DVDs Free +s&p Join Now", "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM", "----------~->", "", "To unsubscribe from this group, send an email to:", "forteana-unsubscribe@egroups.com", "", " ", "", "Your use of Yahoo! Groups is subject to http://docs.yahoo.com/info/terms/ ", "", "",	c("martin", "a", "post", "tasso", "papadopoulo", "greek", "sculptor", "behind", "plan", "judg", "limestorr", "mount", "kerdylio", "mile", "east", "salonika", "far", "athos", "monast", "communiti", "ideal", "patriot", "sculptur", "as", "alexand", "granit", "featur", "ft", "high", "wide", "museum", "restor", "amphitheatr", "car", "park", "admir", "crowd", "so", "mountain", "lif", "itll", "weather", "prett", "fast", "yahoo", "group", "sponsor", "dvds" "free", "sp", "join", "now", "httpusclickyahoocomptybbnxieaamghaagsolbtm "to", "unsubscrib", "send", "email", "forteanaunsubscribeegroupscom", "your", "use", "subject", "httpdocsyahoocominfoterm")

Figure 1: Dataframe showing the embedded list-columns for all the preprocessing described above

- removal of common stop words
- removal of excess whitespaces
- word stemming

A total of 7 functions in R achieve this initial cleanup. Thereafter, the list of bag of words is converted to a term-frequency matrix, which stores the number of emails containing each unique word grouped by the response variable. Table 1 shows a few lines from this matrix.

Table 1: Word frequency count, by class

bow	n	class
i	2695	ham
list	2047	ham
email	897	spam
list	738	spam

Data Dictionary

To prepare the data for the rest of the classification approaches, the following features are created:

- `isRE` : Does the subject line start with Re?
- `numLines` : How many lines are in the body, sans attachment?
- `perCapsSub` : Number of capital letters in the subject
- `perCapsBody` : Number of capital letters in the body
- `underscore` : Does the sender's email have an underscore in it?
- `subMarkCnts` : How many ? and ! are present in the body?
- `dlrCnt` : How many $ are present in the body?
- `numRecip`: How many people is the email sent to?
- `Priority` : What is the priority tag of the email?

Ten `tidyverse` functions are writting which allow the dataframes to be altered using pipes. This makes the data preparation steps easy to read, and replicable for both the training and the test datasets. The appendix shows a code sample for this.

Table 2: Modeling dataframe with feature engineering variables

class	isRe	numLines	perCapsSub	perCapsBody	bodyCharCt	underscore	sub_exclMarkCnt	sub_quesMarkCnt	dlrCnt	numRecip	priority
spam	FALSE	50	0.09	0.09	2540	FALSE	0	0	3	0	NA
spam	NA	68	NA	0.26	1671	NA	NA	NA	0	0	NA
spam	FALSE	142	0.10	0.03	5525	TRUE	0	0	0	1	1
ham	FALSE	25	0.05	0.04	963	FALSE	0	0	2	1	NA

EDA

Some basic univariate and bivariate exploration shows us some key predictors towards deciding if an email is spam or not. In the figure below, we can see that almost all messages marked priority 1, 2 or 5 are spam messages. Priority 3 (which is usually the default in most email service applications) is nearly an equal split between the two classes. The graph to the right shows the splits by `isRe` which measures if the subject starts with 'Re:'. Here, we can see that almost all email messages which do start with 'Re:' are not spam.

Figure 2: Exploratory Data Analysis showing a few predictor variables which show markedly different distributions for ham vs spam

Model Building

In total, 5 models are built:

1. Naive Bayes using a custom built code (following the directions in the chapter)
2. Decision Trees (using `rpart`)
3. Random Forests (using `randomForest`)
4. Logistic Regression (using `glm`)
5. Support Vector Machines (using `e1071`)

While investigating the models 2-5, the packages mentioned above are used. `caret` is used for some hyper-parameter tuning. Finally, to compare all the models together, the `mlr` package is used which offers the excellent `benchmark` functionality with associated plotting and model comparison tools.

Resampling & Validation Strategy

The model evaluation strategy of a 70-30 Holdout for train and testing is adopted for all the models. The splits are stratified on the response variable to keep the probability distributions consistant across the two sets.

The resulting datasets are 6,055 and 2,616 long for training and testing respectively.

A 10-fold cross validation resampling strategy is used while hyper parameter tuning some of the machine learning models. This cross validation is performed on the 70% training split only.

Model #1: Naive Bayes

For the naive bayes model, the objective is to count the relative frequencies of each word in the spam and ham emails. Based on these observed frequencies in the training dataset, a prior model is build. Then, for each email in the test dataset, the priors are used to calculate a log odds ratio. If the log odds ratio is less than a threshold (we start out with 0 as the threshold), then the email is classified as ham. If the ratio is positive, the email is classified as spam.

This logic is applied to the dataset prepared for NB modeling. A sample dataframe is shown below.

Table 3: Naive Bayes dataframe showing calculations of the log odds ratio for each word in the training dataset

bow	ham	spam	ham_present_ratio	spam_present_ratio	ham_absent_ratio	spam_absent_ratio	log_ratio_present	log_ratio_absent	log_odds_ratio
i	2695	299	0.59	0.20	0.41	0.80	-1.09	0.67	-0.42
list	2047	738	0.45	0.49	0.55	0.51	0.09	-0.08	0.01
use	1912	382	0.42	0.25	0.58	0.75	-0.50	0.25	-0.25
the	1900	549	0.42	0.36	0.58	0.64	-0.13	0.09	-0.05

Since this modeling approach uses priors calculated from the training dataset, this method only works for words which the model has already seen before. Thus, the method does require a large sample size of representative spam and ham data to be able to make good calls. In this exercise, there seems to be enough data, given that the test set performance is quite good. The mmce for the test set is 0.01, with the fpr and fnr at 0.009 and 0.014 respectively.

We can visualize the distributions of the ham and spam log odds ratio calculations as shown in the figure below. The clear separation of the two density distributes shows us that the model performs exceptionally well for being such a simple (or naive) approach towards classification.

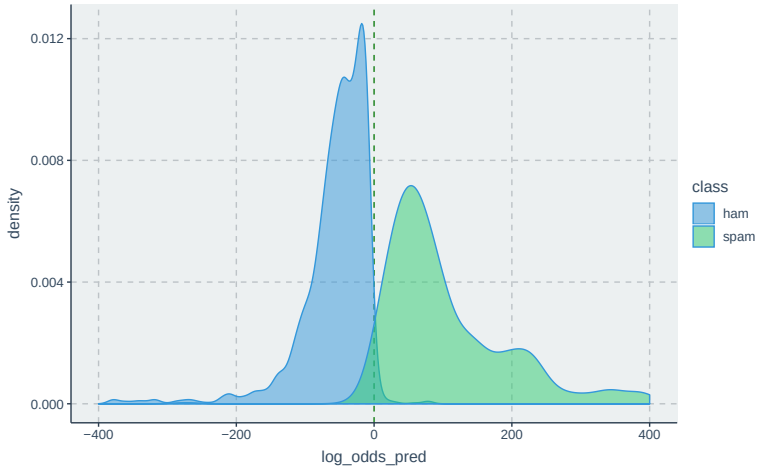


Figure 3: Density plots showing the difference in the log odds ratio between the spam and ham messages for the NB model applied to the test dataset

The threshold used to classify spam vs ham in the scenario above is 0. However, it is possible to trade off FPR for FNR by varying the threshold value. This is investigated and plotted below. By increasing the threshold, a larger number of emails are correctly classified as ham, at the risk of misclassifying correct emails as spam.

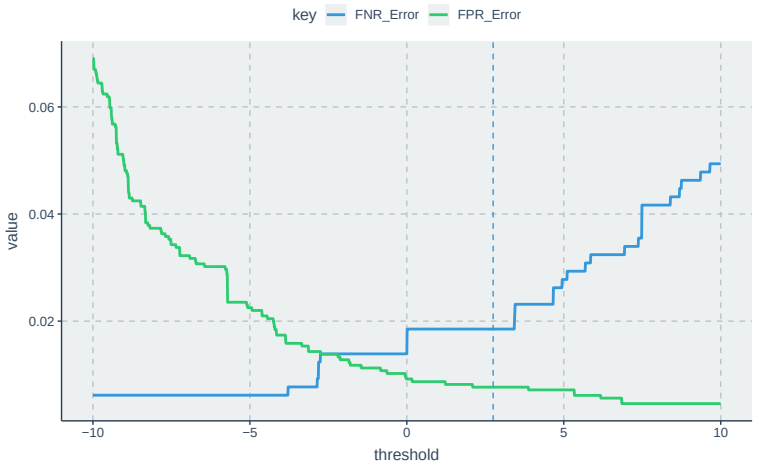


Figure 4: Change in FPR and FNR as the threshold for classifying spam message is varyied from -10 to 10. The dotted line at 2.75 shows where the two error rates are equal.

Model #2: Decision Tree

A decision tree is built on the training data with tuning parameters `minimum split size` = 20, `minimum bucket size` = 50, `complexity` = 0.01. The performance of this model is modest - 89% accuracy, with a FPR of 0.8 and FNR of 0.91. Per the decision tree, the most important variables seem to be `perCapsBody`, `isRe` and `bodyCharCt`.

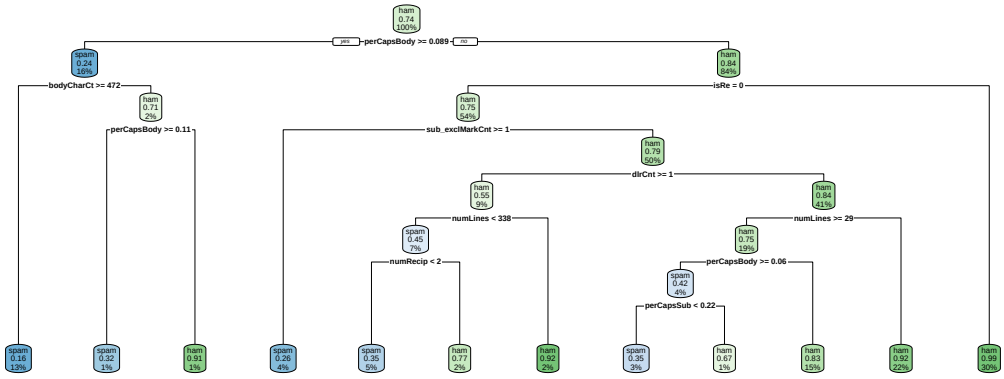


Figure 5: Decision tree build on the training dataset

Model #3: Random Forest

The random forest model has only one important hyperparameter to tune - mtry which is the number of random variables selected for each iteration. This is set to 5, using cross validation to minimize the test set misclassification error.

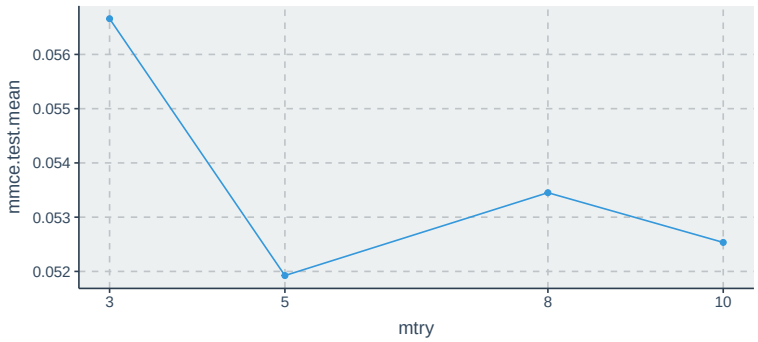


Figure 6: Hyperparameter tuning for ‘mtry‘ for random forest using 5 fold cross validation

The model performance is very good. Misclassification error rate for spam messages is 11% while that for ham messages is 1.7%, with an overall error rate of 4.3%. 200 trees are adequate for the errors to stablise. The variable importance plot shows similar variables to the decision tree.

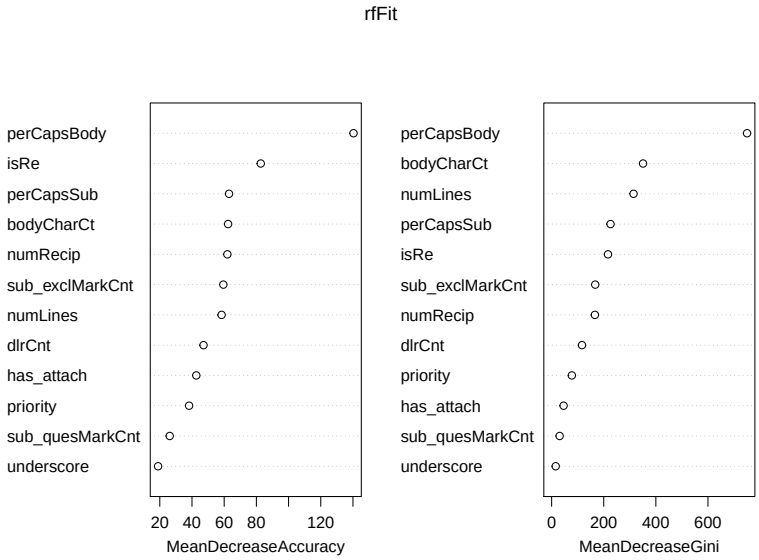


Figure 7: Variable importance plots for random forest model

Model #4: Logistic Regression

For this approach, a lasso logistic regression model is fit to the data. Lasso regression offers the advantage of inherent variable selection as coefficient estimates are allowed to go to zero. The only tuning parameter `lambda` which is set at 0.044 after running a 10 fold cross validation and using the 1-SE rule to decide the value. The graph below shows the results of the cross validation.

Table 4: Beta estimates for lasso model

variable	beta_estimate
(Intercept)	0.746
has_attach	-0.02
isRe	0.14
numLines	-
perCapsSub	-0.117
perCapsBody	-1.562
bodyCharCt	-
underscore	-
sub_exclMarkCnt	-0.044
sub_quesMarkCnt	-
dlrCnt	-
numRecip	0.08
priority	-0.012

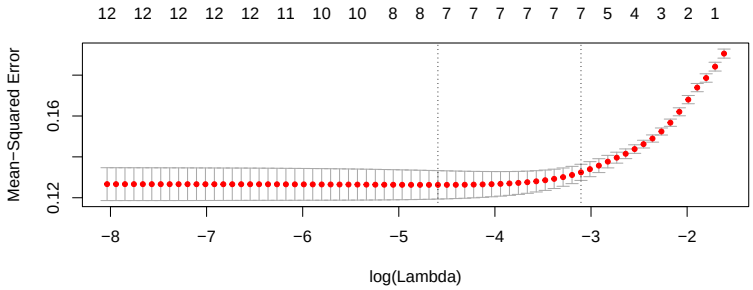


Figure 8: 10 fold cross validation result of search for optimal value of 'lambda' for lasso logistic regression model

The resulting lasso model eliminates 5 of the 12 variables. The model itself is of mediocre performance, with a misclassificiation error rate of 0.188. While it is possible to improve this model by introducing interaction terms, this was not investigated further.

Model #5: Support Vector Machine

The final model is a support vector machine classifier, in particular a kernel SVM classifier. There are a variety of kernels avaialble: linear, polynomial, radial, sigmoid, and for each kernel, there is a bandwidth and cost parameter which can be tuned. The choice of the hyper parameters is selected via 10 fold stratified cross validation based grid search across:

- `cost` : 1, 10, 15 or 20
- `gamma` : 0.1, 0.3 or 0.5
- `kernel` : linear, polynomial, radial or sigmoid

A total of 480 models are fit (made simpler to define, execute and manage via the `tuneParams` function in `mlr`). The best model was found to be for combination `cost` = 20, `gamma` = 0.5, and `kernel` = radial.

The performance of this model is decent, with a misclassification error rate of 6.8%, with a FPR of 2% and FNR of 20%.

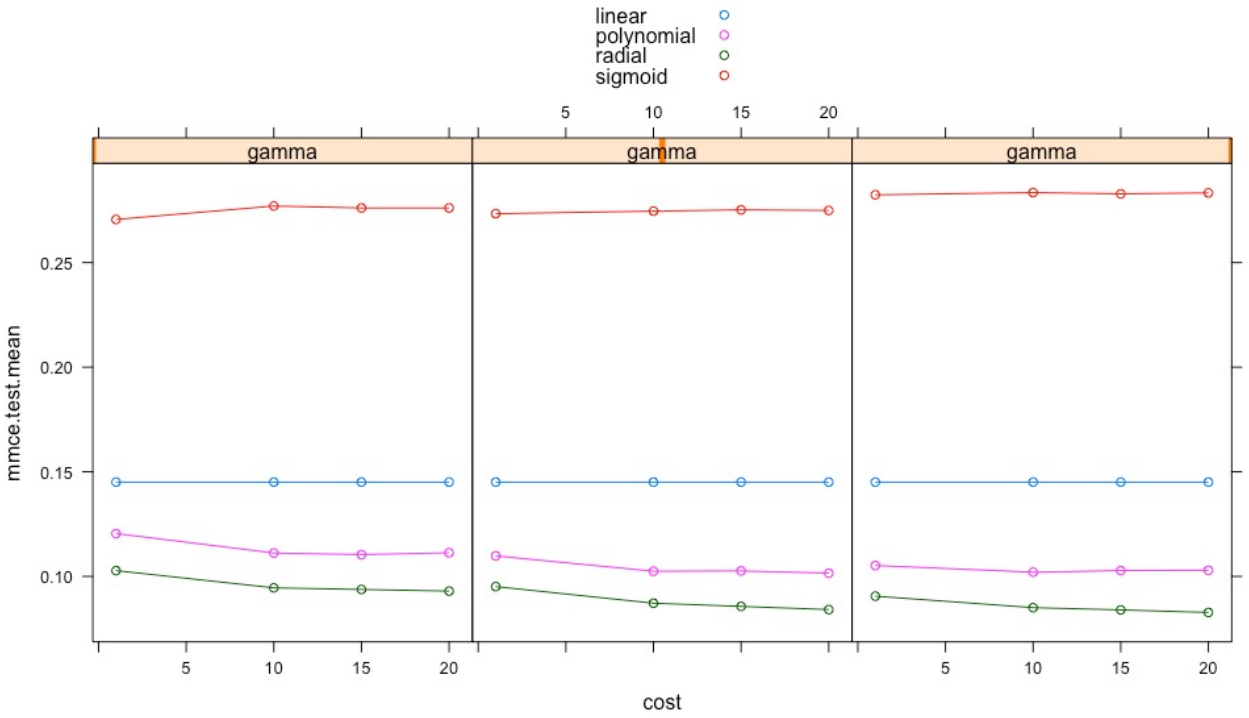


Figure 9: "Hyper parameter tuning results for SVM models, using 10 fold cross validation"

Model Comparison

Measures

To compare the models against each other, three measures are considered:

1. `mmce` : Mean misclassification error, defined as: mean(response != truth)
2. `fpr` : False positive rate, defined as percentage of misclassified observations in the positive class. Also called false alarm rate or fall-out.
3. `fnr` : False negative rate, defined as percentage of misclassified observations in the negative class.

The results discussed in the model comparison below are all for the *test* portion of the hold-out resampling strategy. Finally, the models are also compared by plotting their performance on the Reciever Operating Curve (ROC) and calculating their Area Under the Curve (AUC) values.

The point estimates for the test set performance are shown in the figure below. There is a large range of performance across the models. The plot shows the FPR vs the FNR. The sizes of the text are given by the misclassification rates. We can see that the NB model, even though the simplest of them all, out performs the remaining models across all three performance measures. Random forest model comes in a close second place.

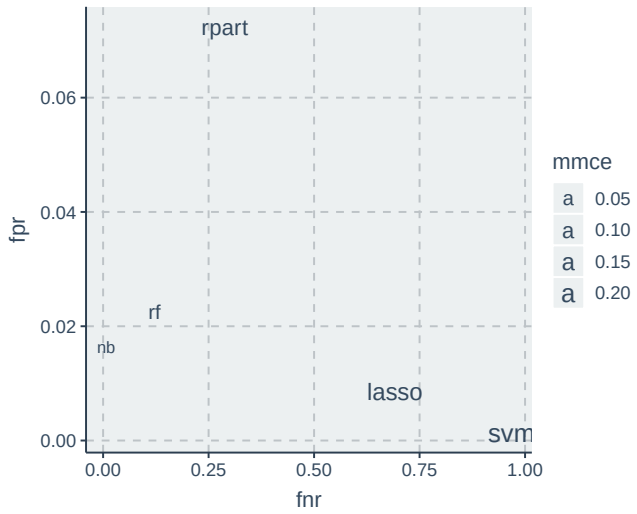


Figure 10: Point estimates of test set performance measures for all 5 models

The following chart shows the ROC curves for 4 of the 5 models. The ROC curve for the NB model wasn't able to be generated since the NB code was a custom code, as compared to the remaining models which use the `mlr` package's integrated interface. This chart confirms that the random forest model is the best performing, with an AUC of 0.98. SVM, rpart, and lasso models follow, with AUC values of 0.92, 0.91 and 0.89 respectively.

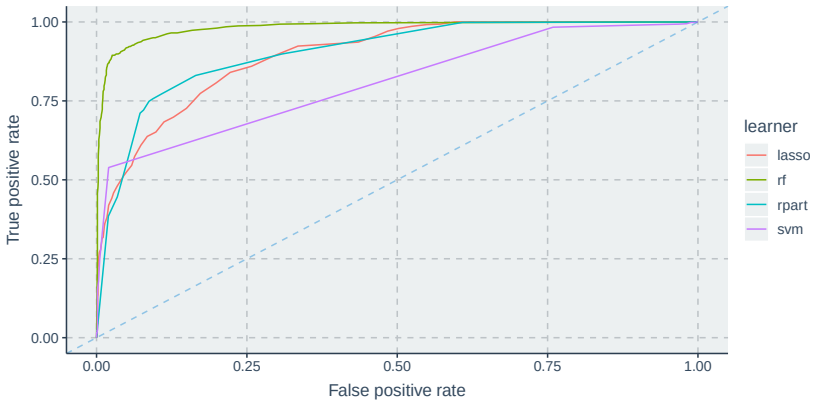


Figure 11: ROC curves for 4 of the models

The machine learning models using the engineered features could be improved by more feature engineering by exploiting data in the header and attachments. Some feature transformations like Weight Of Evidence could also add predictability for multi-level predictors.

Appendix

Piping of the data processing steps

```
> train_df %>%
>    get_processedHeader() %>%
>    get_contentTypes() %>%
>    get_attachInfo() %>%
>    get_isRE() %>%
>    get_numLines() %>%
>    get_perCapsSub() %>%
>    get_perCapsBody() %>%
>    get_underscore() %>%
>    get_subMarkCnts() %>%
>    get_dlrCnt() %>%
>    get_numRecipPriority()
```

Code Base

The code base is stored in github at this link