

Spam filtering

Alessandro Zanni and Israel Saeta Pérez, UPC
January 2012

Abstract

Spam is one of the major threats in the use of electronic mail nowadays. It consumes precious connection bandwidth, slows down mail servers and wastes people's time. In this project we study the performance of a variety of supervised-learning models including Logistic Regression, Naïve Bayes, Neural Networks, K-NearestNeighbours, and Quadratic and Linear Discriminant Analysis as well as their behaviour after a number of feature selection methods. Finally, we briefly compare our results with those of some previous studies using our same database.

I. INTRODUCTION AND DATA ANALYSIS

Our goal is to build a general classifier to determine if a certain mail message is spam or not according to a some statistics like the frequency of a certain words or the length of the longest word all in capital letters. False positives, i.e. marking good mail as spam, are extremely undesirable.

To build the classifier we make use of an e-mail spam database created in June and July, 1999, by Mark Hopkins, Erik Reeber, George Forman and Jaap Suermondt of Hewlett-Packards labs, consisting on a table of 4601 rows (representing the messages) and 58 columns (the variables). The first 57 ones are the input (explanatory) variables and the last one indicates if a mail is spam (equal to 1) or not (equal to 0).

The first 48 input variables are continuous real and indicate the frequency of certain words appearing in the mail messages. The next 6 ones are continuous real and express the frequency of certain symbols in the message. The last three input variables are of integer nature and indicate the average, longest and total length of uninterrupted sequences of capital letters in a message.

There are not missing values in the dataset. The class distribution is as following:

- Spam: 1813 (39.4%)
- Non-Spam: 2788 (60.6%)

The words and symbols whose frequencies and lengths are represented by the data variables are summarized in Table 1.

II. VALIDATION PROTOCOL

Here we describe the process followed to find the optimal model given the data (model selection) and assess the accuracy of the final model (model calibration).

We first divide the available data in two parts at random: 1/3 will be the test (holdout) sample and the remaining 2/3 will be the data used for training and validation. The holdout sample is never used for model selection nor feature selection/extraction. We keep the proportion of spam/nospam messages in each of the sets (40/60% approximately), i.e. we perform a stratified sampling to avoid any bias in the performance results coming from here.

We apply all potential models to the learning data set to compare their 10-fold cross-validation performances. Since we are interested in keeping the number of false positives

Table 1. Words and symbols present the input data

1. make	10. mail	19. you	28. 650	37. 1999	46. edu
2. address	11. receive	20. credit	29. lab	38. parts	47. table
3. all	12. will	21. your	30. labs	39. pm	48. conference
4. 3d	13. people	22. font	31. telnet	40. direct	49. ;
5. our	14. report	23. 0	32. 857	41. cs	50. (
6. over	15. addresses	24. money	33. data	42. meeting	51. [
7. remove	16. free	25. hp	34. 415	43. original	52. !
8. internet	17. business	26. hpl	35. 85	44. project	53. \$
9. order	18. email	27. george	36. technology	45. re	54. #

(legitimate messages classified as spam) low, we use the $F_{0.5}$ measure as a measure of the model performance, defined as:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_{0.5} = \frac{(1 + 0.5^2) precision * recall}{0.5^2 precision * recall}$$

The F_{β} measure is defined in a way it gives β times as much importance to recall as precision, so here precision is given twice as much importance as recall, and therefore favoring less false positives. It's worth noting that the F-measure of classifying everything as the majority class (non-spam) is zero.

After having trained and estimated the generalization (validation) error of all possible models with different functional forms, we select the model with the largest F-measure and re-train it using all the available learning data. To estimate the final (real) accuracy of our selected model we make use of the holdout sample.

III. DATA PREPROCESSING

As a first step, since not all the input variables have the same units (the first 54 ones are percentages while the next three ones are lengths) all variables are standardized so the last ones don't acquire non-deserved importance in the classification.

The non-spam messages come from the personal mail of George Forman, so it contains some word frequencies that are important for classifying his own email but not general legitimate messages: "hp", "hpl", "george", "650" and "857". (The last two ones come from the fact that George's phone number was 650-857-7835.)

To build a generic spam filter for every kind of mail, which is what we want, we don't need to take care of the frequency of the Georges, or HP's words, so they're filtered out of the dataset. After this first "manual" filtering we use a number of feature selection algorithms to reduce the set of attributes, including both filters and wrappers: the correlation- and consistency-

based feature selection, backwards-elimination using AIC over a logistic regression and random forest. We will study the performance effect of all these later.

The purpose of this feature selection is two-fold: first, try to reduce the amount of noise to help those algorithms where it affects more negatively (the curse of dimensionality) and second, improve their efficiency, making them run faster. For some algorithms this dimensionality reduction might result in a worse model in terms of accuracy. We will see this in the results.

Correlation-based Feature Selection

The correlation-based feature selection (CFS) algorithm is a filter that evaluates subsets of attributes and returns the best subset on the basis of the following hypothesis: "Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other". Running this algorithm on our data set reduces the number of attributes from 52 to only 9.

The formula obtained is:

```
class ~ w_f_remove + w_f_free + w_f_your + w_f_000 + w_f_money +  
w_f_labs + c_f_..3 + c_f_..4 + c.r.length_mean
```

Consistency

This algorithm is based on the consistency measure [1] according to which an attribute subset is consistent if there don't exist two instances with exactly the same attribute values but different class labels. Using this measure the feature selection is formalized as finding the smallest set of attributes that can distinguish classes as if with the full set.

Our implementation (from FSelector) uses the best first search strategy and discretizes the data (since this measure only works with discrete data by definition) with the Minimum Descriptive Length (MDL) method. Using it the number of attributes is reduced from 52 to 23.

The formula obtained is:

```
class ~ w_f_make + w_f_our + w_f_over + w_f_remove + w_f_internet +  
w_f_mail + w_f_will + w_f_free + w_f_business + w_f_you +  
w_f_credit + w_f_your + w_f_money + w_f_85 + w_f_pm + w_f_re +  
w_f_edu + c_f_..1 + c_f_..3 + c_f_..4 + c.r.length_mean +  
c.r.length_longest + c.r.length_total
```

AIC Optimization using Logistic Regression

This method is a wrapper that performs a backward-elimination of attributes using the Akaike Information Criterion (AIC) over a logistic regression. The AIC is basically the deviance of the fitted model with an extra penalization for the number of attributes (regularization).

Using this algorithm the number of attributes is reduced from 52 to 39. In the following we will refer to this set of attributes as the logistic attributes subset.

The formula obtained is:

```
class ~ w_f_address + w_f_all + w_f_3d + w_f_our + w_f_over +  
w_f_remove + w_f_internet + w_f_will + w_f_free + w_f_business +  
w_f_email + w_f_you + w_f_credit + w_f_your + w_f_font +  
w_f_000 + w_f_money + w_f_lab + w_f_labs + w_f_telnet + w_f_data +  
w_f_415 + w_f_85 + w_f_1999 + w_f_pm + w_f_direct + w_f_cs +  
w_f_meeting + w_f_original + w_f_project + w_f_re + w_f_edu +  
w_f_conference + c_f_. + c_f_..3 + c_f_..4 + c.r.length_mean +  
c.r.length_longest + c.r.length_total
```

Random Forest

This algorithm generates multiple decision trees (1000 in the implementation we use) using different random subsets of attributes and instances sampling and calculates the averaged “importance” of each attribute based on performance measurements in each tree.

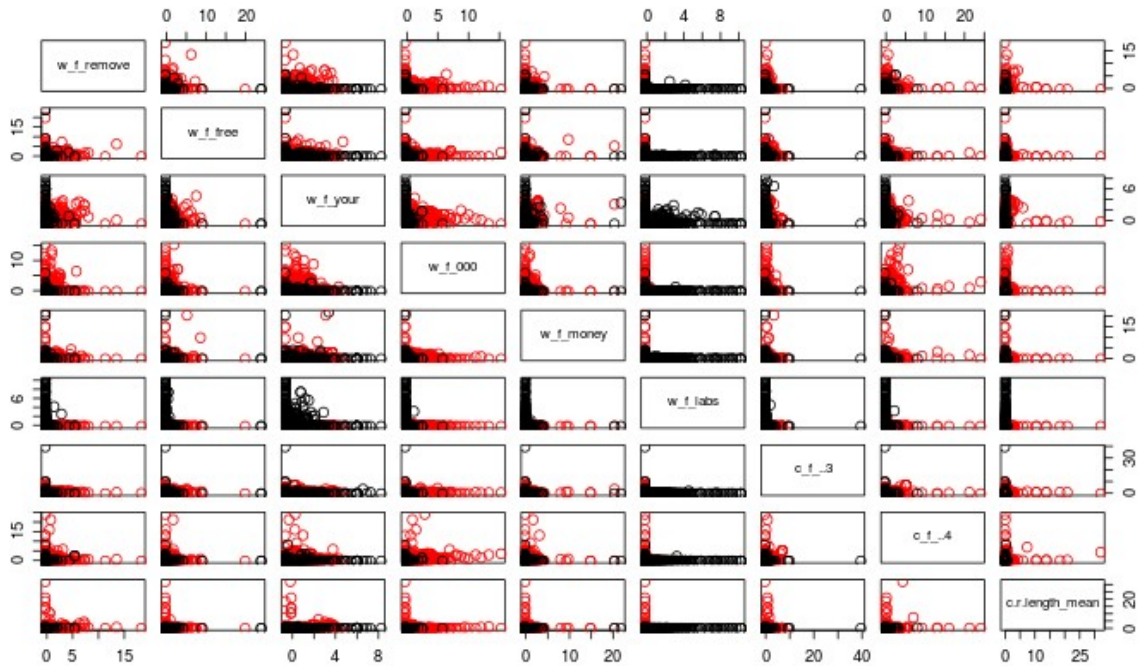
We can then sort these attributes descendently by importance. To decide where to cut we tested the performance of the different models keeping a varying number of attributes (28, 29, 30, 31, 32...) but the best performance was always obtained with 30. So we keep this number for all tests done later with this feature selection method.

The formula obtained is:

```
class ~ c_f_..3 + c_f_..4 + w_f_remove + w_f_free + c.r.length_mean +  
c.r.length_longest + w_f_1999 + w_f_your + c.r.length_total +  
w_f_edu + w_f_our + w_f_you + w_f_re + w_f_000 + w_f_money +  
c_f_..1 + w_f_85 + w_f_font + w_f_labs + w_f_business + w_f_will +  
w_f_internet + w_f_meeting + w_f_email + w_f_pm + c_f_. +  
w_f_over + c_f_..2 + w_f_all + w_f_telnet
```

IV. DATA VISUALIZATION

We construct a pairs plot to visualize the relation of the different variables coming from the CFS with the classification.



As we can observe, the frequency of the word “lab” is a good measure to separate spam from non-spam, paired with any other variable. This is probably related with the fact that many spam messages are related to black-market drugs selling. The rest of the pairs don't allow to separate the different classes very efficiently by themselves.

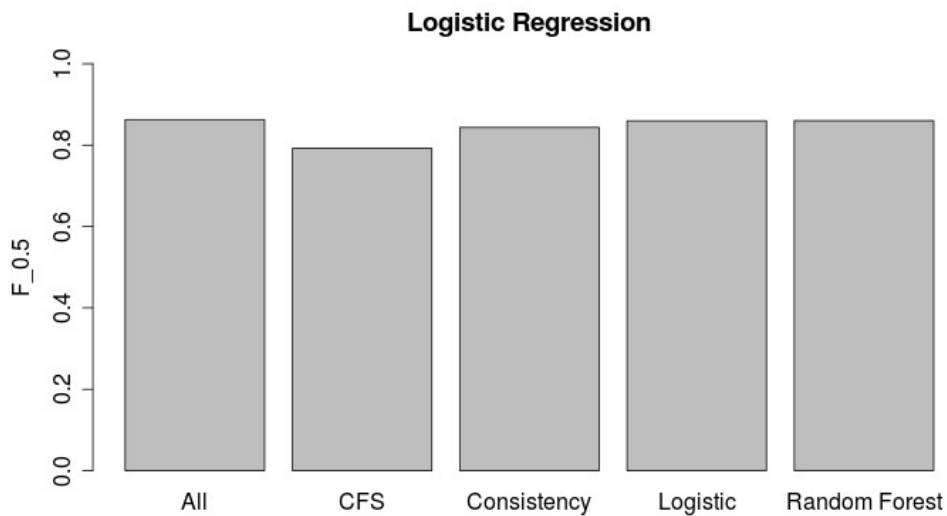
We also perform clustering with two classes to see if the examples of one class are closer one to each other (in the selected input variables space, with CFS). Using *kmeans* with $k=2$, the results show that only an 11% of the total inertia is “inertia between”. This suggests that the spam and non-spam can't be easily separated in two class just by attribute proximity and the knn model for $k=2$ might give poor results.

V. MODEL EVALUATION AND SELECTION

Logistic Regression

The logistic regression estimates the probability of a certain event to occur (a message to be classified as spam, in our case) based on the fit of the available data to a logistic function. The more attributes we have, the more degrees of freedom this method has to fit the data better, but also the larger possibility to overfit.

Subset	All	CFS	Consistency	Logistic	Random Forest
Logistic Regression	0.8629	0.7923	0.8459	0.8598	0.8602

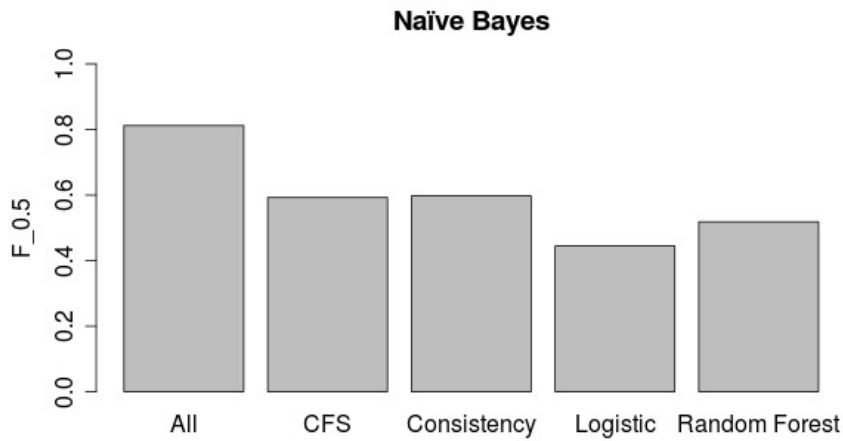


We can see that the Logistic Regression performs fairly well in general and it's quite stable on the deletion of some of the attributes. This is so because the most significant attributes are probably the ones with highest weights (regressors) in the model and a feature selection method is unlikely to remove the most significant attributes. Therefore, when the less relevant attributes are removed the output of the model doesn't vary too abruptly.

Naïve Bayes

This is a simplistic model that estimates the class of a message given its attributes using the Bayes' rule and of works on the (naïve) assumption of that the different attributes are all independent given the class. Surprisingly, it frequently provides acceptable results.

Subset	All	CFS	Consistency	Logistic	Random Forest
Naïve Bayes	0.8121	0.5925	0.5974	0.4446	0.5185



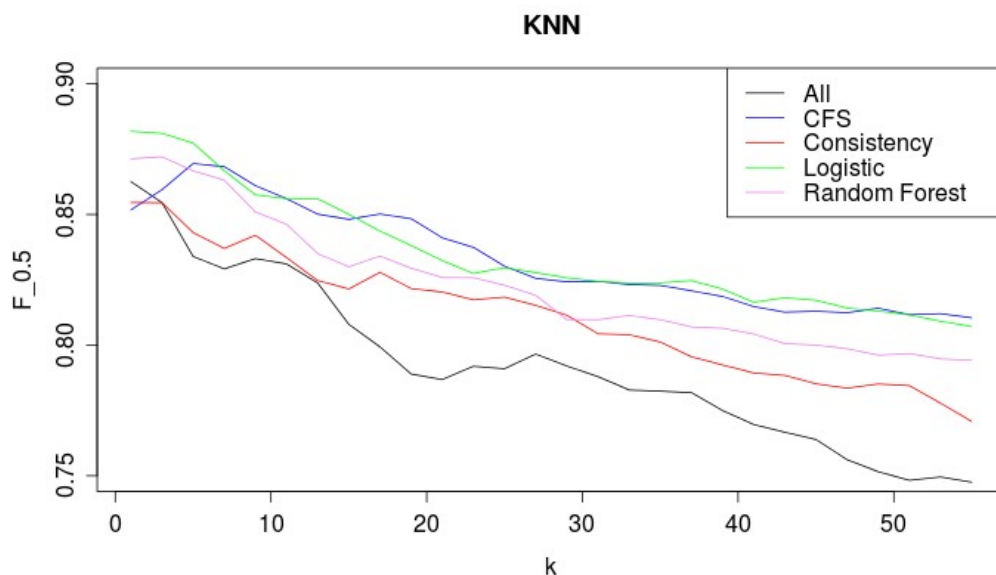
As we can observe, Naïve Bayes generates acceptable results when working with all the attributes but its performance drops rapidly when dealing with a subset of them.

KNN

K-Nearest-Neighbours is a simple but effective method that takes the k neighbours closer to the one we want to classify and performs a voting to decide the class. There are multiple ways to calculate the distances (Euclidean, Cosine Similarity, Jaccard Distance) and the vote can be weighted according to this distance, but our implementation (from R's "class" package) uses Euclidean distance and majority unweighted vote.

The k 's we use range from 1 to 55 (the square root of the number of examples in the training set) in steps of two. We use always an odd number of neighbours to avoid ties.

Subset	All	CFS	Consistency	Logistic	Random Forest
k	1	5	1	1	3
KNN	0.8626	0.8695	0.8546	0.8819	0.8721



As we can see, the models using only the selected attributes perform much better in average than the ones with all the attributes. This means that the similarity to other examples related to non-important attributes adds harmful noise here, i.e. having a close-to-average frequency for widely-used words doesn't add information when it comes to classification. One way to overcome this problem is to use other measures instead of the frequency of the word in the message, like the TF-IDF weight, which favours unfrequent words.

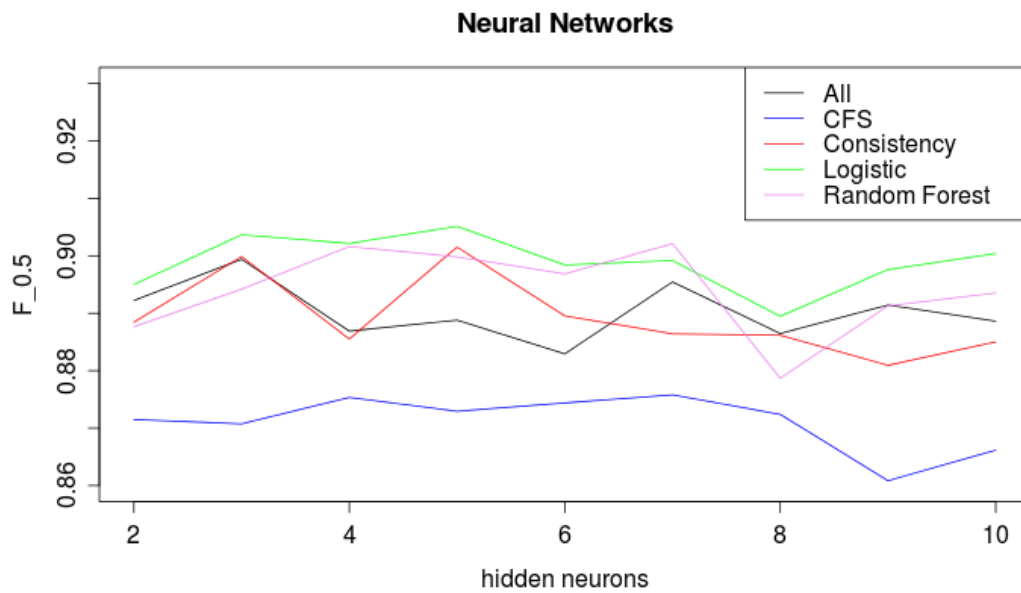
Also, the figure above shows that increasing the number of neighbours considered affects the performance negatively in general.

The best model corresponds to $k=1$ and the logistic subset.

Neural network

This method features a single-hidden-layer feed-forward neural network trained using the BFGS method. The number of neurons in this hidden layer is optimized in the range from 2 to 10 using 10-fold cross-validation with the F-measure.

Subset	All	CFS	Consistency	Logistic	Random Forest
Hidden units	3	7	5	5	7
F_0.5	0.8994	0.8758	0.9015	0.9051	0.9021



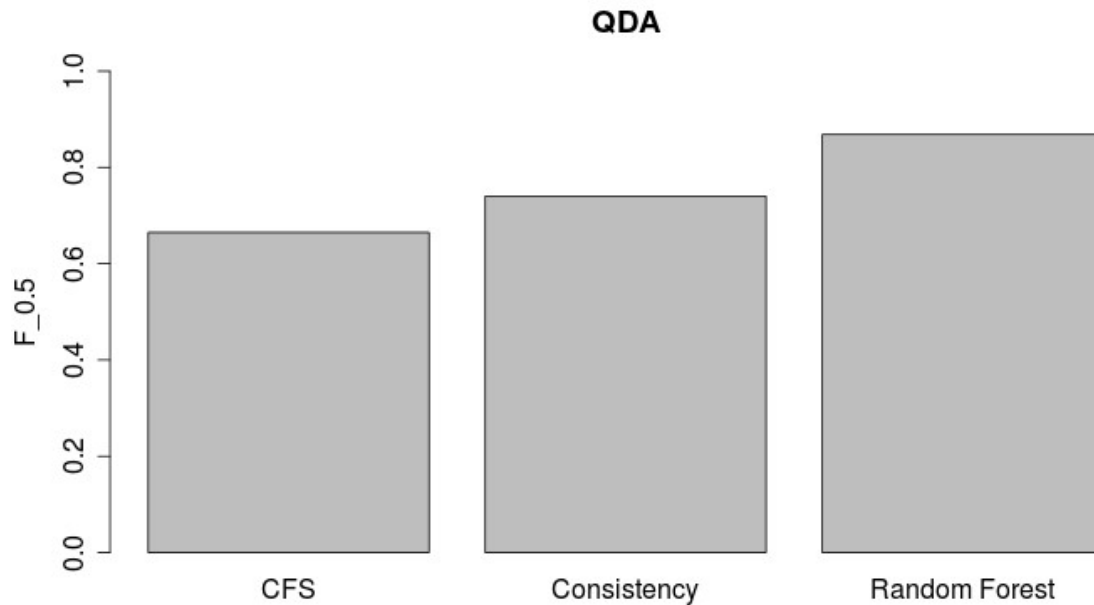
In this case the models with a subset of attributes performed as well or better than the models using all of them, with the exception of the CFS subset. However, these differences are not very large. Varying the number of neurons affects the performance slightly but we cannot observe a general tendency.

The best results correspond to a neural network of 5 hidden units using the logistic subset.

Quadratic Discriminant Analysis

This algorithm builds a quadratic separation hyper-surface assuming that the attributes follow a Gaussian distribution and using Bayes rule and basic linear algebra.

Subset	All	CFS	Consistency	Logistic	Random Forest
QDA	-	0.6649	0.7402	-	0.8685



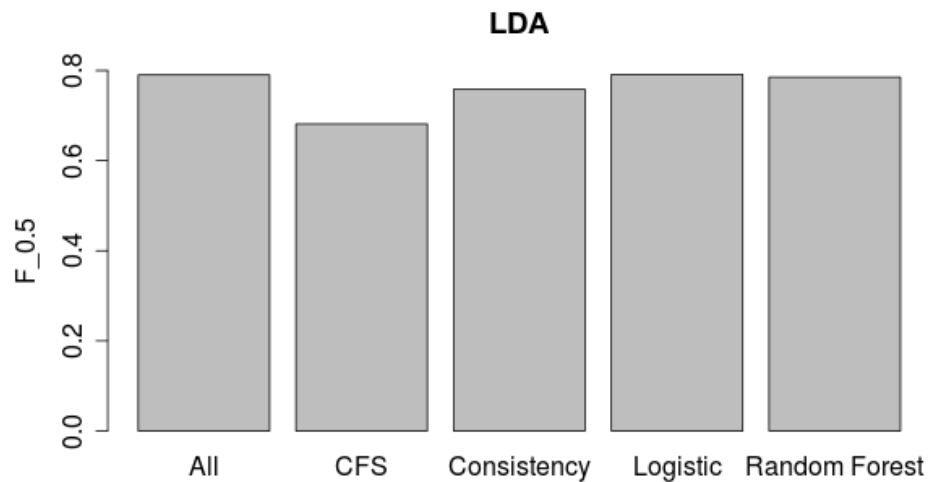
The algebraic solution involves calculating the inverse of the covariance matrices for each class and the spam ones reported to be singular (up to the precision of the inversion routines) for the cases of all the attributes and the logistic subset, so we omit the results of those.

We can see that using the random forest subset we obtain a really high performance (close to 0.87). At contrary, using the CFS selection, we have a result close to 0.6. This means the random forest selection has some important value useful in the QDA that the CFS doesn't have. Or it can be also be because the random forest subset has lots of variables whereas the CFS one use only 11.

Linear Discriminant Analysis

This is the same as QDA but here the covariance matrices for all class-conditional distributions are assumed to be equal. This of course doesn't have to be true, but reduces the complexity of the calculations and can still provide reasonably good results.

Subset	All	CFS	Consistency	Logistic	Random Forest
LDA	0.7903	0.6815	0.7585	0.7913	0.6815



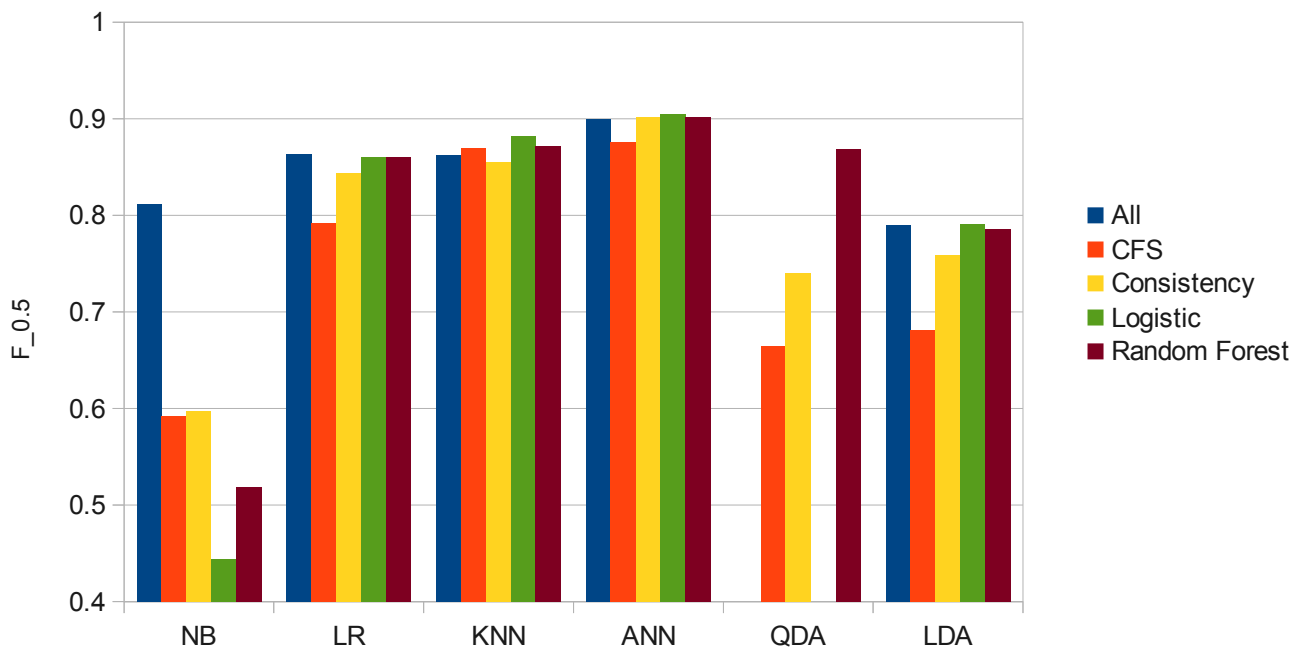
As we can see the performances obtained, all results are quite similar. And the F_{0.5} measure with all the attributes, and with the Logistic and Random Forest subsets are similar. This is quite the same result than the Logistic regression model so the explanation can be similar. One or more significant attributes are probably situated on the highest weights in the model and the feature selection method has unlikely removed it, so the output of this model doesn't change a lot and are quite similar.

Comparative results

The obtained F-measures for the different models are summarized in the following table and figure.

Model/Subset	All	CFS	Consistency	Logistic	Random Forest
Logistic Regression	0.8629	0.7923	0.8459	0.8598	0.8602
Naïve Bayes	0.8121	0.5925	0.5974	0.4446	0.5185
KNN	0.8626	0.8695	0.8546	0.8819	0.8721
Neural Network	0.8994	0.8758	0.9015	0.9051	0.9021
QDA	-	0.6649	0.7402	-	0.8685
LDA	0.7903	0.6815	0.7585	0.7913	0.6815

Performance comparison



As we can observe, the model with the best F-measure is the neural network using the logistic subset, with 5 hidden units.

The AIC optimization using Logistic Regression is the feature subset selection method that provided better performance results in average, with the notable exception of Naïve Bayes, where it's the one that performs the worst. On the other hand, CFS was the one with less success in general.

VI. MODEL CALIBRATION

We can now test the neural network model accuracy using the holdout sample. The results are the following.

Test Set confusion table

predicted / true	non-spam	spam	correct
non-spam	875	55	94.09%
spam	55	550	90.91%
correct	94.09%	90.91%	

Performance results comparison

	Cross-Validation	Test Set
F-0.5 measure	0.9051	0.9091
Accuracy	93.48%	92.83%

Surprisingly, the percentage of false positives (9%) is larger than the percentage of false negatives (6%), although we used the F-0.5 measure to try to reduce the number of false positives. We could insist on reducing this number by using a F-measure that gives even more importance to precision, or use directly precision, at the cost of reducing the recall and ending up with a lot of spam messages in our mailbox.

VII. RESULTS COMPARISON

It is interesting to compare our obtained accuracies with the ones claimed in previous studies using our same spam database. For example, the description information coming with the data set reports an approximate 7% of misclassification error, in agreement with our results.

The report [2] shows a result of 93.8% accuracy using a 5-hidden-layers MLP. This is slightly better than our result but comes with the complexity of added hidden layers, probably making the model less capable of classifying more general new incoming mail.

In [3] a feed-forward neural network with one 7 hidden units in a single hidden layer is trained using all the attributes, gradient descent and 5-fold cross-validation, leading to an average accuracy of 91.44%, also very close to our results.

In [4] a neural network is built using an interesting mechanism called “abductory inductive mechanism” that is capable of perform attribute selection by itself, among other amazing features. Using one single training/evaluation set split they achieve an accuracy of 91.7%.

In conclusion, our results show a similar or even better performance than similar previous works.

REFERENCES

[1] *Consistency Measures for Feature Selection: A Formal Definition, Relative Sensitivity Comparison and a Fast Algorithm*. Kilho Shin, Danny Fernandes and Seiya Miyazaki. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.

[2] *Spam? Not Any More! Detecting Spam emails using neural networks*. Thiagarajan Sivandayan. ECS/CS/ME 539 Project.

[3] *Anti-Spam Filtering Using Neural Networks and Bayesian Classifiers*. Yue Yang and Sherif Elfayoumy. Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation.

[4] *Spam Filtering with Abductive Networks*. El-Sayed M. El-Alfy and Radwan E. Abdel-Aal. 2008 International Joint Conference on Neural Networks.