Data Mining - Homework 3

Dmitry Donetskov (ddonetskov@gmail.com)

May 5, 2018

1 First Part

The dataset contains information whether a textual message was fake or real. Building a classifier for it is a supervised learning text processing workflow. I have used the following text mining pipeline to train and score different classifiers (models):

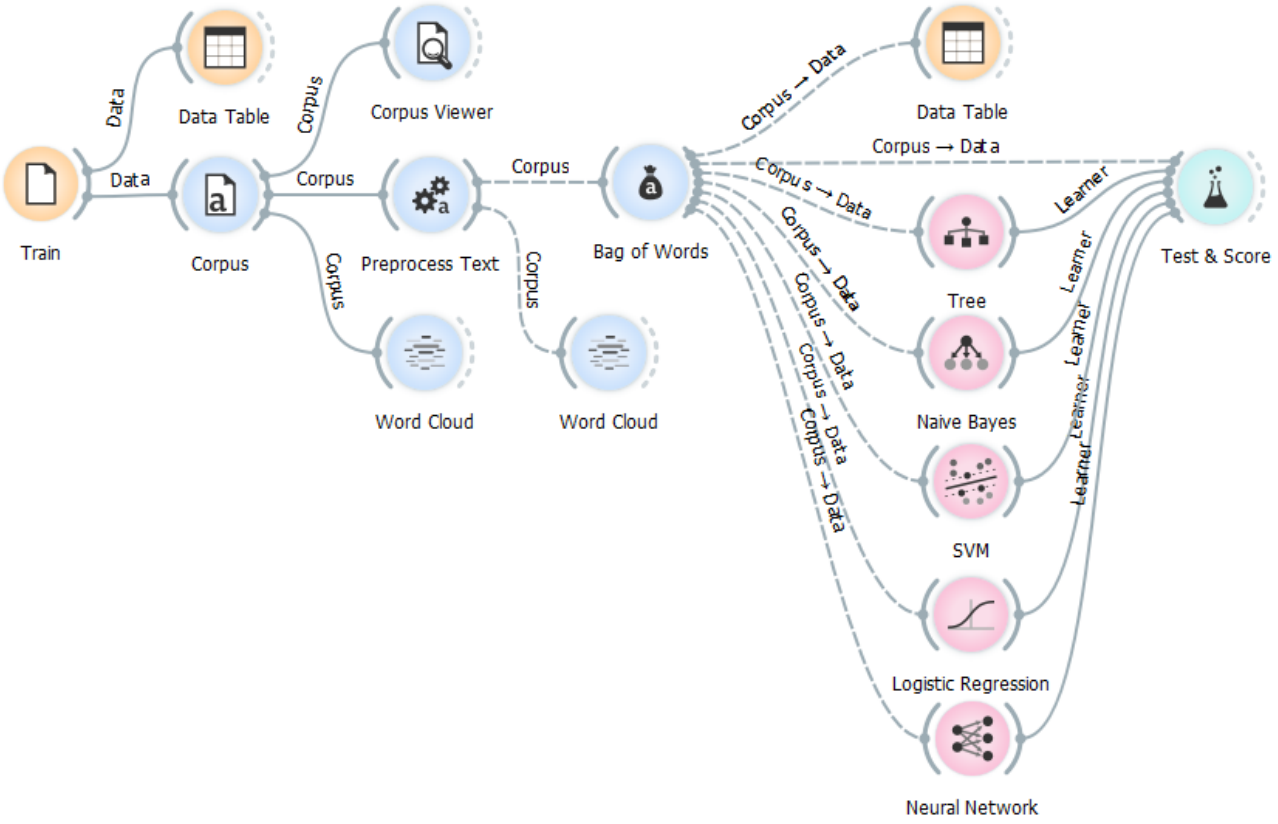

Figure 1: Workflow with Test & Score (CV, k=10)

Let's consider major factors that could influence the model performance:

- **Corpus**. There are two textual features in the dataset: *title* and *text*. It seems to be intuitively wise to use either only *text* or both the features. However, I also evaluate a model with only *title*.
- **Preprocess Text**. The typical configuration is used here, consisting of the steps:
  - **Converting to the lower case**: I presume a same word but in different cases still carry same meaning (which is a latent factor for the dependent variable) therefore I transform all words to the same case (the lower one).
  - **Removing URLs**: URLs naturally tend to be unique as they are basically unique addresses, including them into the model may bring the risk of overfitting.
  - Tokenization by splitting into words: the data is textual, the basic way of working with it is to break sentences into words.
  - **Removing stop words and punctuation**: I checked Corpus based on the original data with Word Cloud, there were a lot of very frequent words like 'from/of/the/to', I don't believe they add much (if any) of information therefore considering them as stop words and removing them along with the punctuation symbols,

  - **Removing outliers**: I remove very rare and very frequent words, which have the document frequency's value outside the 10^{th}-90^{th} percentiles).

- **Bag of Words**:
  - **Term Frequency**: I use 'Count',
  - **Document Frequenc**: I try both None and IDF.

- **Algorithm**: I have chosen Tree, Naive Bayes (NB), SVM, Logistic Regression (LR) and Neural Network (NN). I doubt Tree and SVM are good choices as they are not seemingly designed for large number of features, I am going to check that.

The results are presented in Table 1. There are few interesting things to note:

- As per the case 1, It's possible to obtain decent results based only on *title*. Which might be useful if we want to derive at least something from even that limited source of information.
- As per the cases 2 and 3, there is a minor difference between choosing the total number of types either as the one hundred most frequent ones or the ones falling into the 10^{th}-90^{th} percentiles.
- As per the cases 3 and 4, adding *title* to *text* does not almost improve the classifier's accuracy. Including *title* adds only four new types, and should change DF/IDF, but not to any considerable information gain.
- As per the cases 4 and 5, there is no difference between either using IDF or not using it for the term frequency across the corpus. It's a bit strange and probably due to the fact we drop the outliers during the pre-processing stage.
- As per all the cases, SVM is not good. There are probably just too many features for it.

Table 1: Classification Accuracy for Different Configurations (Ten folds CV)

	Model Parameters				Classifier Accuracy				
No.	Used Text Features	Preprocessing Options	Bag of Words Options	Total Types	Tree	NB	SVM	LR	NN
1	title	lowercase, remove URLs, words, stopwords, 100 most frequent	TF*IDF	100	0.561	0.729	0.439	0.734	0.705
2	text	lowercase, remove URLs, words, stopwords, 100 most frequent	TF*IDF	100	0.753	0.723	0.506	0.807	0.808
3	text	lowercase, remove URLs, words, stopwords, DF = 0.1-0.9	TF*IDF	509	0.753	0.737	0.618	0.826	0.873
4	title, text	lowercase, remove URLs, words, stopwords, DF = 0.1-0.9	TF*IDF	513	0.757	0.738	0.562	0.839	0.859
5	title, text	lowercase, remove URLs, words, stopwords, DF = 0.1-0.9	TF*None	513	0.757	0.738	0.674	0.839	0.868

1.1 Best Classifier

The good classifiers look to be those based on Logistic Regression and Neural Network. Given they are on par, I'd choose the former from them as it can provide additional information about how it works as it's an opener model than Neural Network.

As to the model parameters, I'd keep those from the case 4. Computationally, it's the most costly one however these parameters look to be right, they make the model to be more universal: taking more than just one hundred most frequent types and using the IDF correction should help with the test data.

1.2 The Most Important Parameters

Nomogram was used to identify which parameters the Logistic Regression classifier found to be the most important. Figure 2 shows the top five of them.

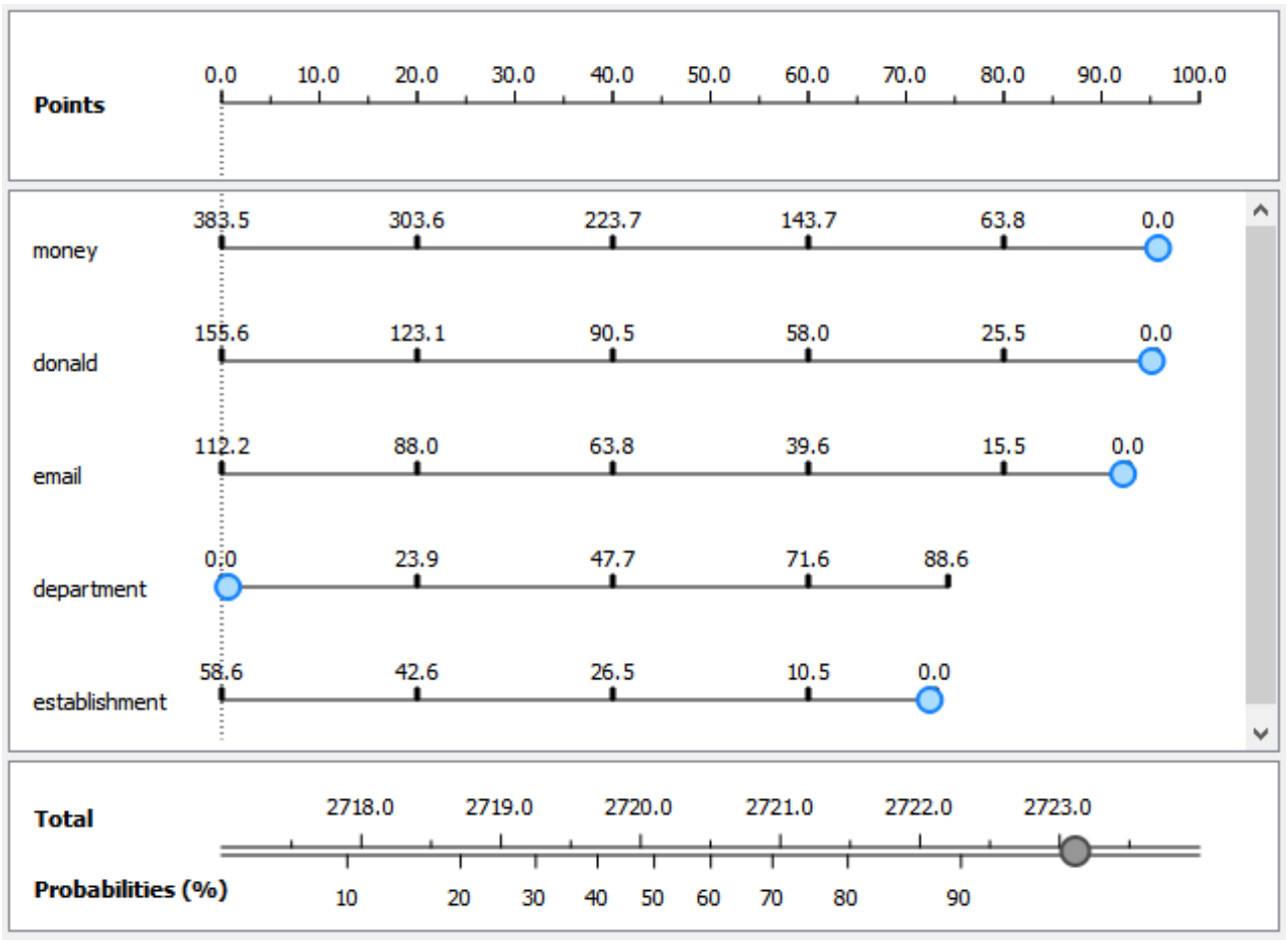

Figure 2: Top 5 Most Important Types (Target = TRUE)

2 Second Part

Putting fake-test.tab through the same text preprocessing steps to Test & Score as the test data gives Classification Accuracy (CA) indicated in Table 2. CA is shown for all classifiers just out of curiosity, CA for the chosen Logistic Regression classifier is marked with bold.

The model looks to be adequate as it shows the same CA on the test data as it was on the train data. Unfortunately, it's not so accurate as let's say 90-95% but that's probably due to the somewhat limited amount of data. Still, this level of accuracy might provide advantages in the real life.

Table 2: Classification Accuracy for Chosen Configuration (Evaluation with Test Data)

	Model Parameters				Classifier Accuracy				
No.	Used Text Features	Preprocessing Options	Bag of Words Options	Total Types	Tree	NB	SVM	LR	NN
4	title, text	lowercase, remove URLs, words, stopwords, DF = 0.1-0.9	TF*IDF	train=513 test=564	0.742	0.724	0.520	**0.825**	0.840

3 Ideas for Future

I am putting here some ideas I had in mind while working on the exercise but due to limited resources I did not try them:

- Explore what URLs are contained in the data (if any at all), check if there are multiple occurrences of same URLs which is unlikely but still worth checking as it might be a useful feature to improve the model.
- Try stacking several models to see if that can improve the total accuracy (e.g. stacking both Logistic Regression and Neural Network).
- Check what important types were chosen by Naive Bayes, they should be different from those chosen by Logistic Regression as Naive Bayes does not take relationship between variables into account.
- Explore analytically into why SVM does look to be an adequate algorithm for this kind of task.