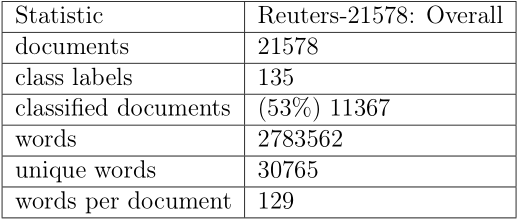
**1. Initial Statistics:**  
The reuters dataset has 21578 rows and 140 variables. Every single row corresponds to an individual document. Almost each document has a title and a text. Also, every document has got details about whether it belongs to the train data set or the test data set or is not used at all. Altogether, there are 135 different topics which have been specified in this data set. Further, a document either belongs or does not belongs to a specific topic which means each topic is a dichotomous variable i.e. has got the value 0 or 1. Also its much likely that one document can belong to multiple topics and the case is similar over here.

In the given dataset there are around 9471 documents without any assigned topic. Statistics for the dataset is as follows:  


Almost half of the document do not have any topics. Similarly, some docs do not have title and some do not have text.

**2. Preprocessing:**

R offers a variety of packages for doing text processing however I have made use of **tm package.** This package is simple to use for data importation, corpus handling, preprocessing, and even the creation of term-document matrices.  
**2.1 Title and Text Analysis**Before progressing with the text preprocessing, the title needs to be handled, also the main text of the document. So, the title is added at the beginning of the text of each document. It's done because we desire to keep the entire information from each document. As few documents don’t have either text or title, placing the title at the beginning of each text seems to be the right way to manage this problem.  
Further, the id of the companies is extracted, followed by replacing the non-apostrophe punctuation with spaces. The output obtained from these steps is converted into character strings and then other tasks like removing strings from company tags, replacing the spaces between company tags. Then other activities like removing duplicate organization identity from main doc is done. This is followed by the basic word processing activities like converting the text to smaller size, removing punctuations, numbers, stop words of English, and whitespace etc. is done. I used the predefined dictionary from our package to remove the stop words. Finally, all the characters and organization id are binded together.   
  
Then we implement the document tagging with respect to the topics available. First task here involves tagging the topics for which a function is implemented. In the function a for loop loops over over columns 4 to 138 for fetching documents with that tag applied. The untagged coulmns are ignored while the desired topics are tagged. Further all the documents that do not relate to the topics on the list are tagged as other and the data is generated with the assigned topic tags. The desired topics are earn, acq, money.fx, grain, crude, trade, interest, ship, wheat and corn.

**3. Feature Engineering:**

Now that we have cleaned the documents, we can go to the next step: creating feature representation of documents. We are going to build build structured data from text which is a unstructured data. Feature engineering is known as an important part of text classification. There are many ways to do it. What's worth mentioning is after the preprocessing step, 431 observations have null representation. We remove these observations. These documents represent approximately 1.6% of the data set.

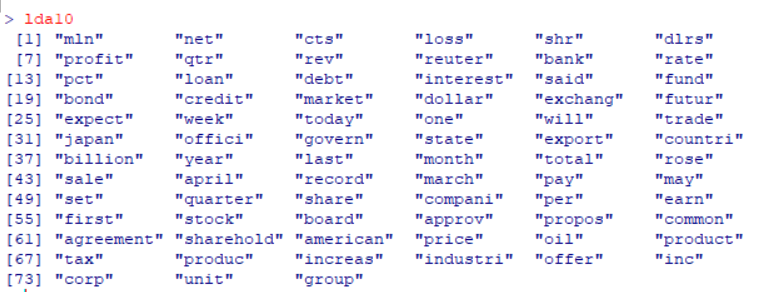
**3.1 LDA (Latent Dirichlet allocation)**

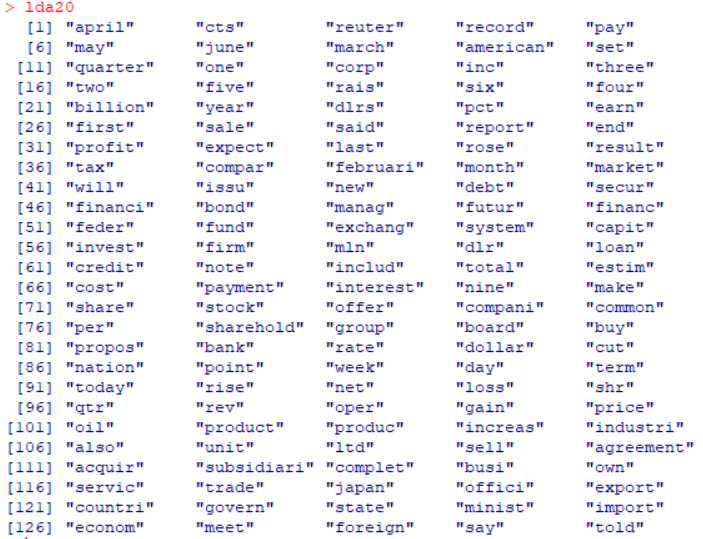
LDA is a generative model which allows sets of observations to be explained by unobserved groups which explains why some parts of the data are similar. It talks about the order of the words in the sentence. I used the R package called ’topicmodels’. There is a function inside named ’LDA’ which can find k abstracts group using VEM or Gibbs algorithm. Given that VEM and Gibbs algorithm give us approximately same results, we decide to use here the VEM algorithm. The biggest query of this function is to find the best k appropriate to our data set. k is the number of group that the function needs to create. We will assume that due to the 10 topics, we will put k as 10.

First, I build a bigram tokenizer function so that the result includes both the unigrams & bigrams. Then, a DocumentTermMatrix is made from the Corpus. As LDA does not works great with sparse terms so remove the sparse terms. Further, I generate the topic models. Before training the LDA, we remove the noise documents from the training set and the empty rows also. Finally, we run the LDA for 10 topics and get the probabilities to use as feature.

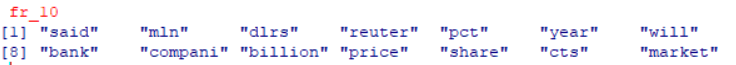
Then, we generate the single labelled feature matrix. In order to do so, DTM, LDA probabilities and topic tags are combined into a data frame, making sure that topic is a factor for classification and the training set is extracted. Further, I have tried to implement some additional features like fetching the 10 and 20 most frequent per LDA topic cluster.

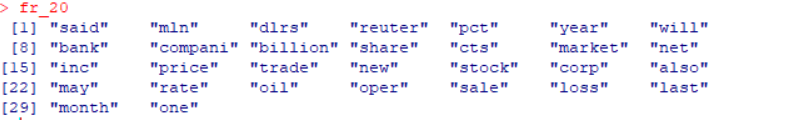
I have implemented N most likely terms per **LDA topic cluster**. A screenshot for 10 and 20 most frequent terms per topic are:



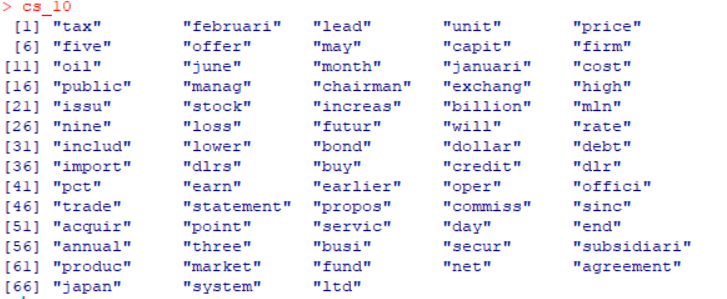


Further, we extract the 10 and 20 most frequent terms per topic using the **document term matrix.** A screenshot for 10 and 20 most frequent terms per topic are:



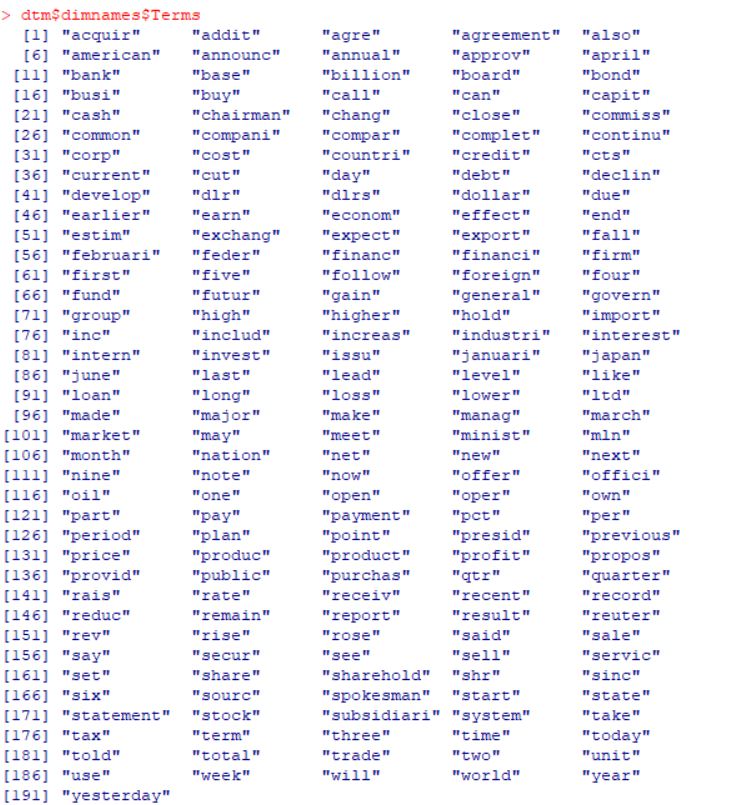


Then, I have implemented a function to fetch the 10 and 20 most significant terms per topic **by Chi-Squared value.** Their details are as shown below:





The feature **all terms** i.e. the list of 191 terms under the Document Term Matrix(DTM) is as follows:



**4. Text Classification**

I have the three classifiers in this exercise: naiveBayes, random forest and SVM. The three first steps will use the training data. Our goal will be to find the best classifier in order to use it on the test data. I will first find the best parameters of these algorithms. Afterwards I will look for which one feature engineering is the best in order to do prediction. Then, I will find which is the best classifier. And finally, I will use the test data and see what kind of result I get.

**Building the model:**

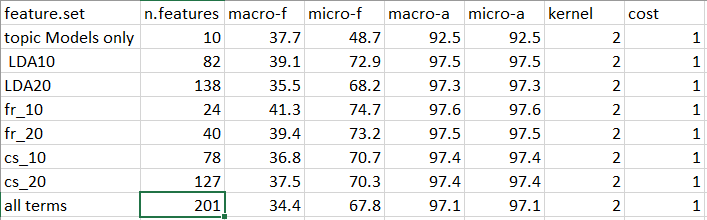
I focus on the ten most popular topics. I will be building one model for each topic thus 10 models will be built all together. Thus, I will be able to build a big model in order to predict a whole matrix of topics prediction. Hence, I will be able to compute per topic and for the whole matrix prediction.

**10 Cross Validation:**I use cross validation to find the best parameters, the best feature engineering and the best classifier. So, I have split the training set in 10 new little training set and 10 test set.

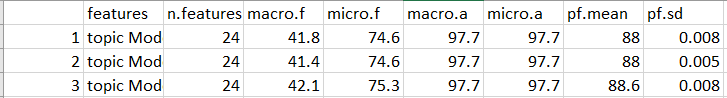
**Analysis for best feature engineering:**

The analysis of classification using 10 cross validation has been summarized with tables below. However, in order to not get a huge report, we will take only important result.

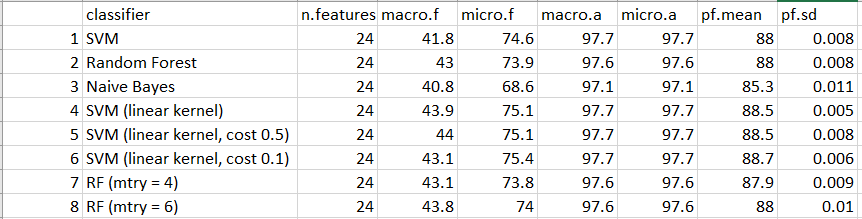
I have implemented a function to measure the performance measure per class. It calculates the confusion matrix and, also Recall, Precision, Fmeasure and accuracy. It also calculates the microaveraged performance. Then for the first pass the data is split into 80:20 ratio for classification for training and test sets. Then a model is built using SVM and model is executed to compare predictions to the expected values. The feature set considered are topic models, lda10 , lda20,fr10,fr20, cs10,cs20 and all terms.  
**Performance across the extra feature sets has been shown below**:



Checking Performance making use of K-Fold Cross-validation which has been implemented with respect to both naïve bayes and random forest classifier:

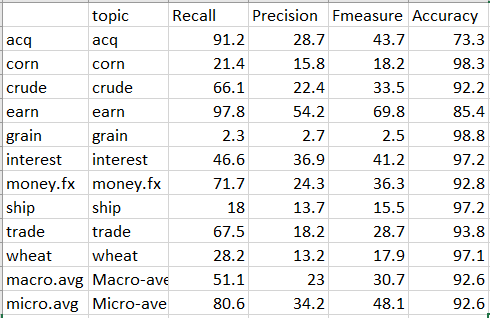
**KFold feature selection:**

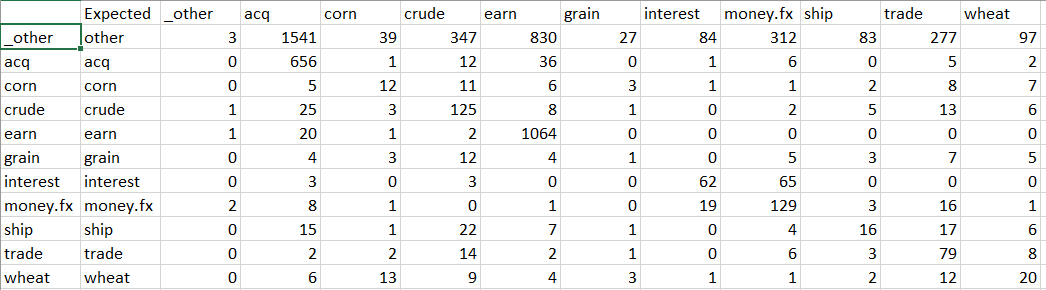
**Kfold Classifier Selection:**



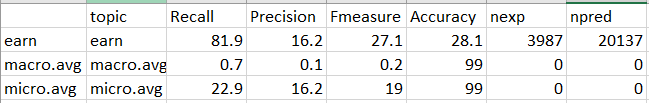
Looking at the performance of the classifiers it is clearly visible that the performance of SVM is best among all the classifiers available over here.

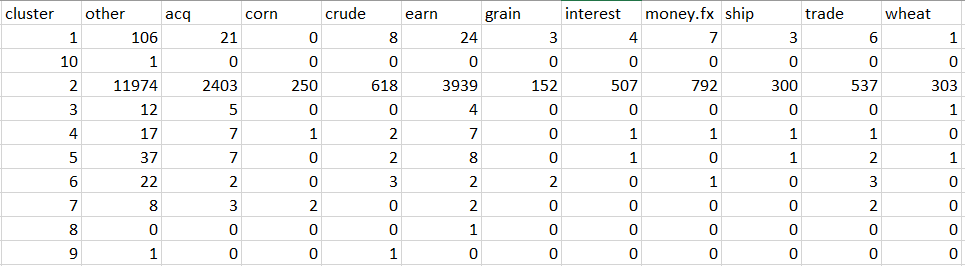
**Classifier Performance:**



**Confusion Matrix: Training the classifier over entire training set and predicting the test set**

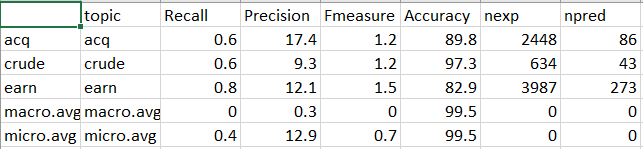
**Partitioning Around K-Medoids Performance:**



**Hierarchical Clustering Confusion Matrix (for hc\_10):**

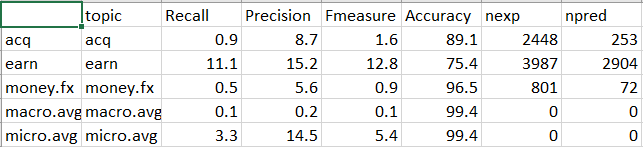
The hierarchical clustering confusion matrix for HC24 (file: hclust-24-cm.csv) and HC150 (file: hc150-performance.csv) can be found in the files attached with the code. They are not being shown here because of their huge size.

**Hierarchical Clustering 10(hc\_10) performance:**



The hierarchical clustering performance matrix for HC24(hc24-performance.csv) and HC150(hc150-performance.csv) can be found in the files attached with the code. They are not being shown here because of their huge size.

**DBSCAN Clustering Confusion Matrix:**It can be found attached with the files under the name dbclust-cm.csv. I am unable to attach it here because of its huge size.

**DBSCAN Clustering Performance Matrix:**

**PAM (Partitioning Around K-Medoids) silhouette figure:**The pdf file for thiscan be found attached with the files under the name: pam10-silhouette.pdf

**Hierarchical clustering dendogram:**

The plot for the hierarchical clustering dendogram can be found under the name hclust.pdf

*Plots for silhouette measures for hierarchical clustering can be found in the files:*

hclust10-silhouette.pdf, hclust24-silhouette.pdf, hclust150-silhouette.pdf

*plot per-class silhouette measure without maximum class can be found in the files:*

hclust10-plot.pdf, hclust24-plot.pdf, hclust150-plot.pdf

**DBSCAN Silhoutte figure:**Its available under the name dbclust-silhouette.pdf