# Practical assignment week 7

*Data Science course*

This is **not** a hand-in assignment. Bring your answers to the next lecture. You need this exercise for the follow-up exercises. Make sure you finish it.

Goals of this exercise:

- Learn how to pre-process text data to be used in a supervised learning task
- Learn to inspect sparse feature matrices

## Preliminaries

- The data frames `n_fb_post` and `c_fb_post`.
- Your experience with feature matrices, supervised learning, and evaluation from week 6.

We are going to follow-up on the classification experiments with the Facebook data (news and clickbait). The research questions that will address in the following weeks are:

- To what extent is it possible to automatically distinguish a clickbait post from a mainstream news post on Facebook?
- What features are the most important in making this distinction?

The classification results with the four features that we used in week 6 (post_type, num_reaction_cleaned, num_comment_cleaned, num_share_cleaned) were not very good. This week we will experiment with the information in the text column `status_message_without_tags`.

## Tasks

This week, we will set up the classification task using the column `status_message_without_tags`. Make a copy of your script from week 6 and adapt the code for this week's exercise.

We are going to create a sparse feature matrix in which words are features. This requires first to get all texts from the column `status_message_without_tags` in an array.

1. Initialize the array `texts`

2. Adapt the loop that walks through both dataframes. For each row in the data, store the content of the column `status_message_without_tags` in the `texts` array. Keep the code that stores the labels ‘n’ and ‘c’ in the label array `y`.

3. Use the function CountVectorizer in sklearn to initiate an object word_vectorizer that transforms the texts to feature vectors (look up what the function min_df=4 means):

```
from sklearn.feature_extraction.text import CountVectorizer
word_vectorizer = CountVectorizer(analyzer='word', min_df=4)
```

4. Use the function `fit_transform` from the `word_vectorizer` to transform the `texts` array into a feature matrix `X`
   Check the dimensionality of your data using `print(X.shape)`
   View the feature set using `print(word_vectorizer.vocabulary_)`

5. Set up your classification experiment by splitting the data (like in week 6)
   Initiate and fit `LinearSVC`, and predict values for the test set (like in week 6)
   Run the evaluation function `classification_report` to evaluate your classifier (like in week 6)

6. How does the result compare to the result that you got with the 4 simple features?