For this project, we need only two columns — “Label” and “Text”.

* Input: Text

Example: “ Vinder Real Madrid Champions League for fjerde år i træk?”

* Output: label

Example: -1( Some sentiment)

We will remove missing values in “text” column. We also create a couple of dictionaries for future use.

# Imbalanced Classes

We see that the number of text per label is imbalanced. Text are more biased towards label **“0”** and **“-1”**.

When we encounter such problems, we are bound to have difficulties solving them with standard algorithms. Conventional algorithms are often biased towards the majority class, not taking the data distribution into consideration. In the worst case, minority classes are treated as outliers and ignored. For some cases, such as fraud detection or cancer prediction, we would need to carefully configure our model or artificially balance the dataset, for example by [undersampling or oversampling](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis) each class.

However, in our case of learning imbalanced data, the majority classes might be of our great interest. It is desirable to have a classifier that gives high prediction accuracy over the majority class, while maintaining reasonable accuracy for the minority classes. Therefore, we will leave it as it is.

# Text Representation

The classifiers and learning algorithms can not directly process the text documents in their original form, as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Therefore, during the preprocessing step, the texts are converted to a more manageable representation.

One common approach for extracting features from text is to use the bag of words model: a model where for each document, a text in our case, the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored.

Specifically, for each term in our dataset, we will calculate a measure called Term Frequency, Inverse Document Frequency, abbreviated to tf-idf. We will use sklearn.feature\_extraction.text.TfidfVectorizer to calculate a tf-idf vector for each of text:

* sublinear\_df is set to True to use a logarithmic form for frequency.
* min\_df is the minimum numbers of documents a word must be present in to be kept.
* norm is set to l2, to ensure all our feature vectors have a euclidian norm of 1.
* ngram\_range is set to (1, 2) to indicate that we want to consider both unigrams and bigrams.
* stop\_words is set to "english" to remove all common pronouns ("a", "the", ...) to reduce the number of noisy features.

We can use sklearn.feature\_selection.chi2 to find the terms that are the most correlated with each of the label:

**# '-2'**:

. Most correlated unigrams:

. har

. vil

. Most correlated bigrams:

. boris johnson

. donald trumps

**# '-1'**:

. Most correlated unigrams:

. blandt

. rige

. Most correlated bigrams:

. pr sident

. boris johnson

**# '0'**:

. Most correlated unigrams:

. sig

. eller

. Most correlated bigrams:

. pr sident

. donald trumps

**# '1'**:

. Most correlated unigrams:

. har

. vil

. Most correlated bigrams:

. boris johnson

. donald trumps

**# '2'**:

. Most correlated unigrams:

. mange

. sident

. Most correlated bigrams:

. boris johnson

. pr sident

They all make sense, don’t you think so?

## 

## Multi-Class Classifier: Features and Design

* To train supervised classifiers, we first transformed the “Text” into a vector of numbers. We explored vector representations such as TF-IDF weighted vectors.
* After having this vector representations of the text we can train supervised classifiers to train unseen “Text” and predict the “label” on which they fall.

After all the above data transformation, now that we have all the features and labels, it is time to train the classifiers. There are a number of algorithms we can use for this type of problem.

# Model Selection

We are now ready to experiment with different machine learning models, evaluate their accuracy and find the source of any potential issues.

We will benchmark the following four models:

* Logistic Regression
* (Multinomial) Naive Bayes
* Linear Support Vector Machine
* Random Forest

The vast majority of the predictions end up on the diagonal (predicted label = actual label), where we want them to be. However, there are a number of misclassifications, and it might be interesting to see what those are caused by.

As you can see, some of the misclassified texts are texts that touch on more than one subjects (for example, textsinvolving both -1 and 1). This sort of errors will always happen.

Again, we use the [chi-squared test](https://en.wikipedia.org/wiki/Pearson%27s_chi-squared_test) to find the terms that are the most correlated with each of the categories:

**# '-2'**:

. Top unigrams:

. vil

. rige

. Top bigrams:

. boris johnson

. donald trump

**# '-1'**:

. Top unigrams:

. jeg

. sig

. Top bigrams:

. det er

. pr sident

**# '0'**:

. Top unigrams:

. blev

. trumps

. Top bigrams:

. donald trumps

. donald trump

**# '1':**

. Top unigrams:

. trump

. rt

. Top bigrams:

. boris johnson

. pr sident

**# '2':**

. Top unigrams:

. eller

. blive

. Top bigrams:

. repr sentanternes

. sentanternes hus

They are consistent within our expectations.

Finally, we print out the classification report for each class.

Conclusion:

I believe the dataset given should have more data points. The text has to be cleaned properly with inclusion of stop word removal for specific danish language. The model built can be improved by passing some hyper parameters to the LinearSVC and Logistic regression. The data needs to be balanced as well.