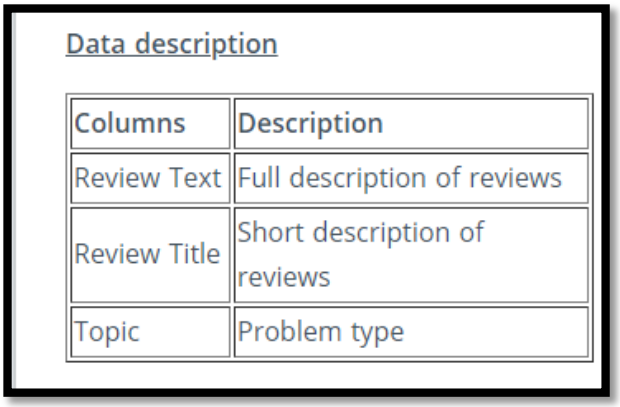
Identify Key Aspects of Review

Data Description

Train set: 5960 x 3

Test set: 2553 x 2



Data description

Columns	Description
Review Text	Full description of reviews
Review Title	Short description of reviews
Topic	Problem type

Platform Used:

Coding R 3.5.0

Platforms: Core i7, 16Gb Ram

Key Libraries

- library(**tidyverse**) # general utility & workflow functions
- library(**tidytext**) # tidy implementation of NLP methods
- library(**topicmodels**) # for LDA topic modelling
- library(**tm**) # general text mining functions, making document term matrixes
- library(**SnowballC**) # for stemming
- library(**janitor**) # clean column names
- library(**h2o**) # Predictive Modelling

Exploratory Data Analysis

Check if **Review.Title** in Train and Test are same or not if not Collate: Merge Title and Text to avoid feature miss.

Topic Modelling

Supervised topic modelling with TF-IDF

The general idea behind how TF-IDF works is this:

- Words that are very common in a specific document are probably important to the topic of that document
- Words that are very common in all documents probably aren't important to the topics of any of them

So a term will receive a high weight if it's common in a specific document and also uncommon across all documents. Following is the TF-IDF for train set.

Key Inferences from TF-IDF Plot:

- Topic **allergic** has keywords associated like stomach, diarrhea and reaction.
- Topic **Bad Taste** has keywords associated like Fishy, disgusting and protein.
- Topic **Pricing** has keywords associated like overpriced and expensive.

Feature Engineering

- Merge Train and test set to create a Corpus
- Key NLP steps
  1. Removing Punctuation
  2. Removing numbers
  3. Converting to lower case
  4. Removing stop words
  5. Stemming
  6. Removing Whitespace
  7. Create Document Term Matrix
  8. **Remove Sparse Terms**

Removing Sparse Terms

Sparsity refers to the threshold of relative document frequency for a term, above which the term will be removed. Relative document frequency here means a proportion. As the help page for the command states (although not very clearly), sparsity is smaller as it approaches 1.0. (Note that sparsity cannot take values of 0 or 1.0, only values in between.)

For example, if you set sparse = 0.99 as the argument to removeSparseTerms(), then this will remove only terms that are more sparse than 0.99. The exact interpretation for sparse = 0.99 is that for term j, you will retain all terms for which $df_j > N * (1 - 0.99)$, where N is the number of documents -- in this case probably all terms will be retained (see example below).

Near the other extreme, if sparse = .01, then only terms that appear in (nearly) every document will be retained. (Of course this depends on the number of terms and the number of documents, and in natural language, common words like "the" are likely to occur in every document and hence never be "sparse".)

In our dataset, I had set Sparsity to 0.95 so that 80 keywords as variables were retained.

Predictive Modelling

Used **H2o_Automl Framework**

Parameter's for Modelling

1. Exclusion of Random Forest and GLM Models, so only deep learning and GBM will be considered.
2. 10 fold Cross Validation
3. Balanced Classes for Modelling (Sampling)
4. stopping_metric = lift top group
5. Runtime 2hours

Results:

StackedEnsemble_AllModels gave better result as compared to **deep learning,** also removal of Sparse terms increases model accuracy and decreases runtime complexity.