Predicting review helpfulness : the case of class imbalance

Auteur : Delic, Leïla
Promoteur(s) : Ittoo, Ashwin
Faculté : HEC-Ecole de gestion de l'Université de Liège
Diplôme : Master en ingénieur de gestion, à finalité spécialisée en Supply Chain Management and Business Analytics
Année académique : 2018-2019
URI/URL : http://hdl.handle.net/2268.2/6399

HEC LIÈGE
Management School - Liège Université

PREDICTING REVIEW HELPFULNESS: THE CASE OF CLASS IMBALANCE

Jury :
Promoter :
Ashwin ITTOO
Readers :
Cédric HEUCHENNE
Anne-Sophie HOFFAIT

Dissertation by
Leila DELIC
For a Master's degree in Business Engineering with a specialization in Supply Chain Management and Business Analytics
Academic year 2018/2019

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my promotor, Professor Ashwin Ittoo, for giving me the opportunity and means to work on this topic, for his availability, immense knowledge, and his precious guidance.

I would also like to thank the other members of my jury, Professor Cédric Heuchenne and Anne-Sophie Hoffait, for taking the time to read this paper and for their interest in the topic.

1 INTRODUCTION

Online consumer reviews have attracted a lot of attention in research.

Online reviews influence both the customers and online retailers (Siering & Muntermann, 2013; Park & Nicolau, 2015).

To customers, they constitute a source of information in their purchase decision. In this regard, a 2010 study made by the e-tailing group (eMarketer, 2010) revealed that 92% of US internet users read product reviews. Furthermore, still according to the study, 46% of these said to be influenced to purchase by the review, while 43% claimed that it deterred them from doing so. Only 3% stated that reviews did not affect their purchasing decision (eMarketer, 2010).

Online reviews also benefit retailers by increasing the number of consumers who consult the product (Siering & Muntermann, 2013). The conclusions regarding the impact of reviews on sales seem to be contradictory: while some authors (Chevalier & Mayzlin, 2006, as cited in Park & Nikolau, 2015) claim that positive reviews positively influence sales and negative ones influence them negatively, others (Chen, Wu, & Yoon, 2004, as cited in Park & Nikolau, 2015) consider that they are not correlated.

The rising challenge with online reviews is that they are becoming increasingly abundant, which can become overwhelming for customers facing then an overload of information (Charrada, 2016).

In order to tackle this, Amazon has implemented a voting system in which community members can vote on whether they deem a given review to have been helpful or unhelpful to them. Reviews then appear with a peer rating of the form "[*number of helpful votes*] out of [*number of members who voted*] found the following review helpful." (Ghose & Ipeirotis, 2011).

Based on these votes, the most helpful reviews are determined and displayed first. While this might help customers navigate through reviews, it is not always an appropriate method to determine a review's quality (Liu, Cao, Lin, Huang, & Zhou, 2007). In fact, Liu et al. (2007) have identified three kinds of biases that the user voting mechanism suffers from: *imbalance vote bias*, *winner circle bias*, and *early bird bias*.

The *imbalance vote bias* occurs as reviews are more likely to be rated as helpful than unhelpful. In a sample of 23,141 reviews, the authors have observed that half of them received more than 90% helpful votes.

Furthermore, top ranking reviews have a higher number of votes than lower-ranked ones. This has as consequence that they are the ones which get the most attention from users, who are then more likely to add their own vote to it. This makes it thus harder for reviews with only a few votes to gain visibility. This is what the authors describe as the *winner circle bias*.

Lastly, the *early bird bias* is due to the fact that most recent reviews tend to get less votes. The earlier a review is posted, the longer it is exposed to the users and thus, the higher number of votes it gets.

To overcome these, research in machine learning has been aimed at finding an automatic yet efficient way of assessing review helpfulness, which would not rely on human votes. A successful prediction system could be just as operational as recommendation systems (Diaz & Ng, 2018).

The aim of this research is dual: first, to analyze different machine learning algorithms to classify text data and report the best performing one, and, second, to investigate the pre-processing techniques which can be applied to a dataset to increase algorithm performance, especially in the context of imbalanced data.

This paper will be organized as follows: in the second chapter, some key theoretical aspects related to the task of helpfulness prediction will be presented. More precisely, the concepts of text classification, vector space representation, dimensionality reduction techniques, classification algorithms, K-fold cross validation, evaluation metrics and class imbalance will be redefined. The last part of the chapter will describe other approaches to the problem of review helpfulness prediction.
The third chapter will be dedicated to the methodology applied in the implementation of the various developed algorithms. It will start with a description of the data collection and preparation steps. It will then follow with the results of a first attempt at classification. Then, class imbalance and feature correlation will be studied through some data exploration. After that, four dimensionality reduction techniques will be presented and applied. Their implementation will constitute the second attempt at classification, the results of which will be

reported. The final part of this third chapter will be dedicated to the presentation and implementation of three data balancing techniques. The results of the third attempt at classification that follows from these will be presented.

The fourth and penultimate chapter will focus on the description of the production of the Master thesis according to the fundamental principles of project management.

Finally, the last chapter will present the conclusions of this research, as well as its limitations and areas of improvement.

2 LITERATURE REVIEW

2.1 Text Classification

Text classification is defined as the process of classifying documents into a fixed set of predefined classes. More formally, if d_i is a document of the entire set of documents D and, in the case of binary classification, C = $\{c_1, c_2\}$ is the set of classes, then text classification is the process of assigning one class c_j to each document d_i (Ikonomakis, Kotsiantis, & Tampakas, 2005).

In the context of machine learning, text classification is done automatically. Prior to that, however, classifiers must be learned. In supervised learning, this is done from hand-labelled examples, referred to as training set, by learning associations between documents and their corresponding class. Once the training has been done and the classifiers have been learned, these can make predictions on unseen documents.

2.1.1 Vector Space Model

The particularity of text classification is that it requires the transformation of the documents, which are unstructured texts, into a structured format that can be processed by machine learning algorithms.

To do so, documents are usually first tokenized, that is to say, broken down into words. This allows for the creation of a set of unique terms belonging to the document set, called vocabulary. Each document is then represented by a vector, i.e. $d = (w_1, \dots, w_{|V|})$ where |V| is the size of the vocabulary (Lan, Tan, Su, & Lu, 2009). Each w_k stands thus for a term of the vocabulary. Since most algorithms function on numerical input, each component w_k of a vector corresponds to the weight of the term in the document the vector represents.

Several weighing techniques exist. The most popular are binary weighting, term-frequency and term frequency-inverse document frequency (TF-IDF).

In the first case, a weight of 1 is attributed to terms which are present in the document and a weight of 0 is attributed to those which are not.

In the second case, the term-frequency is used as feature. The term-frequency is defined as the ratio of the number of times a term appears in a document to the total number of words in the same document. It measures the importance of the term in the document.

In the third and last case, the weight is computed using the TF-IDF score. The TF-IDF score is the product of the term-frequency (tf) defined above and the inverse document frequency (idf).

The inverse document frequency is measured as the total number of documents in the document set divided by the number of documents containing the term. The rarer the term, the highest its inverse document frequency. Incorporating it into the TF-IDF score allows thus to attenuate the effect of terms which appear frequently in many documents. Indeed, these are considered as less discriminating (Sebastiani, 2002). '*One'* and '*the'* make up examples of such terms which are potentially relatively common in the corpus but do not add any information.

Mathematically, the TF-IDF score of term *i* in document *j*, $TF - IDF_{ij}$, is computed as follows:

$$TF - IDF_{ij} = \underbrace{\frac{n_{ij}}{\sum_i nij}}_{tf_{ij}} \times \underbrace{\log\left(\frac{N}{df_i}\right)}_{idf_{ij}}$$

where index *i* represents a term

index *j* represents a document

n_{ij} is the number of times term *i* appears in document *j*

N is the total number of documents

df_i is the number of documents containing the term *i* (document frequency)

The tf-idf score is always greater or equal to zero, the ratio $\frac{N}{df_i}$ always being greater or equal to 1 and log(1) being equal to 0. It is the lowest for terms which appear in all documents. One can indeed see that the ratio $\frac{N}{df_i}$ is higher for terms which appear in more documents, which

translates into an idf score, and thus a tf-idf score, closer to 0. Conversely, the rarer the term, the higher the tf-idf score. The tf-idf is highest when a term appears with a high frequency in a small number of documents.

This last technique is the one that will be applied in this research as it is considered as a standard technique (Sebastiani, 2002).

2.1.2 Dimensionality Reduction

Technically, once the documents have been vectorized, machine learning algorithms can be applied to them. However, some authors recommend going a step further.

In text classification, the vector space is vast, or, equivalently, data is high-dimensional (|V| is large). This presents a challenge as machine learning models tend to overfit, which then results in poorer performances (Alelyani, Tang, & Liu, 2013). Overfitting occurs when a classifier is tuned to the characteristics of the training dataset, rather than to those of the classes (Ikonomakis et al., 2008).

To mitigate this problem, various techniques have been studied. They offer the advantage of reducing the size of the dataset, which translates into a more efficient learning process as the computational requirements decrease. They also reduce the risk of overfitting and lead to a better performance of the classification algorithms.

Ikonomakis et al. (2008) use feature selection. They formally define feature selection as the "reduction of the dimensionality of the dataset by removing features that are considered irrelevant for the classification". In the document representation presented above, features are the terms encoded using TF-IDF.

In order to apply feature selection, one must first determine the features to discard using an evaluation function (Ikonomakis et al., 2008). This function attributes a score to each feature so that the top scoring ones are selected. Amongst the feature-scoring methods is TF-IDF. This is the one that will be used in this research as it follows logically from the chosen document representation.

Sebastiani (2002) promotes a linguistic approach: stemming. Stemming is a word normalization technique, which converts words to their root form. Often, lemmatization, which essentially pursues the same objective, is compared to stemming. Some authors (Toman, Tesar, & Jezek, 2006) argue however that neither stemming nor lemmatization increases performance. Both these techniques will be presented in a detailed manner and applied in the third chapter.

Another linguistic approach recommended by the literature consists in replacing words in the documents by their corresponding part-of-speech (POS) in other to make the features more general. More sophisticated variants have been developed. For instance, Tisserant, Prince and Roche (2013) have created a method based on POS aimed at replacing the least relevant words in the classification task by their POS.

2.2 Text Classification Algorithms

One of the greatest challenges in automatic helpfulness prediction, as stated by Diaz and Ng (2018), is that while there is a wealth of literature on the topic, comparing studies is highly complex. The two main reasons that the authors put forward are the use of different datasets and the lack of concertation. According to Diaz and Ng (2018), it is difficult to determine the state-of-the-art as researchers have not necessarily used each other's successes as building blocks.

Still, the support vector machine (SVM), multinomial naive Bayes (MNB) and random forest (RF) are widely used in the context of helpfulness prediction. They will be thoroughly presented in the following section.

SVMs are considered to be prominent classification methods due to their high accuracy, potential to handle high-dimensional data (Ben-Hur & Weston, 2010) and their robustness (Joachims, 1998). The MNB model is often used because of its simplicity and effectiveness (Ikonomakis et al., 2008). As to random forests, they are said to be robust, and Ghose and Ipeirotis (2011) argue that they perform better than SVMs for the problem of review helpfulness prediction. They further claim that their training time is significantly shorter than that of SVMs. Overall, random forests are performant on high-dimensional data, have generally a high accuracy and run fast on large datasets (Ali, Khan, Ahmad, & Maqsood, 2012).

Even though these are generally reported to perform well, the use of different datasets amongst researchers makes it difficult to determine a baseline performance.

2.2.1 Support Vector Machine

Kowalczyk (2017) defines the goal of support vector machines as "to find the optimal separating hyperplane which maximizes the margin of the training data".

To simplify matters, one can first consider the case of binary classification of two-dimensional instances. In that case, each instance can be scattered on a plot according to the two dimensions. It is then possible to divide the data points into two groups, each of these containing data points which correspond to a same class. Actually, there exist several ways (or lines) to divide the data points.

The optimal separation is the one which is at greatest distance between its closest data point from either class. Considering that this distance multiplied by two is the margin, the problem is equivalent to choosing the line that has the largest margin.

The line then chosen is said to be optimal because it correctly classifies the training data and reduces the chances of misclassification of unknown instances.

This problem can be scaled up to high-dimensional data. In that case, the separation is no longer a line but a hyperplane.

SVM is in practice however not suited for large datasets (beyond ten of thousands of samples, according to Pedregosta et al. (2011)). Indeed, the time complexity of the algorithm is quadratic. Pedregosta et al. (2011) therefore suggest using the Stochastic Gradient Descent Classifier instead.

2.2.2 Stochastic Gradient Descent Classifier

The stochastic gradient descent classifier that will be used in this context will be a linear SVM that uses stochastic gradient descent for learning.

As previously mentioned, the goal of SVMs is to maximize the classification margin. In fact, this is done by minimizing a loss function called hinge loss function. Mathematically, it is defined as:

$$l(y) = \max[0, 1 - (t \times y)]$$

where *l(y)* is the loss function

t is an indicator that takes value +1 when the output is correctly predicted and – 1 when it is wrongly predicted.

y is the value of the classification function. In the case of linear SVM, this function is linear with the input vector *x* (document): $y = w^T x + b$

It is equal to 0 when *t* and *y* have the same sign and $|y| \geq 1$, and it grows linearly with *y* when either *t* and *y* have different signs or when they have the same sign but $|y| < 1$. In the latter case, the model penalizes correct predictions which are not distant enough from the hyperplane.

The idea of gradient descent is to find the parameters of the hinge loss function which minimize it. This is equivalent to finding the parameters w^T and b which minimize the loss function.

To do so, it starts with an initial guess on the parameters. Then, it iteratively adjusts the values so as to minimize the function. It uses only a single training sample at each iteration. This sample is chosen randomly in the case of a stochastic gradient descent.

The process stops when convergence has been reached; when the gradient descent cannot further decrease the loss function.

2.2.3 Multinomial Naive Bayes

The multinomial naive Bayes classifier assigns a document *d* to the class *c** that maximizes the probability of *d* belonging to *c*, written as P(*c*|*d*).

$$c^* = \arg\max_{c \in C} \{P(c|d)\} \tag{1}$$

This probability, called posterior probability, is computed using Bayes' rule as:

$$P(c|d) = \frac{P(d|c) \times P(c)}{P(d)} \tag{2}$$

where P(*c*) is the prior probability of *c*

P(*d*) is the probability of the evidence *d*

P(*d*|*c*) is the conditional probability of *d* given *c*

The prior in Equation 2 can be estimated by:

$$P(c) = \frac{\text{number of documents that belong to the class c}}{\text{total number of documents}} \tag{3}$$

By applying marginalization and the chain rule, one can re-write the probability of the evidence *d* in Equation 2 as:

$$P(d) = \sum_{c} P(d, c) = \sum_{c} P(c)\, P(d|c) \tag{4}$$

Furthermore, a document can be seen as a vector of K features (words): $d = (w_1, w_2, \dots, w_K)$ (Peng, Schuurmans, & Wang, 2003) where K is the size of the vocabulary. To simplify the computation, the model assumes that the features are conditionally independent given class *c*. This assumption is considered to be a strong, "naive" one. Models based on Bayes' Theorem which assume this independence are therefore called naive Bayes. Given this hypothesis and assuming the features follow a multinomial distribution, P(*d*|*c*) can be re-formulated as:

$$P(d|c) = \left(\sum_{i=1}^{K} f_{id}\right)! \prod_{i=1}^{K} \frac{P(w_i|c)^{f_{id}}}{f_{id}!} \quad (5)$$

where f_{id} is the number of times the feature *i* occurs in document *d*.

This multinomial model has been shown the outperform the multi-variate Bernoulli model, also based on the naive Bayes assumption (McCallum & Nigam, 1998).

It can be noted that ignoring the terms $(\sum_{i=1}^{K} f_{id})!$ and $\prod_{i=1}^{K} f_{id}!$ does not affect the result as they are independent from *c* (Kirbiya, Frank, Pfahringer, & Holmes, 2004).

All in all, Equation 1 can be re-expressed as:

$$\mathrm{c}^* = \arg\max_{c \in C} \left\{ \frac{\mathrm{P(c)} \times \prod_{i=1}^{K} P(w_i|c)^{f_{id}}}{\mathrm{P(d)}} \right\} \quad (5)$$

$$= \arg\max_{c \in C} \left\{ \mathrm{P(c)} \times \prod_{i=1}^{K} P(w_i|c)^{f_{id}} \right\} \quad (6)$$

Since P(d) is a constant, it can be ignored.

Lastly, the probability of a feature w_i given class *c*, $P(w_i|c)$, is estimated according to the maximum likelihood method by:

$$P(w_i|c) = \frac{F_{ic}}{\sum_{i=1}^{K} F_{ic}} \quad (7)$$

where F_{ic} is the number of times feature *i* occurs in all the documents that belong to class *c*.

This estimation method presents an issue: words *i* which did not appear in any documents of the training set but do appear in the test set will set the probability $\mathrm{P}(w_i|c)$ and therefore P(*c*|*d*) to zero. To overcome this issue, Laplace smoothing is introduced. It consists in the addition of 1 to the numerator of the expression of $\mathrm{P}(w_i|c)$. In the denominator, K is added so as to ensure that the sum of probabilities equals to 1.

Using Laplace smoothing, the expression of Equation 7 becomes thus:

$$P(w_i|c) = \frac{1 + F_{ic}}{K + \sum_{i=1}^{K} F_{ic}} \quad (8)$$

2.2.4 Random Forest

The random forest grows a multitude of decision trees and uses the prediction of each to determine the class to which a document belongs. The advantage of combining trees is that it helps alleviate the overfitting that decision trees have known to present.

There exist different approaches to random forests. The one that will be used here will be based on bagging (Breiman, 1996).

A large number K of trees are grown (here, 200 trees will be constructed). The training set used in each of these trees is determined according to the bootstrap aggregating, or bagging, method: samples are selected from the original dataset at random and with replacement. The size of the sample set is the same as the original input size.

The randomness is represented by a random vector: for each of the *k* trees, a random vector θ_k is generated. All θ_k's are i.i.d. as they are independent of the $\theta_1 \ldots \theta_{k-1}$ random vectors and are identically distributed. These determine thus how the sample are drawn from the original dataset.

Once all the trees are constructed, they can be used for classification. Each tree is therefore a classifier that depends both on this random vector and on an input vector *x* (in the case of text classification, the input vector is a document). It is represented by Breiman (2001) as $h(x, \theta_k)$.

To determine the class to which the input vector *x* belongs, the random forest chooses the most popular class amongst the prediction of each individual tree. The notation suggested by Koulis (2003) offers a clearer view on this:

Let P = $\sum_{k=1}^{K} I(\mathrm{h}(x, \theta_k) = 1)$ and let Q = $\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{I}(\mathrm{h}(\mathrm{x}, \theta_{\mathrm{k}}) = 0)$, with I() being the indicator function:

If P $>$ Q, then the forest classifies *x* as 1.

If Q $<$ P, then the forest classifies *x* as 0.

Culter, Cutler and Stevens (2012) further define a prediction function:

$$f(x) = arg\ max_{y \in Y} \sum_{k=1}^{K} I(y = h_k(x))$$

where f(x) is the prediction function, which takes as input the vector *x*

Y is the set of possible values of the target

K is the total number of trees grown

$h_k(x)$ = $\mathrm{h}(x, \theta_k)$

This can all be summarized into the definition of a random forest given by Breiman (2001), who defines it as "a classifier consisting of a collection of tree-structured classifiers $\{ h(x, \theta_k), k = 1, \ldots \}$ where $\{\theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input *x*."

2.3 K-fold Cross Validation

The performance of the classifiers will be tested against several validation sets: this method is called K-fold Cross Validation. It is usually more reliable than the traditional approach of splitting the dataset into a single training and a single test sets as the results can vary a lot depending on the test set.

K-fold Cross Validation works in the following way: the dataset is randomly divided into K folds of equal size. Usually, K is chosen to be 10 (Ghose & Ipeirotis, 2011). There are K iterations. In the k^{th} iteration, the k^{th} fold is used to validate the model, while the remaining ones are used to train the model. In total, each fold is thus used once and exactly once for testing. At

each iteration, some performance measures are computed. The reported result for each performance indicator is its average over the K iterations.

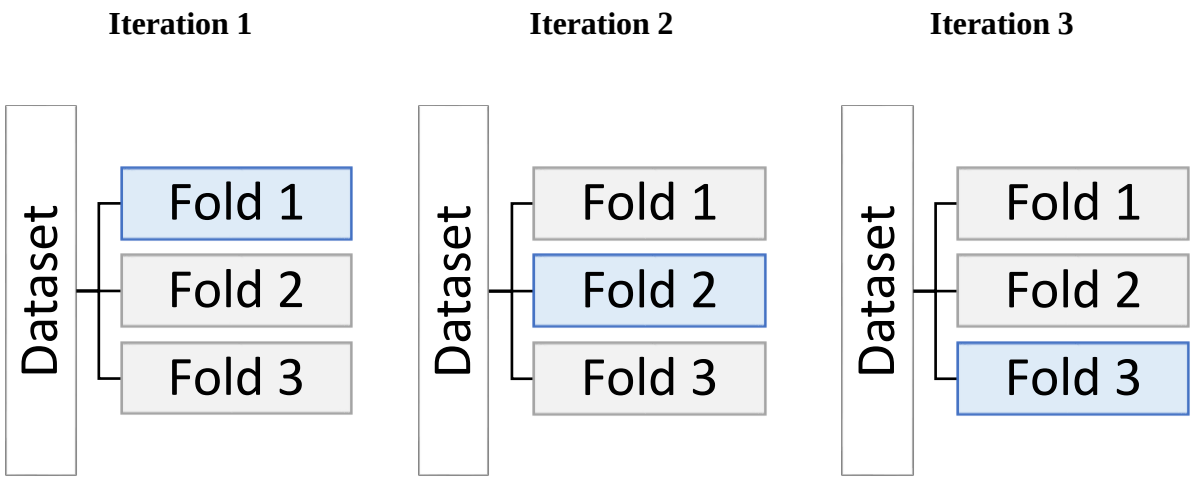

Figure 1. Illustration of a K-Fold Cross Validation with K = 3. The blue boxes represent folds used for validation, while the grey ones represent folds used for training.

2.4 Evaluation Metrics

To evaluate the performance of the classifiers, several metrics will be used. The first is the confusion matrix, which is a table that allows to visualize the performance of a classifier. Namely, it identifies the number of *true positives*, *false negatives*, *false positives* and *true negatives*.

True positives (TP) represent observations that are correctly predicted as positive. *False negatives* (FN) are observations which are in fact positive but predicted as negative. Conversely, observations which are in fact negative but predicted as positive are called *false positives* (FP). Finally, *true negatives* (TN) are observations that are correctly predicted as negative.

The positive class will be considered to be that of helpful reviews and the negative class to be that of not helpful reviews. The classification matrix can therefore be represented as displayed in Figure 2.

		Actual	
		Not helpful	Helpful
Predicted	Not helpful	TN	FN
	Helpful	FP	TP

Figure 2. Confusion matrix

The accuracy can be obtained from the confusion matrix in a straightforward manner according to the formula:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

It corresponds thus to the ratio of correctly predicted observations to the total number of observations.

However, the accuracy is not the most reliable metric. As stated by Hossin and Sulaiman (2015), this metric can be biased towards the majority class. Indeed, if one of the classes is underrepresented, a naive/degenerated classifier which only predicts the majority class would have a high accuracy. This phenomenon amplifies with the skewness or scarcity of the data; the smaller the minority class, the higher the accuracy of the naive classifier.

Furthermore, as put forward by these authors, accuracy does not always lead to choosing the best solution. In some cases, a naive classifier can have a higher accuracy than a classifier which would correctly predict all the examples of the minority class, at the cost of wrongly predicting some of the majority class. Reasoning according to this metric would thus result in choosing the first classifier. However, in most applications, it would be more desirable to correctly predict all the rare cases. A famous example of this is the case of disease detection.

For the reasons expressed above, other metrics will be used to evaluate the performance of the models, namely precision, recall and F1-score.

The precision (P) corresponds to the proportion of correctly predicted positive observations over the total of predicted positive observations

$$P = \frac{TP}{TP + FP}$$

The recall (R) is defined as the proportion of correctly predicted positive observations over the total of all observations which are actually positive.

$$R = \frac{TP}{TP + FN}$$

The F1-score is the harmonic mean of the precision and the recall. Actually, it is a special case of the F-score, which can be interpreted as the weighted harmonic mean of the precision and recall. In the case of F1-score, the weights are both set to ½.

$$F - score = \left(\frac{\propto P^{-1} + (1-\propto)R^{-1}}{2}\right)^{-1}$$

$$= \frac{2}{\propto \frac{1}{P} + (1-\propto)\frac{1}{R}} = \frac{2}{\frac{\propto R}{PR} + \frac{(1-\propto)P}{PR}}$$

$$= 2 \times \frac{PR}{\propto R + (1-\propto)P} = 2 \times \frac{{}^{1}/_{\propto} PR}{\frac{\propto}{\propto} R + \frac{(1-\propto)}{\propto} P}$$

$$= 2 \times \frac{(1+\beta^2)PR}{R + \beta^2 P}$$

$$where\ \beta^2 = \frac{1-\propto}{\propto}\ and\ \propto \in [0;1]$$

$$Thus, with\ \propto = \frac{1}{2}$$

$$F1 - score = 2\ \times \frac{PR}{P + R}$$

All three metrics are defined over the [0,1] interval. The higher the metrics, the better the classifier.

Sokolova and Lapalme (2009) claim that the three metrics presented above reflect the correct identification of positive instances and rather neglect that of negative ones. This was one of the issues previously raised in the case of accuracy. In order to avoid bias, the precision, recall and F1-score will be computed from the point of view of both classes. The recall of the positive class is known as sensitivity and that of the negative class as specificity. In addition to these, a support-weighted average will be computed for each metric. Support is defined as the number of observations belonging to one class.

2.5 Class Imbalance Problem

Another challenge in helpfulness prediction, and more generally in text classification, is that one often has to deal with the class imbalance problem. Dealing with unbalanced data has been identified as one of "the top 10 challenging problems in data mining research" (Yang & Wu, 2005). In fact, this problem has been shown to be present in many real-world applications such as bankruptcy prediction (Veganzones & Séverin, 2018), medical diagnosis (Rastgoo et al., 2016), customer churn prediction (Burez & Van den Poel, 2009) and software defect prediction (Yang, Tang, & Yao, 2013).

The imbalance class problem is defined by Prati, Batista and Monard (2009) as a problem which occurs "whenever one class represents a circumscribed concept, while the other represents the counterpart of that concept, so that examples from the counterpart class heavily outnumber examples from the positive concept class". In other words, this problem happens when one class is represented by a greater number of examples than the other one. Usually the first class is then referred to as the majority class, and the second one as the minority class. In that case, the dataset is said to be imbalanced.

This class imbalance is caused either by the nature of the event at hand or by the lack of availability of the data. For instance, in the problem of melanoma detection, there are usually more benign cases than malignant ones (Rastgoo et al., 2016). The imbalance in that case is due to the nature of the phenomenon. Lack of availability of data happens when its collection is limited because of either privacy or economic reasons (Prati et al., 2016).

As pointed out by Liu et al. (2007), Amazon reviews suffer from such imbalance with helpful reviews being more abundant than unhelpful ones.

Imbalanced datasets hinder the learning process and result in poorer performance of the standard classification algorithms (Lemaître, Nogueira, & Aridas, 2017 ; Prati et al., 2016). These tend to show bias towards the majority class by performing well on this one and poorly on the minority class (Ganganwar, 2012). Sometimes, the algorithms simply classify every new example as belonging to the majority class (Prati et al., 2016). One can thus imagine that there is only little use of such classifiers. In the case of helpfulness prediction, predicting all reviews as helpful would not allow to achieve efficient ranking, or any ranking at all as a matter of fact.

The reason for this bias is that the traditional algorithms pursue an objective of maximization of the accuracy, or, equivalently, an objective of minimization of the error rate (i.e. the percentage of incorrect classification). To do so, they tend to correctly classify all examples of the majority class, so as to have no misclassification amongst them. The cost of doing so is the misclassification of some of the examples of the minority class. However, since the overall number of examples from the minority class is already comparably small, the contribution of the misclassification of minority class examples to the overall error rate cannot be as significant as that of the majority class ones (Ganganwar, 2012).

To overcome this issue, several approaches have been suggested in the literature. However, there is still some confusion so as to which of them are the most efficient and results sometimes contradict.

These approaches can be classified into two categories: data-level approaches and algorithm-level approaches. The first ones modify the class distribution of the dataset in order to restore balance whereas the second ones consist of fine-tuning of existing algorithms in order to make them more performant to imbalanced datasets (Ali, Shamsuddin, & Ralescu, 2015). Amongst the most popular data-level approaches, Ali et al. (2015) identify sampling and feature selection. Improved algorithms, one-class learning, cost-sensitive learning and ensemble methods are categorized as algorithm-level approaches (Ali et al., 2015).

Sampling is by far the most common one. There are two types of re-sampling techniques: under-sampling and over-sampling.

Under-sampling is defined by Veganzones and Séverin (2018) as an approach which aims at creating “a balanced subset from the original data set by removing samples from the majority

class". Its main advantage is that it reduces the dataset and therefore the processing time. Its disadvantage is that it potentially leads to a loss of information (Liu, 2004).

Over-sampling aims at creating "a balanced subset from the original dataset by duplicating samples of the minority class" (Veganzones & Séverin, 2018). In contrast to under-sampling, over-sampling does not lead to a loss of information. However, it increases the size of the dataset, which in turn increases the computation time and memory need (Liu, 2004).

There is no unanimous winner amongst the two types of re-sampling: while Drummond and Holte (2003) claim that under-sampling beats over-sampling, Maloof (2003) states that they create similar classifiers.

In the context of this research, several re-sampling techniques will be investigated. These will be presented in the third chapter.

2.6 Other Approaches

More recently, researchers have been working on the identification of the most relevant features in review helpfulness. Diaz and Ng (2018) distinguish them amongst two categories: content and context features. The first ones are those directly obtained from the review. The second ones are those obtained from other sources such as reviewer or user information.

Diaz and Ng (2018) report that the majority of researchers consider review length to be positively correlated to helpfulness. According to them, another content feature which is said to be correlated to helpfulness is the readability of the review, expressed as a metric which indicates how easy to read a review is.

They further state that both the review's star rating and its extremity (whether it is positive, negative or neutral) are generally considered to be useful for the task of predicting helpfulness.

3 RESEARCH METHODOLOGY

The first section of this chapter will be dedicated to data collection. The second one, called data preparation, to the operations carried out on the collected data in order to make it exploitable for classification. The third section will focus on the straightforward implementation of the MNB, RF and SGD classifiers using the prepared data. Some data exploration in the fourth stage will allow a better understanding of the results obtained in the preceding section. With the intention of improving these results, dimensionality reduction methods will be applied to the dataset as obtained at the end of the second step. The classifiers will then be re-run with these. Lastly, three re-sampling techniques aimed at balancing the data will be developed, each creating thus three balanced versions of the dataset from the second section. The classifiers will then be tested on each of these balanced datasets and their performance reported.

3.1 Data Collection

The dataset used was extracted from Amazon Product Data (He & McAuley, 2016 ; McAuley, Targett, Shi, & van den Hengel 2015). It is actually a subset of a larger dataset, which contains product reviews and metadata from Amazon for 24 product categories. In total, it gathers 142.8 million reviews that span a large period of time (from May 1996 to June 2014). As state by Diaz and Ng (2018), the use of pre-collected datasets, such as the Amazon Product Data, allows for fairer research comparisons.

The product category selected for this work is ‘Toys and Games’. The file contains the following information for 167,597 reviews: ID of the reviewer, ID of the product, name of the reviewer, helpfulness rating of the review, text of the review, rating of the product, summary of the review, time of the review (unix), time of the review (raw). Here, only the fields deemed as most relevant were kept, that is to say: helpfulness rating of the review and text of the review.

3.2 Data Preparation

This step consists in manipulations on the raw dataset in order to make it exploitable for classification. Reviews underwent two main manipulations: the computation of a score to define their helpfulness/unhelpfulness and the cleaning of the review texts themselves.

These were carried out in R. The cleaned text and calculated score were then re-written into a text file to facilitate the following steps.

3.2.1 Computation of a Helpfulness Score

In the raw file, ratings appear under the form: [2,4]. Immediately, reviews which received a rating of the form [0,0] were discarded. These are reviews which were not rated and considering them would imply wrongly attributing them a score of 0 when they could actually have had a higher score had they been rated. The dataset was thus first subsetted so as to exclude these. As a result, its size decreased from 167,597 to 62,907 reviews. In other words, more than half of the reviews were discarded.

For those which did receive a rating, the two numbers in the string were extracted. Dividing the first number so obtained by the second results in the computation of a ratio *r* between 0 and 1. In the example mentioned above, this ratio would have been of $\frac{2}{4} = 0.5$.

The combination of this ratio and a helpfulness threshold value allows to compute a score used to classify reviews into the two desired categories: helpful and not helpful. Mathematically, it can be expressed as follows:

$$score = \begin{cases} 1 \; if \; r \geq 0.60 \\ 0 \; otherwise \end{cases}$$

This implies that ratios inferior to the threshold are given a score of 0, while those superior to it are attributed a score of 1. Reviews with a score of 0 are then considered as ‘not helpful’ and those with that of 1, as ‘helpful’. The threshold chosen is based on a threshold widely considered as optimal in prior research. (Ghose & Ipeirotis, 2011 ; Hong, Lu, Yao, Zhu, & Zhou, 2012 ; Krishnamoorthy, 2015).

3.2.2 Cleaning of the Review Texts

The reviews which appeared without any text were excluded from the dataset. In total, 70 empty-text reviews were identified. The size of the dataset was thus further decreased to 62, 807. For the remaining reviews, a total of six modifications were carried out.

The first manipulation involved setting all upper cases letters to lower case. The second one consisted in removing punctuation and replacing it by a white space. This has indeed proven to be more efficient than using the R removePunctuation function, which only removes punctuation without inserting a white space and often results in distinct words getting concatenated. An example of this undesirable phenomenon is presented below.

Raw review text

"(…) The size is great for my 2-year-old grandson to play with, and there are 2 sides to use. This is a big hit in my home and will grow with children as they become more creative...great learning tool too. (…) I believe it is also compatible with dry-erase markers for older children. I highly recommend this board...great size to travel with also."

Review text with punctuation removed using the removePunctuation function

"(…) size great yearold grandson play sides use big hit home will grow children become creativegreat learning tool (…) believe also compatible dryerase markers older children highly recommend boardgreat size travel also"

Review text with punctuation replaced by blank

"(…) size great year old grandson play sides use big hit home will grow children become creative great learning tool (…) believe also compatible dry erase markers older children highly recommend board great size travel also"

The third alteration brought was to remove numbers and the fourth, to exclude so-called stopwords. These are 174 of the most common English words. They can be discarded without affecting the meaning of a sentence because they do not bring valuable information. Amongst the top 10, one can find the words 'I', 'me', 'myself', 'we', 'our', 'ours', 'ourselves', 'you' and 'your'.

The fifth and penultimate manipulation was the stripping of whitespaces. In other words, the replacement multiple whitespaces by a single blank. Finally, the last modification consisted in the removal of leading and trailing blanks.

The example below illustrates the result of this preparation step on a sample review.

Raw review

{"**reviewerID**": "ANHM34OZMW5HN", "**asin**": "0786958731", "**reviewerName**": "Dark Trainer", "**helpful**": [1, 3], "**reviewText**": "So I liked the idea of co-op. Avoiding my nephews and kids hitting constant competition and some kid whining about losing, or another rubbing it in as a bad winner, etc. This game was meant to offer them a simple boardgame delve into a dungeon. It does that, does it well, and comes with TONS of minis, missions, etc. That said, it's designed very well, the characters in this version stand up very well to the mobs/missions, but for some reason unlike some other games we own, the kids don't get immersed in it as well. It just kind of feels linear and you feel lead along. Maybe I need to work better at narrating or something, but that's just my 2 cents.", "**overall**": 3.0, "**summary**": "I really liked the idea of co-op. It's a very good game, easy to play too!", "**unixReviewTime**": 1358380800, "**reviewTime**": "01 17, 2013"}

Cleaned review

liked idea co op avoiding nephews kids hitting constant competition kid whining losing another rubbing bad winner etc game meant offer simple boardgame delve dungeon well comes tons minis missions etc said designed well characters version stand well mobs missions reason unlike games kids dont get immersed well just kind feels linear feel lead along maybe need work better narrating something thats just cents|0

One can first see that all upper case letters were successfully transformed into lower case. The punctuation, numbers and stopwords were all removed. There are also no extra white spaces, whether that is at the beginning, in the middle or at the end of what was originally a sentence. In this example, the ratio r mentioned previously is of $\frac{1}{3} = 0.33$, which translates into a helpfulness score of 0.

3.3 Classification: First Attempt

Once the dataset is prepared for processing, classifiers can be applied to it. As mentioned in the second chapter, the ones considered in this research are multinomial naive Bayes (MNB), random forest (RF) and stochastic gradient descend (SGD). They were implemented in the programming language Python.

Below, in Table 1, are reported the previously presented performance metrics obtained by running these three classifiers using 10-Fold Cross Validation.

		PRECISION	RECALL	F1-SCORE
MNB	0	0.28	0.00	0.00
	1	0.72	1.00	0.83
	Weighted average	**0.59**	**0.72**	**0.60**
RF	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**
SGD	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**

Table 1. Classification report of MNB, RF, SGD on the initial dataset. Class 0 represents the class of unhelpful review, while class 1 represents that of helpful reviews.

The random forest and stochastic gradient descent classifiers lead to comparable results. When looking at the metrics of class 1, one can see that a little less than 75% of the reviews predicted as helpful are actually helpful, while 100% of the helpful reviews are correctly predicted as such. While these seem like reasonable results, a glance at the metrics of class 0 allows to identify a perverse effect: all reviews are predicted as helpful. These are cases of naive classifiers, as previously introduced.

In the case of MNB, the metrics related to class 0 are slightly better, while those of the other class are equally as good as for the previously mentioned classifiers. The improvement is due to the fact that the perverse effect explained above is not observed. The class 0 is predicted.

However, only 81 reviews out of 62,807 are predicted not helpful and 58 are wrongly predicted as such.

These classifiers can be considered as baseline algorithms in terms of performance for the considered dataset. All the models which will be developed later on will be compared against these references. The aim is naturally to develop models which outperform these baselines. However, before doing so, one has to first investigate the reasons of the poor performances of the classifiers.

3.4 Data Exploration

3.4.1 Class Distribution

A frequent question in data exploration is the distribution of the observations between the classes. The graph in Figure 3 shows the number of reviews in each class.

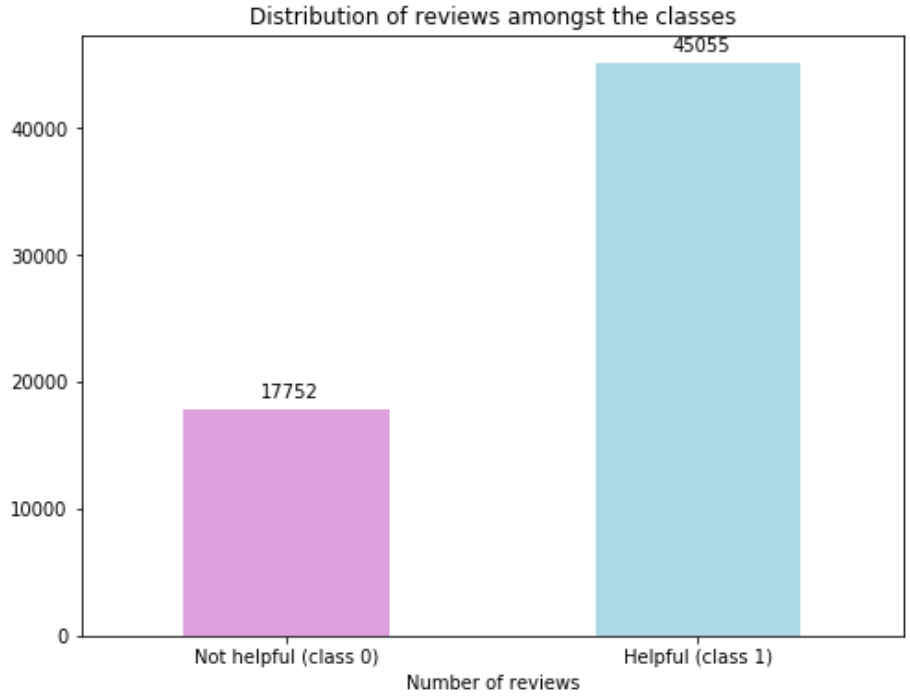

Figure 3. Bar graph representing the distribution of reviews amongst the two classes.

What Figure 3 above allows to highlight is that the classes are imbalanced. In fact, the imbalance ratio is of 1:3.

As mentioned in the second chapter, most classification algorithm tend to perform poorly with imbalanced datasets. The specific reasons that will be argued in this research are the following:

First, SVMs (and thus SVMs with SGD learning) are characterized by a loss function associated to misclassifications. Since unhelpful samples are underrepresented, they contribute less to the loss function than the helpful samples. What is observed here in the case of SVMs and imbalanced data is that the separating hyperplane is shifted towards the minority class. The further the hyperplane is from the helpful samples, the closer their hinge loss value will be to 0.

Then, random forest consists of multiple decisions trees. Each tree is trained on samples chosen amongst the input dataset, with replacement. Since there are fewer unhelpful reviews, there is a certain probability that the imbalance is amplified in the training set of each tree. Each tree receives thus an insufficient number of unhelpful samples to learn from.

Lastly, for MNB, the justification is not so straightforward. The Python implementation offers only very little parameter tuning of the model. According to the literature, similarly to linear SVMs, MNB attribute less weights in the decision boundary (the hyperplane in the case of SVMs) to underrepresented samples. It is also not very apparent why MNB works slightly better than the two other models.

3.4.2 Corpus Comparison

One can also rightfully ask themselves which features distinguish helpful reviews from unhelpful ones. The question is thus to determine the words which are most correlated with each class. Table 2 shows the result of the ten most frequent terms in the set of reviews belonging to the class ‘helpful’ (left) and that in the set of those belonging to the class ‘not helpful’ (right).

10 most frequent words in helpful reviews:	**10 most frequent words in unhelpful reviews:**
1) one	1) one
2) can	2) toy
3) game	3) like
4) like	4) can
5) toy	5) game
6) just	6) just
7) play	7) play
8) old	8) old
9) will	9) will
10) little	10) fun

Table 2. Top 10 most frequent terms in the set of reviews belonging respectively to the class helpful (left) and to the class not helpful (right)

An observation that can be made from Table 2 is that, except for the last term, both lists contain the same terms in different order.

These lists can be extended to include all the words respectively in the set of helpful reviews and the set of unhelpful reviews, rather than just the ten first, as well as their corresponding frequency. For the sake of clarity, the first will be referred to as the list of helpful words, while the second one will be referred to as the list of unhelpful words.

To be able to quantify exactly how similar the set of words in both classes are, several correlation coefficients can be computed. In the literature, Huang (2008) suggests measuring the similarity of text documents using the Jaccard coefficient and the cosine similarity.

The *Jaccard index* is a measure of similarity between sets. It is defined as the size of the intersection of the sets divided by the size of their union.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

This measure ranges between 0 and 1, 0 indicating that the sets share no member and 1 that they share all members.

When applied to the lists of helpful words and unhelpful words, the resulting index equals to 0.3927. One would conclude that the two sets are dissimilar. However, it appears that the size of the first set is bigger than that of the second one (53,466 versus 31,256). The difference in size could explain the low score.

A more relevant question would thus be to what extent the smaller set is contained in the bigger one or in other words, what the degree of overlap between the two is. To answer these questions, the *overlap coefficient*, which derives from the Jaccard index, can be computed (Majumder, Pakray, Gelbuk and Pinto, 2007). It is mathematically defined as:

$$overlap(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

The overlap coefficient for the lists of helpful and unhelpful words is 0.7643. This seems to indicate while the lists are not similar according to the Jaccard index, over ¾ of the words from the list of unhelpful words are also included in the list of helpful words.

Another approach to this is to consider the two lists of words as term-frequency vectors. In that case, their cosine similarity can be measured. Prior to that, however, the lists have to be completed in order to have the same size and therefore allow comparison. To do so, words which only appear in the list of unhelpful words are added to the list of helpful words with a frequency of zero. Symmetrically, those which only appear in the list of helpful terms are added to that of the unhelpful ones with a null frequency. As a result, the list of helpful terms is increased from 53,466 to 60,832, while that of unhelpful terms requires the addition of 29,576 helpful words to reach the size of 60,832.

The *cosine similarity* is defined as the cosine of the angle between two vectors. Mathematically, it is derived from the Euclidian dot product formula, which is given by:

$$\mathrm{X} \cdot \mathrm{Y} = \|\mathrm{X}\|\|\mathrm{Y}\| \cos(\theta)$$

Therefore, by isolating $\cos(\theta)$, one can get the expression of the similarity metric.

$$cosim(X,Y) = \cos(\theta) = \frac{X \cdot Y}{\|X\|\|Y\|} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$$

The resulting measure is always between 0 and 1 since the vector space is positive. The closer it is to 1, the greater the similarity. A metric equal to 0 indicates that the angle between the two vectors is 90° and, in this case, that the lists have no words in common. Conversely, a cosine similarity close to 1 indicates parallel vectors or in this application, lists with a great number of common terms.

Applying this to the lists at hand yields a coefficient of 0.9945, which seems to indicate that the helpful and unhelpful words are highly similar. A variant based on the tf-idf instead of term-frequency has also been computed. The similarity index with this variant is of 1.

This high correlation and overlapping degree could be a source of explanation to the poor performance of the classifiers. If features amongst classes are fairly similar, it means that most features do not have a high predictive (or discriminating) power.

3.5 Dimensionality Reduction

In order to reduce the dimensionality of the data and potentially improve the classifiers' performance, two types of techniques can be applied to the cleaned text reviews: linguistic and frequency-based techniques. In the first category, this research focuses on the methods of stemming, lemmatization and part-of-speech tagging. In the second one, the approach developed is that of TF-IDF thresholding.

All the modified reviews were re-written in text files in the optic of applying to them the previously mentioned classifiers. In total, seven new files were created.

3.5.1 Stemming

The stem of a word is the part of that word that is common to the word itself and all its inflected forms. For nouns, the inflected forms are their plural. For verbs, they are all its conjugated forms. Adjectives and adverbs are inflected in the superlative and comparative. Stemming therefore aims at reducing the words to their stem. Stemming does not necessarily map a word to an existing word. For instance, the stem of the verb "producing" is "produc". Indeed, the inflected forms of "**produc**ing" include "**produc**ed" and "**produc**e" and these all share the common substring "**produc**".

The stemmer used here is Porter Stemmer, which is based on the Porter Stemming Algorithm (Porter, 1980). It consists of sets of rules applied under specific conditions:

(condition(s)) S1 → S2

The suffix S1 is replaced by the suffix S2, which can be null, if and only if the condition is satisfied. These rules are based on common suffixes that can be found in inflected words.
All these rules aim at removing the suffix of a given word and that of the word so obtained and so on until only the stem of the original word remains.

3.5.2 Lemmatization

A lemma is the base form of a word, as opposed to inflected forms. Lemmatization is thus the process of converting a word to its lemma. For instance, the verbs "ran", "running", "run" and "runs" are all lemmatized as "run", the bare infinitive.

The major difference between stemming and lemmatization is that stemming often creates non-existent words, whereas lemmas are actual words. For the example mentioned above, the lemma of "producing" is "produce", while its stem is "produc". The latter has no lexical meaning while "produce" has.

An efficient implementation of lemmatization implies knowing the part-of-speech of the word in the sentence. Indeed, words should be lemmatized differently based on the role they play in a sentence. For instance, in the phrase '*A famous saying.*', the correct lemma of the noun

‘saying’ is *saying*. However, in the sentence ‘*Can you hear what I am saying?*’, since ‘saying’ is a verb, the correct lemma is the bare infinitive of the verb, which is *say*.

Furthermore, in Python, the default part-of-speech tag is ‘noun’. Failing to provide the right part-of-speech tag would result in all the words being identified as noun. As a consequence, only the noun of the lemma is extracted. Therefore, verbs like ‘caring’ or ‘playing’ are not reduced to respectively ‘care’ and ‘play’ as they should be.

The WordNet lemmatizer is the one chosen in this research. WordNet is a large, publicly available, lexical database of English words. For a given input word and its part-of-speech, the lemmatizer looks up the corresponding lemma in the database and returns it. A pre-treatment is done on input words which are inflicted forms as these are not present in the entries of the database. The pre-treatment consists of morphology functions and are aimed at transforming the input into a word which belongs to the dataset.

Displayed below is an example of a review and the results of respectively stemming and lemmatization applied to each of its words.

Original text

“taboo classic board party game hall fame taboo family plays games regularly rare pull taboo team game one person team gives clues word card without saying one six taboo words whole time sand timer adds pressure game fun whenever play always plenty shouting jokes laughter beware never play team people know well example remember talking yesterday college correct plenty type clues played men versus women knowing women know well”

Number of features: 55

Stemmer

“taboo classic board parti game hall fame taboo famili play game regularli rare pull taboo team game one person team give clue word card without say one six taboo word whole time sand timer add pressur game fun whenev play alway plenti shout joke laughter bewar never play team peopl know well exampl rememb talk yesterday colleg correct plenti type clue play men versu women know women know well”

Number of features: 50

Lemmatizer

"taboo classic board party game hall fame taboo family play game regularly rare pull taboo team game one person team give clue word card without say one six taboo word whole time sand timer add pressure game fun whenever play always plenty shout joke laughter beware never play team people know well example remember talk yesterday college correct plenty type clue play men versus woman know woman know well"

Number of features: 50

Globally, stemming allows to reduce the number of features in the corpus from 60,832 to 41,306, while lemmatization to 54,286.

3.5.3 Part-of-Speech Tagging

Part-of-speech (POS) tagging is the process of assigning a word its corresponding part-of-speech. The part-of-speech represents the syntactic function of a word in a sentence. For instance, in the sentence 'She has blue eyes', *she* is a personal pronoun, *has* is a verb conjugated in the 3rd person singular in the simple present, *blue* is an adjective and *eyes* is a plural noun.

The tagging algorithm used is based on the Penn Treebank, a corpus of over 4.5 million words of American English annotated with their part-of-speech. In total, the Penn Treebank POS tag set contains 36 POS tag and 12 tags related to punctuation and currency symbols. A detailed view of this set can be found in Appendix.

The result of applying the tagging algorithm is shown below on a sample review.

Original review

My kids are 4 and 6 and they just love puzzles, so when that Amazon box arrived my kids jumped with joy! The pieces are big and bright and great for hand and eye coordination. Its enough pieces to make them slow down and think, but not so many that they become overwhelmed and frustrated. Overall it is a great puzzle I highly recommend it!

Cleaned review

kids just love puzzles amazon box arrived kids jumped joy pieces big bright great hand eye coordination enough pieces make slow think many become overwhelmed frustrated overall great puzzle highly recommend

POS tagged review

NNS RB VB NNS RB VBP JJ NNS VBD NN NNS JJ JJ JJ NN NN NN JJ NNS VBP JJ VBP JJ VBP JJ JJ JJ JJ NN RB VB

One can see that the algorithm produces mistakes. For instance, the word 'amazon' is a proper noun (NNP) and not an adverb (RB). Similarly, the word 'box' is a noun (NN) and not a verb in the non-3rd person singular present (VBP). This can be explained by the fact that in the cleaning of the review, stop words, punctuation and numbers are removed and words are re-written the lower case. These however often provide essential hints about the structure of a sentence, and thus the POS. This observation can be further confirmed by applying the same algorithm on the review before cleaning, as displayed below.

My (PRP$) kids (NNS) are (VBP) 4 (CD) and (CC) 6 (CD) and (CC) they (PRP) just (RB) love (VB) puzzles (NNS) , (,) so (RB) when (WRB) that (DT) Amazon (NNP) box (NN) arrived (VBD) my (PRP$) kids (NNS) jumped (VBD) with (IN) joy (NN) ! (.) The (DT) pieces (NNS) are (VBP) big (JJ) and (CC) bright (JJ) and (CC) great (JJ) for (IN) hand (NN) and (CC) eye (NN) coordination (NN) . (.) Its (PRP$) enough (JJ) pieces (NNS) to (TO) make (VB) them (PRP) slow (VB) down (RP) and (CC) think (VB) , (,) but (CC) not (RB) so (RB) many (JJ) that (IN) they (PRP) become (VBP) overwhelmed (JJ) and (CC) frustrated (JJ) . (.) Overall (JJ) it (PRP) is (VBZ) a (DT) great (JJ) puzzle (NN) I (PRP) highly (RB) recommend (VBP) it (PRP) ! (.)

One can then see that the error rate of the tagging algorithm is much lower when considering the full sentences.

Replacing terms by their corresponding POS allows to reduce the number of features down to 35. Since punctuation was removed in the cleaning phase, it is logical that none of the POS tags related to punctuation and symbols appear. In the remaining 36 tags, no presence of list item markers was identified in the studied reviews.

3.5.4 TF-IDF Thresholding

The underlying principle of TF-IDF thresholding is to discard terms which have a TF-IDF score below a threshold. Since the latter score is computed for a single term and a single document, one has to sum the scores of a term i for all the documents j in which it appears: $\sum_j (TF - IDF)_{ij}$.

This allows then to create a list which gives for each term, its overall tf-idf score. Ordering this list in descending order gives a clear view of the relevance of each term. Namely, it allows to identify for instance the top 10 terms with highest overall score and the bottom 10 with lowest overall score.

Features top 10	**Features bottom 10**
1) game	1) witnesses
2) toy	2) watanabe
3) old	3) unshaken
4) great	4) uekawa
5) play	5) tournamentsonics
6) year	6) tikals
7) like	7) teleportation
8) fun	8) sustains
9) little	9) stardust
10) just	10) spindash

Table 3. Top 10 most important and bottom 10 least important features in terms of TF-IDF score across all reviews

The scores range from 0.005 to 1,824.6. The threshold that are considered for this research are .6, 1, 5 and 10. These were chosen empirically. In Table 4 are displayed, for each threshold considered, the number of features respectively before and after applying tf-idf thresholding.

Threshold	Number of features kept		Number of features discarded
No threshold	60,832		0
≥ .6	22,920	(+/- 38%)	37,912
≥ 1	12,117	(+/- 20%)	48,715
≥ 5	7,673	(+/- 13%)	53,159
≥ 10	5,025	(+/- 8%)	55,807

Table 4. Number of features kept (with the corresponding percentage) and discarded for each threshold level considered

One can now analyze the impact on the classification of applying dimensionality reduction techniques to the original dataset.

3.6 Classification: Second Attempt

3.6.1 Multinomial Naive Bayes

		PRECISION	RECALL	F1-SCORE
MNB baseline	0	0.28	0.00	0.00
	1	0.72	1.00	0.83
	Weighted average	**0.59**	**0.72**	**0.60**
Stemming	0	0.26	0.00	0.00
	1	0.72	1.00	0.83
	Weighted average	**0.59**	**0.72**	**0.60**
Lemmatization	0	0.24	0.00	0.00
	1	0.72	1.00	0.83
	Weighted average	**0.58**	**0.72**	**0.60**
POS tagging	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**
TF-IDF threshold .6	0	0.26	0.01	0.03
	1	0.72	0.98	0.83
	Weighted average	**0.59**	**0.71**	**0.60**
TF-IDF threshold 1	0	0.23	0.01	0.03
	1	0.72	0.98	0.83
	Weighted average	**0.58**	**0.71**	**0.60**
TF-IDF threshold 5	0	0.25	0.01	0.03
	1	0.72	0.98	0.83
	Weighted average	**0.58**	**0.71**	**0.60**
TF-IDF threshold 10	0	0.29	0.01	0.02
	1	0.72	0.99	0.83
	Weighted average	**0.60**	**0.71**	**0.60**

Table 5. Classification report of MNB classifier when applying stemming, lemmatization, POS tagging and TF-IDF thresholding (thresholds : .6, 1, 5 and 10).

Among the linguistic techniques, it appears that stemming outperforms both lemmatization and part-of-speech tagging.

On the one hand, stemming has as result an increase in the number of correctly predicted reviews of the class 0 (not helpful). Indeed, 15 non-helpful reviews which are classified as helpful by the baseline classifier are correctly classified with stemming. On the other hand, 48 of the correctly predicted helpful reviews are now wrongly predicted as not helpful. While the first one is a desirable effect, the second one deteriorates the precision of the class 0.

As a result, although stemming seems to perform equally as poorly as the baseline classification when looking at the weighted results, the precision of class 0 is actually worse in the case of stemming. One can thus conclude that stemming does not improve the classifier's performance.

Lemmatization and POS tagging both lead to a worse performance of the classifiers, taking the first attempt as reference. While the decrease is not so significant in the case of lemmatization, in the second case, however, the effect is drastic. One can indeed see that applying POS tagging basically results in all reviews being classified as helpful, or, equivalently, in a degenerated classifier.

In the case of lemmatization, only undesirable effects are observed: a part of the reviews belonging to class 0 which were correctly predicted by the baseline classifier are now wrongly predicted. At the same time, a similar transfer is observed for class 1; helpful reviews predicted as such in the brute force classification are now predicted as not helpful.
These effects deteriorate the metrics of both classes, although this deterioration only seems to be statistically significant for class 0. It is straightforward that decreasing the number of true positives affects the recall negatively, the sum of the true positives and false negatives remaining unchanged. The precision is negatively influenced both by a diminution of the true positives and an increase of the false positives.

The effects of lemmatization and stemming when using the multinomial naive Bayes classifier are rather unfavorable. In case of absolute necessity, Porter's stemmer would be preferred as its influence on the precision is not so significant.

Applying TF-IDF thresholding seems to have no significant impact on the metrics of class 1. Regarding class 0, the precision seems to first decrease as the threshold increases until it slightly picks up when a threshold of 5 is applied, and exceeds the baseline precision when a threshold of 10 is applied. The recall, on the other hand, improves for all threshold levels (in comparison with the baseline) but rather insignificantly.

A deeper analysis allows to highlight that the precision of class 0 starts to improve from a certain point only because the total number of predicted unhelpful reviews (as a percentage of the total number of reviews) decreases and with that the total number of correctly predicted unhelpful reviews (once more, as a percentage of the total number of reviews). This can be seen in the last two columns of Table 6. It can also be seen that the percentage of true negatives peaks at a threshold of 2.

What can also be seen from Table 6 is that the recall of class 0 improves when applying TF-IDF thresholding. However, this improvement is so small that it is not visible when the metrics are expressed as ratios rather than percentages. More precisely, the recall first increases and then decreases, recording a peak at a threshold of 2.

These two effects are encompassed in the F1-score. Once more, expressing the metrics in percentages rather than ratios allows to point out that the F1-score of class 0 is higher than in the baseline model for all thresholds, and highest when applying a TF-IDF threshold of 2.

Overall, when choosing a threshold of 2, the weighted F1-score only improves from 59.97% to 60.19%. Still, it will be preferred to the baseline model as the number of predicted unhelpful reviews increases from 81 to 1256 and the correctly predicted unhelpful reviews climbs from 23 to 288. This will be considered as improvement as it creates a model which is further from a degenerated classifier.

Model	TN	FN	P (class 0)	R (class 0)	F1-score (class 0)	TN+FN /total	TN/total
Baseline (no threshold)	23	58	28.40%	0.13%	0.26%	0.13%	0.04%
.6	245	706	25.76%	1.38%	2.62%	1.51%	0.39%
1	261	851	23.47%	1.47%	2.77%	1.77%	0.42%
2	288	968	22.93%	1.62%	3.03%	2%	0.46%
5	258	789	24.64%	1.45%	2.74%	1.67%	0.41%
10	217	520	29.44%	1.22%	2.35%	1.17%	0.35%

Table 6. Metrics of class 0 (in percentages) when applying TF-IDF thresholding on the dataset used on the MNB classifier.

One can thus conclude that none of the techniques shows significant improvement of the performance of the MNB classifier. Part-of-speech tagging deteriorates it and results in a naive classifier. Stemming will be preferred to lemmatization as it performs equally as poorly as the baseline MNB classifier, while lemmatization leads to a lower specificity. The effect of TF-IDF thresholding is unclear to the naked eye. However, further analysis allows to point out that applying a threshold of 2 leads to fairly insubstantial increase of the weighted F1-score. This will still be considered as (modest) improvement as it generates a less degenerated model.

3.6.2 Random Forest

None of the techniques have an effect on the classifier. Its performance remains thus the same as in the case of the first attempt. This is problematic given the fact that the class 0 is actually never predicted as mentioned before. A potential explanation of this might be that random forests (decision trees) have embedded feature selection.

		PRECISION	RECALL	F1-SCORE
	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**

Table 7. Classification report of RF classifier when applying stemming, lemmatization, POS tagging and TF-IDF thresholding (thresholds: .6, 1, 5 and 10).

3.6.3 Stochastic Gradient Descent

Similar conclusions as those highlighted for the random forest can be drawn for the SGD classifier. SVMs, and thus SVM with SGD learning, are rather independent of the dimension of the feature space and quite robust to useless features.

3.7 Balancing Data

In view of balancing the dataset and improve the performance of the classifiers, re-sampling techniques will be tested in this section.

As previously mentioned, prior studies have reached conflicting results with regards to whether under-sampling or over-sampling leads to a better performance. As a result, three re-sampling methods will be applied. These are the methods of random under-sampling, random over-sampling and Synthetic Minority Over-sampling TEchnique (SMOTE).

Random under-sampling is an under-sampling method which randomly selects samples from the majority class to eliminate and does so until the classes are equal in size. As pointed out by Liu (2004), on top of being relatively straightforward, this method has been proven to be one of the most effective and usually more effective than the more sophisticated methods. Applying this under-sampling technique to the dataset used in this research reduces the number of observations in each class to 17,752.

Applying over-sampling on the other hand will have for effect to increase the size of the class 0 to 45,055 observations.

With random over-sampling, samples are randomly selected from the minority class, with replacement and from the original dataset, and copied into the new dataset until the classes are balanced.

The SMOTE approach (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) consists in re-sampling the minority class by creating synthetic examples. The algorithm starts with the computation of the *k* nearest neighbors for each sample belonging to the minority class. In the implementation chosen, *k* is set to 5. The nearest neighbor is defined as the sample with the smaller Euclidean distance with the one considered.

Depending on the over-sampling required, some or all of the nearest neighbors are chosen. In the imbalanced dataset used in this research, the amount of over-sampling needed is roughly 154%. Therefore, all 5 neighbors are not required.

For each sample and chosen neighbor, a new synthetic sample is created by first taking the difference between the sample and the neighbor. This difference is then multiplied by a random number between 0 and 1. Finally, the result of this last operation is added to the original sample.

Let us now analyze the results of applying these three re-sampling methods to the same three models that have been studied all along.

3.8 Classification: Third Attempt

3.8.1. Multinomial Naive Bayes

		PRECISION	RECALL	F1-SCORE
MNB baseline	0	0.28	0.00	0.00
	1	0.72	1.00	0.83
	Weighted average	**0.59**	**0.72**	**0.60**
Random under-sampling	0	0.57	0.51	0.54
	1	0.55	0.61	0.58
	Weighted average	**0.56**	**0.56**	**0.56**
Random over-sampling	0	0.62	0.75	0.68
	1	0.68	0.54	0.60
	Weighted average	**0.65**	**0.64**	**0.64**
SMOTE	0	0.63	0.77	0.69
	1	0.70	0.55	0.61
	Weighted average	**0.67**	**0.66**	**0.65**

Table 8. Classification report of MNB classifier when applying different re-sampling methods: random under-sampling, random over-sampling and SMOTE.

The first observation that can be drawn from Table 8 is that all re-sampling methods lead to an increased performance of the MNB classifier.

Moreover, out of the two over-sampling techniques, SMOTE leads to better precision, recall and F1-score than random over-sampling. As a result, when combined with the MNB classifier, SMOTE will be preferred to random over-sampling.

It can be further noted that, in this case, both over-sampling techniques seem to outperform random under-sampling. It is also interesting to note that, considering the F1-score, over-sampling tends to classify better on class 0 than class 1, in contrast with under-sampling.

3.8.2 Random Forest

		PRECISION	RECALL	F1-SCORE
RF baseline	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**
Random under-sampling	0	0.56	0.67	0.61
	1	0.59	0.47	0.52
	Weighted average	**0.57**	**0.57**	**0.57**
Random over-sampling	0	0.56	0.69	0.62
	1	0.60	0.46	0.52
	Weighted average	**0.58**	**0.58**	**0.57**
SMOTE	0	0.67	0.80	0.73
	1	0.75	0.61	0.67
	Weighted average	**0.71**	**0.71**	**0.70**

Table 9. Classification report of RF when applying different re-sampling methods: random under-sampling, random over-sampling and SMOTE

Two main conclusions can be drawn from the results displayed in Table 9.

First, re-sampling the classes clearly and significantly improves the performance of the RF classifier. Out of the three re-sampling techniques, SMOTE seems to be the winner, once more. Between random under-sampling and random over-sampling, the gap in performance is fairly tight. Still, random over-sampling leads to higher precision, recall and F1-score.

Then, one can notice that when SMOTE is applied, the RF model largely defeats the MNB one. It is interesting to highlight that before re-sampling, the RF classifier yielded a degenerated model where all reviews were predicted as helpful and none as unhelpful. After applying SMOTE, a very large percentage (80%) of unhelpful reviews are correctly predicted, while 60% of the helpful reviews are categorized as such. Re-sampling results thus in the shift from a very poor model in regards with class 0 to one which is more performant for class 0 than class 1. This was also observed in the case of the MNB classifier.

3.8.3 Stochastic Gradient Descent

		PRECISION	RECALL	F1-SCORE
SGD baseline	0	0.00	0.00	0.00
	1	0.72	1.00	0.84
	Weighted average	**0.51**	**0.72**	**0.60**
Random under-sampling	0	0.56	0.67	0.61
	1	0.59	0.47	0.52
	Weighted average	**0.57**	**0.57**	**0.57**
Random over-sampling	0	0.59	0.79	0.68
	1	0.69	0.46	0.55
	Weighted average	**0.64**	**0.62**	**0.61**
SMOTE	0	0.64	0.78	0.70
	1	0.72	0.55	0.63
	Weighted average	**0.68**	**0.67**	**0.67**

Table 10. Classification report of SGD classifier when applying different re-sampling methods: random under-sampling, random over-sampling and SMOTE

Unsurprisingly, re-sampling also improves classification with SGD. When combined with the SGD classifier, the most efficient re-sampling technique is SMOTE. It seems thus that amongst the three studied re-sampling methods, SMOTE is the most accurate one, regardless of the classifier considered. When SMOTE is applied, neither MNB nor SGD dethrone RF.

In the case of SGD, random over-sampling outperforms random under-sampling quite notably. It seems thus that over-sampling is a better choice, regardless of the model at hand.

4 PROJECT MANAGEMENT

The aim of this chapter is on the one hand to present some theoretical aspects related to project management and on the other hand, to explain how this thesis relates to a structured project management approach.

4.1 Theoretical Framework

All the theoretical information was collected from a seminar held at HEC Liège in September 2018 by Dr. Stijn Van de Vonder, Frédéric Debouche and Prof. Dr. Stefan Creemers.

4.1.1 What is a project?

A project is formally defined by the Project Management Institute (2017) as "a temporary endeavor undertaken to create a unique product, service or result".

Furthermore, they identify four common characteristics to all projects. They must be important and have well-defined deliverables, yet be temporary and developed progressively. This progressive development manifests itself through what is called the project life cycle: each project is divided into a number of phases. Their number depend on the industry or on the nature of the project at hand.

This growing interest for projects results in a silent transition from functional organizations to project-oriented organizations. Operational tasks within companies are thus increasingly being defined as projects. With that shift comes a need for project management techniques and tools.

4.1.2 What is project management?

Project management is defined as the "application of knowledge, tools, skill, and techniques to activities in order to meet project requirements" (Project Management Institute, 2017).

The primary challenge of project management is to meet the project requirement while complying with six constraints:

- ◊ Schedule: the project must be delivered within a certain time limit.
- ◊ Budget: the project must respect the financial limit on the total amount that can be spent.
- ◊ Scope: the scope (i.e. the boundaries of the project in terms of goals, deliverables and tasks) must be respected throughout the execution of the project.
- ◊ Quality: the project must meet the requestor's requirements and expectations.
- ◊ Resource: human and material resources are scare and must be shared across activities.
- ◊ Risk: any event that can compromise the successful execution of the project must be managed and ultimately mitigated.

These constraints are interconnected. An issue with any of the constraints is likely to impact at least one of the others. For instance, if the right resources are not available or not at the right time, it might cause a delay in the execution of the project and the time to deliver might increase. Not disposing the necessary resources could also cause an increase in project cost, as obtaining resources on short notice can be expensive. Additionally, if the alternate resources do not meet the usual quality standard, the quality of the deliverable could also be affected.

To guarantee that good development of a project, each actor receives a specific role. Three main types of roles can be identified:

- ◊ Project sponsor: the owner of the business case (i.e. document which encompasses the reasoning for initiating a project) of the project. In larger projects, the sponsor is part of the project board, which includes senior users and senior suppliers. Senior users are representatives of the final users of the deliverable, while senior suppliers represent the interests of the suppliers.
- ◊ Project manager: the person who receives a request from the sponsor to manage the executive of a phase or all phases of the project. The role of the project manager is thus to ensure that the project is delivered with respect to the deadline, scope and budget.
- ◊ Project team: people who execute the project. These are often experts who report to the project manager and are under his steering.

4.2 Application: the case of a research thesis

A thesis can be considered as a project as defined in the first section of this chapter. First of all, a thesis is worth about one third of the credits of an academic year. It is thus undeniably important. Then, it has clearly defined deliverables: it must be an analysis of a management problem dealt with in the form of a research question. Moreover, the production of the Master thesis is temporal: it extends over a single academic year. Lastly, it consists in a progressive development. As mentioned previously, a thesis is a task that is done gradually, throughout a whole year and which has defined phases.

As discussed in the first section of this chapter, the project life cycle is heavily dependent on the task at hand. In the case of this Master thesis, one can identify four stages:

The first stage involved choosing the topic. As I am pursuing a master in Digital Business, the first step was to find a topic related to this field. Through a discussion with my thesis supervisor, several propositions were made available to me. After further consideration, I chose the prediction of online reviews as it appealed to me the most, and seemed to coincide the most with my interests.

The second step consisted in the definition of the scope of the project. Together with my thesis supervisor, we determined the content of the thesis, and more precisely, the tasks that had to be carried out. The scope was open-ended: the aim was to go as far as possible in the conception of the model and its variants. Still, there were some minimum requirements were defined. We did not set time constraints on the individual tasks. As this project did not involve any external stakeholders, I could allow myself to be more flexible on the timeline. In retrospect, it did not prevent me from working efficiently.

Following this second step, the project implementation started. It mainly consisted in the coding of the different models that were used in this Master thesis. It also naturally involved some prior research on the models. Due to unforeseen events and unexpected results, I often had to meet with my supervisor and re-define the scope of the project. The working methodology adopted here could thus be considered as agile.

The fourth and last step in the production of the Master thesis was its formulation in writing. In this step, a time constraint was imposed on the delivery date.

Managing this kind of project is undeniably less constraining than in the case of a project managed by a company. Indeed, there were no budget limitations nor real time constraints (besides the final delivery deadline). As stated before, I worked on my own and could therefore manage the execution of the tasks as I wished. I did though need to partition my time between working on my thesis and working on other university-related projects. I found however that dedicating one day per week to my thesis was efficient.

Concerning the scope, I did not experience any constraints related to it either. The goal of this research was… to research and the scope evolved in the direction where my work took me. This is not surprising given that in this context, I had the three roles of project sponsor, project manager and project executer. I could thus easily adapt the initial requirements.

I did however have to deal with quality constraints. The thesis is subjected to an assessment by a jury. The assessment aims at examining whether the thesis meets the academical requirements. I also imposed quality constraints on my own self. A thesis is an important piece of work that should reflect one's academic path.

I also faced unexpected events. The first notable one was the poor performance of my three baseline models. I did however investigate the cause and tried to mitigate it through pre-processing techniques. This lack of success of this solution constituted the second unforeseen event. Eventually, I managed to mitigate its effect by applying re-sampling techniques. I had not done any prior risk analysis, as one should in a company project. I deemed it was not necessary, given that there was no financial exposure. I also believe that unexpected events enrich research and help get a deeper understanding of the topic. This is not necessarily true for company projects where a clear product/service has to be delivered.

In terms of resources, whenever my own knowledge would restrain me, I would ask help to my supervisor, who has expertise in the domain. I also found a lot of help on online fora.

In a nutshell, a research thesis can be considered as a project. Indeed, it satisfies the four criteria that projects typically present: important, well-defined deliverables, temporary and developed gradually. However, the production of a research thesis does not follow the typical framework of project management.

The primary reason of this is that its production is somewhat more informal than those of projects carried out by companies. As a result, they are less constrained by time or budget and resource management plays a limited role. On the one hand, a research thesis involves very few stakeholders, which greatly simplifies communication and does not call for specific role definitions. On the other hand, its main material resource is past research, which in this case was fairly abundant.

The second reason is the nature of the thesis: with research, it is generally harder to define a clear-cut scope beforehand.

Still, the production of a Master thesis can be incorporated within the project management approach, even though it does not perfectly meet the definition of it.

5 CONCLUSION

Online reviews constitute a wealth of information. More and more online retailers (such as Amazon) are relying on peer reviews. The goal is then to determine which reviews are most helpful, and help customers by displaying these first. However, this method is not fully reliable as claimed by Liu, Cao, Lin, Huang and Zhou (2007). This rating system presents indeed several biases: imbalance vote bias, winner circle bias and early bird bias. The first one is at the heart of one of the greatest challenges in text classification: the curse of class imbalance. To overcome these biases, researchers are looking into automatic ways of determining helpfulness.

The goal of this research was on the one hand, to study the classification algorithms which are most performant for the task of helpfulness prediction and on the other hand, to investigate the techniques which enhance their performance in the context of class imbalance.

The three considered models were multinomial naive Bayes (MNB), random forest (RF) and support vector machine with stochastic gradient descent learning (SGD). They were first implemented using the original Amazon dataset as input, which underwent only the standard modifications that are required to be treatable by classification algorithms. This constituted the first attempt at classification.

The results of this first attempt were rather poor: while MNB showed insufficient performance towards the unhelpful class (class 0), RF and SGD lead to degenerated classifiers only predicting the helpful class (class 1).

This research attributes these to two factors. First, class imbalance. Most classification algorithms are indeed known to show bias to the majority class (in this case, the helpful class) when data is imbalanced, as shown by Ganganwar (2012). Then, the high correlation between the helpful and unhelpful features. This could potentially make it complex for classifiers to determine the most discriminating features.

The original dataset underwent then additional pre-processing so as to reduce the dimensionality of the data. In that context, the techniques of lemmatization, stemming, part-of-

speech tagging and TF-IDF thresholding were applied. Each technique led to the creation of a new dataset, which was then given as input to the three studied models.
In this second attempt, the techniques of lemmatization and stemming showed respectively no and marginal improvement on MNB, while POS tagging resulted in a degenerated MNB classifier. The ideal TF-IDF threshold level for MNB was determined empirically and is equal to 2. The resulting classifier is preferred to all the other MNB ones as it leads to an increase of predicted unhelpful reviews, in absolute terms.
None of the pre-processing techniques had any effect on RF and SGD, which remained biased towards the helpful class.

In a third classification attempt, the original dataset was re-sampled using three techniques: random under-sampling, random over-sampling and SMOTE. Re-sampling in general resulted in a drastic increase of performance for all classifiers. This confirms that the main issue with the first attempt was class imbalance. Moreover, SMOTE systemically outperformed the other two techniques, while random over-sampling proved to lead to better results than random under-sampling.

All in all, the best performing model in the first attempt was MNB with an F1-score of class 0 of 0.00 and that of class 1 of 0.83 (support-weighted average: 0.60). In the second attempt, TF-IDF thresholding applied to MNB showed the most improvement: 0.03 for the F1-score of class 0 and 0.83 for that of class 1 (support-weighted average: 0.60).
The overall winner results from the third attempt: RF with an F1-score of 0.73 for class 0 and 0.67 for class 1 (support-weighted average: 0.70).

Three main areas of improvement exist. For this research, 11 datasets were tested, including the original one. However, all were derived from a single dataset. One could thus reproduce the experiment on different datasets. This would allow to draw more general conclusions.

Then, the models here only considered the text review as input. As mentioned before, recent research has shown that other elements contribute to helpfulness. One could thus imagine incorporating additional features such as review length, readability, emotion, and so on.

Lastly, one could also combine both dimensionality reduction and re-sampling techniques in a single dataset. However, given the marginal improvement dimension reduction has shown, one is not to expect substantial impact on the performance of the algorithms.

TABLE OF CONTENT

BIBLIOGRAPHY

[1] Alelyani, S., Tang, J., & Liu, H. (2013). Feature Selection for Clustering: A review. In C.C. Aggarwal & C.K. Reddy (Eds.), *Data Clustering: Algorithms and Applications*. (pp. 29-60). Boca Raton: Chapman and Hall.

[2] Ali, A., Shamsuddin, S.M., & Ralescu, A. (2015). Classification with class imbalance problem: A Review. *International Journal of Advances in Soft Computing and its Applications*, *7*(3), 176-204. Retrieved from http://home.ijasca.com/data/documents/13IJASCA-070301_Pg176-204_Classification-with-class-imbalance-problem_A-Review.pdf

[3] Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random Forest and Decision Trees. *International Journal of Computer Science Issues*, *9*(3), 272-278. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.3863&rep=rep1&type=pdf

[4] Ben-Hur, A., & Weston, J. (2010). A User's Guide to Support Vector machines. In O. Carugo & F. Eisenhaber (Eds.), *Methods in Molecular Biology*. (pp. 223-239). New York: Humana Press.

[5] Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. Retrieved from https://www.cmi.ac.in/~madhavan/courses/dmml2019jan/literature/Breiman1996_Article_BaggingPredictors.pdf

[6] Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. doi : 10.1023/A:1010933404324

[7] Burez, J., & Van den Poel, D. (2009). Handling Class Imbalance in Customer Churn Prediction. *Expert Systems with Applications*, *36*(3), 4626–4636. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.477.1151&rep=rep1&type=pdf

[8] Charrada, E.B. (2016). Which One to Read? Factors Influencing the Usefulness of Online Reviews for RE. *Proceedings of the IEEE 24th International Requirements Engineering Conference Workshops*, 46-52.

[9] Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321-357. doi: 10.1613/jair.953

[10] Chen, P-Y., Wu, S-Y., & Yoon, J. (2004). The Impact of Online Recommendations and Consumer Feedback on Sales. *Proceedings of the International Conference on Information Systems ICIS 2004*, 711– 724.

[11] Chevalier, J.A., & Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research*, *43*(3), 345–354. doi: 10.3386/w10148

[12] Cutler, A., Cutler, D.R., & Stevens, J.R. (2012). Random Forests. In C. Zhang & Y. Ma (Eds.), *Ensemble Machine Learning : Methods and Applications* (pp. 157-176). New York : Springer.

[13] Diaz, G.O, & Ng, V. (2018). Modeling and Prediction of Online Product Review Helpfulness: a Survey. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 698-708.

[14] Drummond, C., Holte, C. (2003). C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. *Proceedings of the International Conference on Machine Learning Workshop Learning from Imbalanced Data Sets II*, 1-8.

[15] eMarketer. (2010). *The Role of Customer Product Reviews: Undeniably influential*. Retrieved from: https://www.emarketer.com/Article/Role-of-Customer-Product-Reviews/1008019

[16] Ganganwar, V. (2012). An Overview of Classification Algorithms for Imbalanced Datasets. *International Journal of Emerging Technology and Advanced Engineering*, *2*(4), 42-47. Retrieved from https://pdfs.semanticscholar.org/239b/2210b3fbc1f4b8246437a88a668bf9a0d2c0.pdf

[17] Ghose, A., & Ipeirotis, P.G. (2010). Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics. *IEEE Transactions on Knowledge and Data Engineering*, *23*(10), 1498-1512. doi: 10.1109/TKDE.2010.188

[18] He, R., & McAuley, J. (2016). Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. *Proceedings of the 25th International Conference on World Wide Web*, 507–517.

[19] Hong, Y., Lu, J., Yao, J., Zhu, Q., & Zhou, G. (2012). What Reviews are Satisfactory: Novel Features for Automatic Helpfulness Voting. *Proceedings of the 35th international ACM SIGIR Conference on Research and Development in Information Retrieval*, 495–504.

[20] Hossin, M., & Sulaiman, M.N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Proces*, *5*(2), 1-11. doi : 10.5121/ijdkp.2015.5201.

[21] Huang, A. (2008). Similarity Measures for Text Document Clustering. *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, 49–56.

[22] Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text Classification Using Machine Learning Techniques. *WSEAS Transactions on Computers*, *4*(8), 966-974. Retrieved from https://www.researchgate.net/profile/V_Tampakas/publication/228084521_Text_Classification_Using_Machine_Learning_Techniques/links/0c96051ee1dfda0e74000000.pdf

[23] Joachims, T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning*, *1398*, 137–142.

[24] Kibriya, A.M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial naive Bayes for text categorization revisited. In G.I. Webb & X. Yu (Eds.), *AI 2004: Advances in Artificial Intelligence*. (pp. 488-499). Heidelberg : Springer.

[25] Koulis, T. (2003) *Random Forests: Presentation Summary*. Unpublished document, University of Waterloo, Waterloo.

[26] Kowalczyk, A. (2017). *Support Vector Machines Succinctly*. Retrieved from https://www.syncfusion.com/ebooks/support_vector_machines_succinctly

[27] Krishnamoorthy, S. (2015). Linguistic features for review helpfulness prediction. *Expert Systems with Applications*, *42*(7), 3751-3759. doi: 10.1016/j.eswa.2014.12.044

[28] Lan, M., Tan, C.L., Su, J., & Lu, Y. (2009). Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(4), 721-735. doi : 10.1109/TPAMI.2008.110

[29] Lemaître, G., Nogueira, F., & Aridas, C.K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, *18*(1), 1-5. Retrieved from http://www.jmlr.org/papers/volume18/16-365/16-365.pdf

[30] Liu, A.Y-C. (2004) *The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets* (Master's thesis, University of Texas, Austin). Retrieved from https://pdfs.semanticscholar.org/cade/435c88610820f073a0fb61b73dff8f006760.pdf

[31] Liu, J., Cao, Y., Lin, C-Y., Huang, Y., & Zhou, M. (2007). Low-Quality Product Review Detection in Opinion Summarization. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 334– 342.

[32] Majumder, G., Pakray, P., Gelbukh, A. F., & Pinto, D. (2016). Semantic Textual Similarity Methods, Tools, and Applications: A Survey. *Computación y Sistemas*, *20*(4) 647–665. doi: 10.13053/CyS-20-4-2506

[33] Maloof, M. (2003). Learning When Data Sets are Imbalanced and When Costs are Unequal and Unknown. *Proceedings of the International Conference on Machine Learning Workshop on Learning from Imbalanced Data Sets II*, 73-80.

[34] McAuley, J., Targett, C., Shi, Q., & van den Hengel, A. (2015). Image-based Recommendations on Styles and Substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 43–52.

[35] McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 41-48.

[36] Park, S., & Nicolau, J.L. (2015). Asymmetric Effects of Online Consumer Reviews. *Annals of Tourism Research*, *50*, 67–83. doi: 10.1016/j.annals.2014.10.007

[37] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A.,

Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *JMLR*, *12*, 2825-2830. Retrieved from http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[38] Peng, F., Schuurmans, D., & Wang, S. (2004). Augmenting Naive Bayes Classifiers with Statistical Language Models. *Information Retrieval, 7*(3-4), 317–345.

[39] Porter, M.F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, *14*(3), 130–137. Retrieved from https://cl.lingfil.uu.se/~marie/undervisning/textanalys16/porter.pdf

[40] Prati, R.C., Batista, G.E., & Monard, M.C. (2009). Data mining with imbalanced class distributions: concepts and methods. *Proceedings of the 4th Indian International Conference Artificial Intelligence*, 359–376.

[41] Project Management Institute. (2017). *What is Project Management*? Retrieved from https://www.pmi.org/about/learn-about-pmi/what-is-project-management

[42] Rastgoo, M., Lemaitre, G., Massich, J., Morel, O., Marzani, F., Garcia, R., & Meriaudeau, F. (2016). Tackling the Problem of Data Imbalancing for Melanoma Classification. *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies, 2*, 32-39. doi: 10.5220/0005703400320039

[43] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1-47. doi: 10.1145/505282.505283

[44] Siering, M., & Muntermann, J. (2013). What Drives the Helpfulness of Online Product Reviews? From Stars to Facts and Emotions. *Proceedings of the 11th International Conference on Wirtschaftsinformatik*, 103-118.

[45] Sokolova, M., & Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing & Management*, *45*(4), 427–437. doi : 10.1016/j.ipm.2009.03.002

[46] Tisserant, G., Prince, V., & Roche, M. (2013). GenDesc: A Partial Generalization of Linguistic Features for Text Classification. In E. Métais, F. Meziane, M. Saraee, V. Sugumaran & S. Vadera (Eds.), Natural Language Processing and Information Systems (343-348). Heidelberg : Springer.

[47] Toman, M., Tesar, R., & Jezek, K. (2006). Influence of Word Normalization on Text Classification. *Proceedings of InSciT, 2*, 354-358.

[48] Veganzones, D., & Séverin, E. (2018). An investigation of bankruptcy prediction in imbalanced datasets. *Decision Support Systems, 112*, 111-124. doi:10.1016/j.dss.2018.06.011

[49] Yang, Q., & Wu, X. (2006). 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology and Decision Making*, *5*(4), 597-604. Retrieved from http://cs.uvm.edu/~icdm/10Problems/10Problems-06.pdf

[50] Yang, X., Tang, K., & Yao, X. (2015). A Learning-to-Rank Approach to Software Defect Prediction. *IEEE Transactions on Reliability*, *64*(1), 234-246. doi: 10.1109/TR.2014.2370891

APPENDIX PENN TREEBANK POS TAGSET

TAG	DESCRIPTION
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential *there*
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNP	Proper noun, singular
NNPS	Proper noun, plural
NNS	Noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb
#	Pound sign
$	Dollar sign
.	Sentence-final punctuation
,	Comma
:	Colon, semi-colon
(	Left bracket character
)	Right bracket character
"	Straight double quote
‘	Left open single quote
‘‘	Left often double quote
’	Right close single quote
’’	Right close double quote

Online reviews are becoming increasingly abundant, which makes them sometimes overwhelming for the users. To mitigate the problem of information overload, online retailers often proceed to display them according to their helpfulness to other users. In recent years, research has been aimed at finding efficient ways to automatically predict review helpfulness.
This paper offers insight on both the most appropriate algorithm for the task of predicting review helpfulness in the specific context of class imbalance and high overlap of class features, and on the pre-processing techniques which can improve classifier performance in that context. To do so, it considers three classification algorithms: random forest, multinomial naive Bayes and linear support vector machine that uses stochastic gradient descent for learning.
It shows that : (1) none of the considered algorithm exhibit satisfying performance when facing imbalanced datasets and similar class features; (2) the use of linguistic pre-processing techniques results in marginal or no improvement; (3) the use of frequency-based pre-processing yields moderate improvement; (4) re-sampling techniques are highly efficient, especially Synthetic Minority Over-sampling TEchnique (SMOTE); (5) Overall, random forest combined with SMOTE shows the best performance in terms of precision, recall and F1-score.