Universität Ulm
The Faculty of Mathematics
and Economics

Determining Information Quality in Social Networks using Message Classification Techniques

Master Thesis

in Management and Economics

Leonid Edelmann

September 11, 2017

Supervision

Prof. Dr. Mathias Klier

Prof. Dr. Leo Brecht

Table of Contents

List of Figures

1 Introduction

1.1 Motivation

placeholder

1.2 Current State of Research

2 Basic Concepts

2.1 Machine Learning

A sub-branch of computer science that rose to prominence and started evolving during the 1950s as part of research in the field of artificial intelligence. Machine learning refers the development of algorithms, which allow computers to learn from presented examples. The supposition being, the computer will thereafter be able to learn from its collected experience and automate the process of solving similar tasks. This process is coined by term *training*.

One definition of Machine - Learning, formulated by one of the fields pioneers, Tom M. Mitchell [12] "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

Nowadays machine learning algorithms are predominantly implemented as instruments for the analysis of real-world data in tasks, at which a concrete human-written application would prove ineffective. Such is the case for example with problems, which a person would be able to solve, but would not be able to determine the rules for solving explicitly. Or alternatively, where the rules are not constant, but rather evolve as time progresses. The purpose of teaching a machine to solve such tasks, probably relates to modeling, prediction or detection of details or certainties about the real world.

Vivid examples of real world uses are speech recognition as used in cellphones or in call routing systems. Another common task for such systems is visual recognition of objects or characters. For such purposes, an algorithm is trained to recognize graphic patterns in images or diagrams. This practice is even implemented in medicine for recognizing malignant anomalies in X-ray or MRI scans.

2.2 Text Mining

Text mining refers to the practice of extracting information from raw meta-data. Such data is obtained by surveying text corpora or other unstructured similar sources of verbatim data. The initial stage of pro-

cessing consists of restructuring the data into a form compatible for statistical analysis. This includes but not limited to, segmenting the text to more basic building blocks such as paragraphs or sentences. This practice is known as **stemming**. Followed by cleaning up the data by removing non-informative words, which serve a grammatical role in human language, but tend to be of no use for language processing done by machines. In the next phase, words are normalized to their base **stem** by removing inflection modifying the word's tense, case number and other grammatical properties. In the professional nomenclature, this is referred to as **stemming** or **lemmatizing**.

Methods used in text mining involve statistical pattern recognition, tagging-annotation and frequency analysis. The end goal of text mining is namely, the production of qualitative information out of raw text, often automatically, by using machine learning.

Most of the usage of text mining methods in this work will be conducted using the **NLTK** module for the Python programming language. **NLTK** is a suite of libraries developed in the Department of Computer and Information Science at the University of Pennsylvania for Natural Language Processing [2].

2.3 Unstructured Data

Data which is derived from the Internet, namely social networks is oftentimes unstructured. Unstructured data refers to information not organized uniformly or in a predefined data model of sorts. Such a model should be composited harmoniously to the structure in which the data is to be eventually utilized. The aforementioned information is prone to heterogeneity in its composition and therefore varies in data types and may contain texts, number, dates as well as multimedia objects. Additionally, the data's integrity is susceptible to fluctuation, recurrently having missing or only partial data.

In our case, the data comes in the form of Tweets originating from the Twitter servers. Tweets arrive packaged formatted as JSON objects. This type of construct has no standard morphology, and many of the Tweets' fields are subject to variation. The most common approaches to dealing with such data are either restructuring it to a new data-model or

conducting a textual analysis aimed at recognizing patterns in its structure. Text Mining and Natural Language Processing are two prevailing mechanisms employed for this purpose. More on those in Chapter 3, *Collecting the Data*.

2.4 Classification

The correct assignment of (often unstructured) data to a category (or class) out of a set of predetermined categories. In the case of unsupervised learning, creation of new categories which best segment the data and in case of unsupervised learning, assignment of new examples to user-defined classes. Supervised and Unsupervised Learning, are different approaches in Machine Learning, where the former requires continues input from the user whereas the latter is more autonomous.

Supervised Learning Using this approach the algorithm is trained on a *labeled training set*. This means that a list of numerous examples along with their features and the predetermined labeling, which denoted their class are fed as input into a learning algorithm. This algorithm than learns from the data, what common feature best characterize every given class. The different algorithms all take different approaches as to how better to split this observations. The finished machine is thereafter usually tested using novel data from the same source as the training data. The machine than measures its performance by trying to classify the testing data and comparing its classification to the actual labels. Some algorithms are even able to optimize themselves during the testing phase. Another common quality assurance test is called Cross-Validation. The data is split to m segments and trains itself in each round using $m - 1$ segments and testing against the 1 remaining segment. In each round another segment is left out for testing. This measure assures the Machine's robustness.

Unsupervised Learning In the case of Unsupervised Learning, the task of classification is commonly similar to the task of *Clustering*. During *Clustering*, the Machine does not require the input (training-) data to be anteriorly **labeled**, or classified to predetermined categories. Instead, the *Clustering* algorithm is usually given the number of categories (with

some approaches, not even that). Next, the data is fractured according to what the Machine observes to be distinguishing features of the each category. Thus extracting and selecting features is of importance, since it further enhances the prediction power for future novel data. The features are often artificial constructs or rather functions transforming the original data from its original dimensions, in order to create distinguishing attributes of the input data. Examples of such features vary from binary presence indicators of a word or character, interaction variables of two or more features to transformations, linear or otherwise. some common models in this field are Hierarchical Cluster Analysis (HCA), K-Means and Mean Shift.

2.5 Information Quality

Social Networks incorporate the interactions of millions of individuals, groups and organizations. A torrent of information of such dimensions easily qualifies for the qualification of Big Data. Social Networks data streams conform the the widely acknowledged 5 characteristics of Big Data, also knowns as the 5 **V**'s[7] standing for *Volume*, *Velocity*, *Veracity*, *Variety* and *Value*. Of interest here is the data's correspondence between *Veracity* and *Value*, referring to the trustworthiness and statistical reliability of said facts, originating from a plethora of sources and presenting little to no accountability for its correctness. How much value can indeed be extracted from this data ?

The questionable quality of such information makes basing critical business decisions on it, risky at best. One solution proposed to overcome these shortcomings would be to attach alongside the information quality metrics, which would describe its correctness, completeness and topicality as proposed by Klier and Heinrich(2016)[10].

3 Methodology

The approach undertaken in this study is similar to previous works in the field of applied Machine Learning. Initially, the main theme of the is selected according to its quality and volume. In the case of this research, the main point of inquiry I wish to study regards Tweets relating to the subject of E-Commerce platforms. Such platforms usually see an abundance of Tweets fulfilling the volume requirement. However, additional steps are taken to insure the informations quality. These cleaning measures are elaborated in the Application chapter.

On of the main motives of this research is to observe whether social media data can be evaluated in real-time and distilled into practical knowledge. With this consideration in mind, the data used in this work should bear as much resemblance as possible to a live Tweeter data stream. Acknowledging this consideration, the raw data in the form of Tweets in gathered from the Twitter databases. Namely, the data is collected synchronously, as it is intercepted by the servers. The *Streaming* Application Programming Interface (API)[18] is implemented for such use-cases. Employing the *Streaming* API, a connection is established to the servers, which captures a narrow stream (about 15%) of all Tweets relevant to a given search term. The reduction of stream is due to the complexity in both sending and receiving such a volume of data. Twitter also offers a full non-reduced access through the *Firehose* API.

The captured data, must then be restructured to fit the data-model of this study, specifically input appropriated for Machine Learning purposes. This includes a preliminary analysis of the data and removal of incomplete observations. The JSON data structure in which Tweets are stored allows for a dynamic non-standard structure, which in turn translates to non-standardized data. Therefore some Tweets must be adapted to conform to a unitary pattern. It is also common for Tweets to retrospectively be removed from the Tweeter servers, either by their owners or by Twitter moderators. Such Tweets cannot be analyzed, since they are no longer available on-line. The process of gathering data and cleansing it is discussed in Part [4.1].

The next stage entails restructuring the raw captured Tweets into datasets.

For this purpose, *features*[1] are extracted from each observation and converted to a standard array, representing the original Tweet. *Features* are unique properties of the data, which are describe it and also its owner, dependent on the approach. Different types of *Features* are used in accordance with the Learning approach undertaken. These approaches are further discussed in Part [4.2].

These features are then analyzed for consistency and correctness. Subsequently, the features are passed into different Machine Learning algorithms, in order to *train*[1] them. *Training* is the process of deducing the decision rules for classifying the data into one of the categories. This deduction is based on the information the algorithm draws from the input data. Afterwards, the empirical success of these different algorithms will be statistically measured and summarized. Additionally, implications are to be drawn about other use-cases. The classification task observed here is subjective in nature, since the classification themselves are abstract and arguable. Success in this experiment could prove that similarly subjective classification projects are practical.

3.1 Training Classifier

The purpose of *training* is to create Classifiers. A Classifier is a function which is fed new observations and automatically recognizes and *labels* them. The intrinsic decision algorithm, through which the Classifier will decide how to allocate a novel observation, usually remains hidden and operates as a black-box of sorts. Seeing that the decision rules could be numerous and considerably not intuitive for human readers, they remain hidden. Particularly so, when said algorithms are convolutional. Convolutional Machine Learning schemes, such as Deep Neural Networks, may incorporate numerous stages of parameter construction which renders them practically incomprehensible to human users. The actual implementation of all the algorithms would be programmed in the Python programming language and will primarily make use of the scikit-learn[13] module.

The data used for the purpose of this study consolidates 12.520 unique Tweets. Different approaches vary in the percentage of the data used

[1]Machine Learning nomenclature

from this corpus. In order to insure robust results, the *Hold-out Method* is used across the board. For each instance of *training* the data is split into two parts - a *training* set and testing set. The *training set* would usually be allocated the larger portion of the data (between 60% and 90%) and would be used, as the name suggests for training the Classifiers. The remaining corpus chunk (between 10% and 40%) would be used for testing the Classifiers success. During each such training-testing session, the data corpus is shuffled. This in turn means, that the training and testing sets constantly differ. This process of splitting, training and testing using different data in each iteration should produce statistically significant results. This method of constantly splitting the data randomly ensures the results robustness.

The following paragraphs expand on the different Machine Learning schemes. Figure 3 illustrates these schemes with relation to computational complexity and their interoperability.

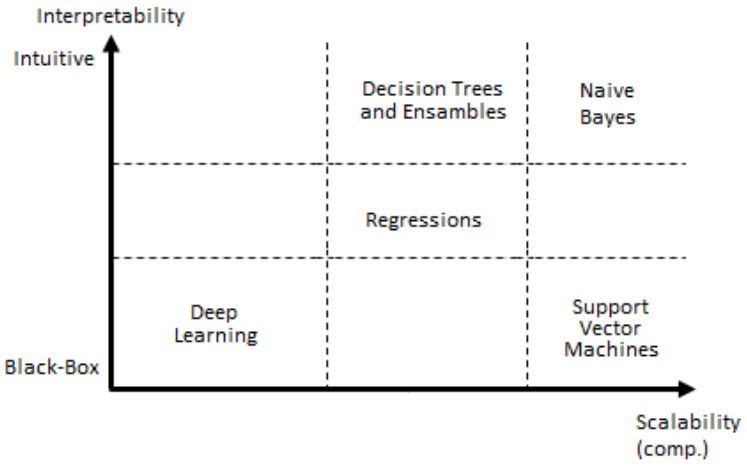


Figure 1: Clustering of Different Classifiers

3.2 Supervised Learning

3.2.1 Regressions

Probably the most basic approach is simply running a regression [1] with all the features as the independent variables and the a numeric representation of the classes as the dependent variable. Some threshold value for the classes must be determined since the estimate for the explained variable, will have a non-deterministic value. Regressions are primarily to

be used as a benchmark for the other algorithms, rather than an actual independent algorithm.

$$\mathbb{E}(\hat{y} \mid \mathbf{x}) \approx \begin{cases} label = \text{News}, & \hat{y} \geq y^* \\ label = \text{Not News}, & \hat{y} < y^* \end{cases} \tag{1}$$

Linear Regression ADD REFERENCE This method builds a linear dependency system between the explained variable (in this case, a numerical representation of the Class) and the explaining variables (Features). With the *Bag-of-Words* approach, each parameter x is a dummy variable indicating whether a given word m is present in Tweet i. The estimations parameters, which are denoted with β_j are derived using Ordinary Least Squares. In itself, the linear regression estimation is a weak predictor for the purpose of classification, but it allows for calculating other useful statistics such as the coefficient of determination, commonly known as R^2. This static demonstrates the part of the variance that is explained using the provided variables and is also called 'goodness of fit'. We therefore strive to maximize it as much as possible. The more variance covered by our variables, the better we are able to predict the outcome of the classification. Another point of interest is the distribution of predicted classifications ($\hat{y}$). In an ideal scenario, the distribution of $\hat{y}$ would be concentrated around the numerical representations of the classes. The basic premise for the use of linear regressions as classification tools, where n is the number of observations, m is the number of features [2].

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_m x_{mi} + \epsilon_i \qquad \forall\ i \in [1, n]. \tag{2}$$

Logistic Regression Despite its name, the actual regression model executed here is linear, and is used primarily for classification rather than regression analysis. This regression scheme differs from linear firstly, in the fact, that the possible outcomes of the dependent variable are discreet rather than continuous. Secondly, the probabilities which describe the different outcomes of a regression instance, are modeled using a logistic function. The discrete outcomes can in turn be converted to labels, which in allows for a more fluent application as a classification tool and is therefore commonly employed in Machine Learning. Logistic regression analysis is formally represented as follows in [3]. The x vector represents

all explanatory variables, in our case, the Tweet features. The term error ϵ follows the standard logistic distribution, which also gives the regression its name.

$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \epsilon > 0 \\ 0 & \text{else} \end{cases} \qquad \forall\ x = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix} \tag{3}$$

3.2.2 Naive Bayes

Probably the most common classification algorithm and usually the go-to classifier when handling text classification tasks [[16]]. The Naive Bayes algorithm is popular because its propensity to perform as good as much complexer classifiers, if not outperform them. This holds true, despite the overly simplified or "naive" assumptions being made at its core. This algorithm is prevailing for numerous classification problems ranging from sentiment analysis, spam filtration and up to classifying data to user-defined classes as is the case in this study. The attitude adopted in this scenario is probabilistic, since it classifies observations according to which class are they most likely to belong to, given their features. The word "Naive" in its name come from its base assumption, which is that the occurrence of a certain feature in a data point is independent from the occurrences of other features. As such, the probabilistic calculus is as in [4], where $\mathbf{X} = (x_1, ..., x_n)$ is a vector of the data features and C is the appropriate class.

$$P(\mathbf{X}|C) = \prod_{i=1}^{n} P(x_i|C) \tag{4}$$

Classification At its heart, the algorithm is based on Bayes theorem [5], which allows for the calculation of conditional probability. In the case of classification, it is the likelihood that a novel data point belong to a given class C_i, given its describing properties (features) $\mathbf{X}$.

$$P(A|B) = \frac{P(X|A) \cdot P(A)}{P(B)} \tag{5}$$

The algorithm thus calculates the posterior probability of the data point

to be part of each given class i out of k classes. After which these probabilities are than compared [6]. The classification of the new point is determined by which class is most likely (has the highest probability). In pursuance of calculating these likelihoods, we must first calculate the numerator of [6] as in [7].

$$P(C_i|x_1, ..., x_n) = \frac{P(x_1, ..., x_n|C) \cdot P(C_i)}{P(x_1, ..., x_n)} \quad \forall \quad 1 \leq \ i \leq \ k \tag{6}$$

In [7] we derive that the conditional probability term, $P(x_j|x_{j+1}, ..., x_n, C_i)$ is equal to $P(x_j|C_i)$ based on the assumption that the occurrence of certain features j is independent of the occurrences of all other features $(x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_n)$ [4].

$$\begin{aligned}
P(x_1, x_2, ..., x_n|C) \cdot P(C_i) &= P(x_1, x_2, ..., x_n, C_i) \\
P(x_1, x_2, ..., x_n, C_i) &= P(x_1|x_2, ..., x_n, C_i) \cdot P(x_2, ..., x_n, C_i) \\
&= P(x_1|x_2, ..., x_n, X_i) \cdot P(x_2|x_3, ..., x_n, C_i) \cdot P(x_3, ..., x_n, C_i) \\
&= \ ... \\
&= P(x_1|x_2, ..., C_i) \cdot P(x_2|x_3, ..., C_i) \cdot ... \\
&\qquad ... \cdot P(x_{n-1}|x_n, ..., C_i) \cdot P(x_n|C_i) \cdot P(C_i)
\end{aligned} \tag{7}$$

The calculation in equation[7] allows us to formulate the classification results in equation [8].

$$P(C_i|x_1, x_2, ..., x_n) = \left(\prod_{j=1}^{n} P(X_j|C_i) \right) \cdot \frac{P(C_i)}{P(x_1, x_2, ..., x_n)} \quad \forall \ \ 1 \leq i \leq k$$

The expression $P(x_1, x_2, ..., x_n)$ is constant for all classes j : (8)

$$P(C_i|x_1, x_2, ..., x_n) \propto \left(\prod_{j=1}^{n} P(X_j|C_i) \right) \cdot P(C_i) \quad \forall \ \ 1 \leq i \leq k$$

Variations of the Algorithm Several derivative forms of the basic Naive Bayes algorithm are also common. These variations usually implement different distributions on the feature probability $P(x_j|C_i)$, thus adding a degree of sophistication to this simplistic idea.

Gaussian A natural assumption when handling continuous data is that the values associated with each class are also continuous and distributed normally. De facto, the data is first segmented by the class c_i. Following, the mean and variance of each continuous variable x_j are calculated for each class c_i. Let μ_i and σ_i^2 be the mean and variance of feature set x associated with class C_i accordingly. The algorithm assumes a normal (Gaussian) distribution for the features, where as the parameters σ_i and μ_i are derived from a maximum likelihood estimation, i.e.,

$$P(x_j|C_i) = \frac{1}{\sqrt{2\pi\sigma^2 c_i}} \cdot e^{-\frac{(x_j - \mu c_i)^2}{2\sigma^2 c_i}} \tag{9}$$

Multinomial Here the features represent the frequencies of word occurrences which are distributed in a multinomial fashion such that p_i is the probability that an event i occurs in the multinomial $(p_1, p_2, ..., p_n)$. The feature set $\mathbf{x} = (x_1, x_2 m..., x_n)$ can than be visualized as a histogram, in which the frequency of event i (appearance of word i in a given data point) is represented by the hight of the appropriate column. This method is especially common when undertaking the Bag of Words approach. This will be further demonstrated in the Application part. Therefore, the likelihood of generating a histogram $\mathbf{x}$ is represented in [10].

$$P(X|C_k) = \frac{(\sum_i)!}{\prod_i x_i!} \cdot \prod_i p_{ki}^{x_i} \tag{10}$$

When extrapolated into log-space, [10] turns into [11], where $b = log\big(P(C_k)\big)$ and $\mathbf{w}_{ki} = log\big(p_{ki}\big)$. It is now evident that the classifier becomes linear in logarithmic-space.

$$\begin{aligned} log\big[P(X|C_k)\big] &\propto log\left(P(C_k)\prod_{i=1}^{n} p_{ki}^{x_i}\right) \\ &= log\Big(P(C_k)\Big) + \sum_{i=1}^{n} x_i \cdot log(p_{ki}) \\ &= b + \mathbf{w}_k \cdot \mathbf{x} \end{aligned} \tag{11}$$

Additionally, the product must be adjusted for the case when a certain event (appearance of word i) does not occur. Since non-occurrence will

be denoted as a frequency equal to zero, such values must be smoothed out to prevent them from nullifying the entire equation.

Bernoulli This variant assumes multivariate Bernoulli distributions for the features. Particularly, each feature can be represented with a boolean variable. Hence, the classifier requires the feature sets to be passed in the form of vectors composed of binary variables each representing a feature. The classification is based on the decision rule in [12], for class C_k and feature x_i. The probability of class C_k producing the feature x_i is denoted by p_{ki}. We can see that the non-occurrence problem from the Multinomial variant is being explicitly addressed here by penalizing for non-occurrence of a given feature x_i.

$$P\big(\mathbf{x}|C_k\big) = \prod p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)} \tag{12}$$

3.2.3 Support Vector Machines

Background The original Support Vector Machine (SVM) algorithm was formulated by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 while working at the institute of Control Sciences in Moscow. Despite the theory behind SVMs being far from new today, it remained mostly theoretical until quite recently. The state of technology at the time of its conception was far behind the theory, and it would take decades until computers would reach the computational abilities that could facilitate SVMs.

Implementation SVM aims at segmenting an input dataset to binary categories [[5]]. For example, positive and negative observations or analogously membership in or absence from a certain category. A common example is whether a given Email should be classified as Spam or not by the mail server. The rational behind SVMs could best be visualized by imposing the training observation on two dimensional Euclidean space. The algorithm strives to define a dividing line, which would create a separating threshold between the two groups. In two-dimensional space, the separator is demarcated using a simple line. Said line is then positioned in a manner, creating maximum separation between the two groups. This is achieved by drawing parallels through the closest positioned members

of both groups. The linear divider lays in the middle between this two parallels. The observations which are positioned on the parallels themselves, define the *curbs* of the so-called *dividing street*, and are dubbed Support Vectors. From them in turn, the algorithm derives its name. Any new observation passed to the trained classifier for classification, will be allocated to the appropriate group, as is defined by its position relative to the separating hyperplane.

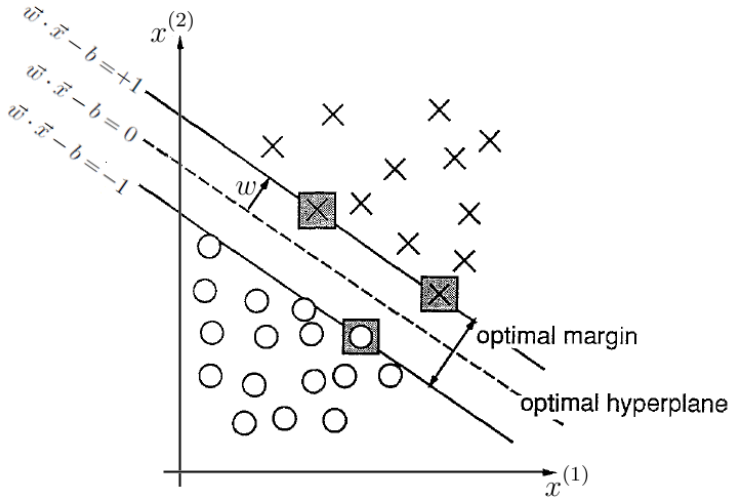


Figure 2: Euclidean Imposition of data. The support vectors, marked with grey squares, define the margin of largest separation between the two classes.

[6]

A dataset of size n as in [13] is to be used as a training set, in which y_i stands for the actual class of the observation i and can have one of either values $y_i \in [+1; -1]$. The values represent a positive and negative classification accordingly. A positive (+1) classification implies the presence of the sought-after class, $News$ in our case. The vector $(\mathbf{x}_i)$ is an array of size p, representing the features which compose each observation. In the 2-dimensional example, the 2 features of $(\mathbf{x}_i)$ can be graphically illustrated in euclidean space as the (x, y) coordinates $(\mathbf{x}_i) = [x_i^{(1)}, x_i^{(2)}]$.

$$\begin{gathered}(\mathbf{x}_1, y_1),..., (\vec{x}_n, y_n), \quad y \in \big[-1; +1\big] \\ \mathbf{x} = (x_{i1}, x_{i2}, ..., x_{ip})\end{gathered} \tag{13}$$

The optimal hyperplane segregates the two data classes, creating a maximally wide barrier between the closest observations of the opposing classes. This hyperplane is delineated in [14]. The vector $\vec{w}$ is the normal

vector [2] to the dividing hyperplane and the parameter b represents the normal vector's offset along the axis. The area between the 2 hyperplanes, which pass through the closest observations is referred to as the *margin* and is defined by the supporting vectors. These support vectors are illustrated in [15].

$$\begin{aligned} &\vec{w} \cdot \mathbf{x} - b = 0 \\ &\vec{w} = (w_1, ..., w_p) \end{aligned} \tag{14}$$

$$\vec{w} \cdot \mathbf{x} - b = \begin{cases} +1 \\ -1 \end{cases} \tag{15}$$

The distance between any point and the separating line [14] is shown in equation [16]. The distance between the 2 hyperplanes is equal to twice the distance between the support vectors and the separating line, hence $\frac{2}{||\vec{w}||}$. The optimization problem at hand is thus the maximization of this distance, in order to create a maximally distinct margin between the classes. Furthermore, no observation from the training data can be positioned inside the margin [17].

$$\begin{aligned} &\text{Dist. for point i :} && \text{Dist. for support vectors:} \\ &\frac{|\mathbf{x}_i \cdot \vec{w} + b|}{||\vec{w}||}, \quad \Rightarrow && \frac{w^T\mathbf{x} + b}{||\vec{w}||} = \frac{\pm 1}{||\vec{w}||} \end{aligned} \tag{16}$$

$$\begin{aligned} &\vec{w} \cdot \mathbf{x} - b \geq 1 \;\; \forall \;\; y_i = +1 \\ &\text{or} \\ &\vec{w} \cdot \mathbf{x} - b \leq 1 \;\; \forall \;\; y_i = -1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} &y_i(\vec{w} \cdot \mathbf{x}_i - b) \geq 1 \\ &\forall \; 1 \leq i \leq n \end{aligned} \tag{17}$$

Finally, the core optimization problem is to enlarge the margin as much as possible, subject to all the training data points being located on the correct side of the margin, given their actual class. As denoted in [18].

$$\begin{aligned} &max\left(\frac{2}{||w||}\right) \Rightarrow min\left(\frac{w^T w}{2}\right) \\ &subject\ to \;\; y_i(\mathbf{x}_i \cdot w + b) \geq 1 \end{aligned} \tag{18}$$

The optimal solution is array of values α, which are the learned weights to the x's and are equal zero for all but the support vectors ($y_i = \pm 1$). Incorporating the line [13] results in [19].

[2] A vector perpendicular to a given object, the diving hyperplane in this case

$$w = \sum_{i=1} \alpha_i y_i \mathbf{x}_i \quad \overset{(13)}{\Longrightarrow} \quad w \cdot \mathbf{x} + b = \mathbf{x} \cdot \sum_{i=1} \alpha_i y_i \mathbf{x}_i + b \tag{19}$$

It is now possible to derive a classification function based on the previous results. When classifying new data, unseen novel observations might fall inside the $[-1 : +1]$ margin, and are therefore classified according their sign (positive/negative) as in equation [20].

$$f(x) = \text{sgn}(\vec{w} \cdot \mathbf{x} + b) = \text{sgn}\left(\sum_{i=1} \alpha_i \underbrace{\overbrace{\mathbf{x}_i}^{\text{New Obs.}} \cdot \overbrace{\mathbf{x}}^{\text{Support Vectors}}}_{dot\ product} + b \right) \tag{20}$$

Notice that the results of the classification function depends solely on the dot product between the vector of the new point and the support vectors $(\mathbf{x} \cdot \mathbf{x}_i)$. It is hence possible to classify as shown in [21].

$$\text{class}(x) = \begin{cases} positive, & f(x) > 0 \\ negative, & f(x) < 0 \end{cases} \tag{21}$$

The basic SVM classification algorithm in itself seems to present a solution to a very limited spectrum of classification problems, that is two-dimensional, linearly-separable binary class classification problem. The following paragraph present solutions to each on of these difficulties. Notice that replacing the dot product from [20] can be generalized to any number $\mathbb{N}$ of dimensions, clearing the dimensionality restriction. The **Kernel Trick** resolves the linear-separability problem, by mapping to a higher dimension.

Kernel Trick A key element of SVMs statistical and computational power is the Kernel Trick [8]. Using the Kernel-Trick, the linearly dimensional training data x are extrapolated onto a higher plane using a mapping function $\Phi : \mathbf{x} \to \varphi(x)$, with the assumption that on the new plane a better linear separating hyper plane can be found. The mapping is carried out using a *kernel trick*, where by the internal dot product of the linear separator is replaced with a Kernel Function. This Function simulates the redistribution of the original Support Vectors in a richer space, barely incurring an additional computational cost in the process. The linear separator in the new space is not bilinear on the original plane,

thus the classification of new observation is also done using the Kernel Function. Whence the data is still not separable using a hyperplane in the new space, the dataset can be mapped to a higher still dimensional space. Mapping to a higher dimension is illustrated in equation [22].

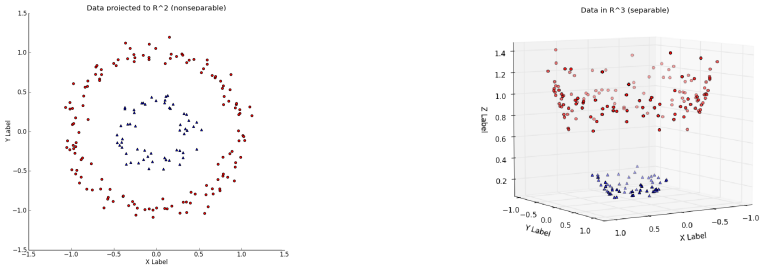


Figure 3: (Left) A dataset in , not linearly separable. (Right) The same dataset transformed: $\varphi(x_1, x_2) \Rightarrow [x_1, x_2, x_1^2 + x_2^2]$. *Source*:[9]

$$\begin{aligned} \Phi : \mathbf{x} &\rightarrow \varphi(\mathbf{x}) \\ K(\mathbf{x}) &\rightarrow \varphi(\mathbf{x}_j)^T \varphi(\mathbf{x}_j) \end{aligned} \tag{22}$$

$$\sum_{i=1} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b \rightarrow \sum_{i=1} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{23}$$

The function K [23] represents a *kernel function* and acts as a similarity measure which corresponds to the inner product in some higher feature space. As long as there exists a dimensional space of greater magnitude, in which the kernel function is the dot product of that higher dimensional space, such a kernel is in fact a dot product, and as such can be classified using the normal linear classifier without loss of generality.

Linear Kernel The kernel is in simply the dot product [24] and must not be mapped to a higher dimension. Therefore, the number of dimensions stays the same $\mathbb{N} \rightarrow \mathbb{N}$.

$$K(x_i, x_j) = x_i^T x_j \tag{24}$$

Polynomial Kernel Training data is extrapolated using a polynomial kernel function [25] into a higher dimensional space. The kernels represents the similarity of training samples in a feature space over polynomials of the original variables, allowing learning of non-linear models.

Additionally, the polynomial kernel adds an interpretive intuition by applying interaction variables (products of x_i and x_j).

$$K(\mathbf{x}_i, \mathbf{x}_j) = (r + \gamma \cdot \mathbf{x}_i^T \mathbf{x}_j)^d, \quad \forall \gamma > 0 \tag{25}$$

In [26] and [27] it is demonstrated that substituting the dot product with a kernel function results in the dot product on a higher dimension. Specifically [27] demonstrates the extrapolation of $d = 2$ space to $d = 6$ by employing a polynomial function.

$$\text{Let } K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \quad \text{such that } \mathbf{x} : \mathbb{R}^2 \Rightarrow \mathbb{R}^6 \tag{26}$$

$$\begin{aligned}
\overbrace{K}^{\mathbb{R}^2}(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2\big(x_{i1} x_{j1} + x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2} x_{j2}\big) + x_{i2}^2 x_{j2}^2 \\
&= \big[1 + x_{i1}^2 + \sqrt{2} x_{i1} x_{i2} + x_{i2}^2 + \sqrt{2} x_{i1} + \sqrt{2} x_{i2}\big]^T \cdot \\
&\qquad \big[\underbrace{1 + x_{j1}^2 + \sqrt{2} x_{j1} x_{j2} + x_{j2}^2 + \sqrt{2} x_{j1} + \sqrt{2} x_{j2}}_{\mathbb{R}^6}\big] \\
&= \varphi(x_i)^T \varphi(x_j)
\end{aligned}$$

such that :

$$\varphi(x) = \overbrace{\big[\, 1 + x_1^2 + \sqrt{2}(x_1 + x_1 x_2 + x_2) + x_2^2 \,\big]}^{\mathbb{R}^6} \tag{27}$$

Gaussian Radial Basis Kernel The Radial Basis Function (RBF) is a prevalent kernel in Machine Learning algorithms.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)} \begin{array}{l} \gamma = \frac{1}{2\sigma^2} \\ \downarrow \\ = e^{-\left(\gamma \cdot ||\mathbf{x}_i - \mathbf{x}_j||^2\right)}, \end{array} \tag{28}$$

$$\forall \gamma \ > 0$$

The RBF kernel ranges between zero (in the limit) and one (when $\mathbf{x}_i = \mathbf{x}_j$) and declines as the distance between the points grows smaller. RBF is likewise interpreted as a similarity measure, which is why it converges well to the kernel function idea. The feature space of the kernel is of infinite dimensionality.

Sigmoid Kernel The Sigmoid function, also known as the Hyperbolic Tangent function, bares semblance to the behavior of the logistic regression by creating an "S"-shaped curve.

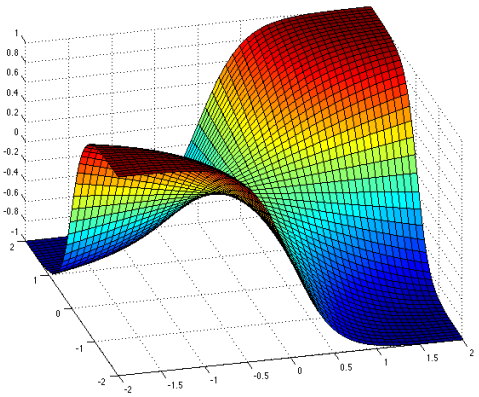


Figure 4: Sigmoid Kernel visualization in d = 3

This kernel originates from Neural Networks, another Machine Learning algorithm. The kernel, when implemented, produces an SVM equivalent to a two-layer Perceptron Neural Network.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \cdot \mathbf{x}_i^T \mathbf{x}_j + r) \tag{29}$$

Margin Rigidity or Slack Current research of SVM usage revolves around the concept of *Soft-Margins* [6]. In the case of not linearly separable data, which is usually the case, a *hinge loss* function is added to formula. Its purpose is to penalize for observations, which are on the 'wrong' side of the margin, that is between the 'sidewalk' and 'separator'. Using this method, observation are allowed to break the constraint present in the hard-margin case [18]. In other words, observations are permitted to cross to the "wrong side" of the margin, but they are being penalized for it, as demonstrated through the variable c in [30]. A trade-off between margin width, which represents the distinctness of each class, and the amount of misclassified observations can than be specified by the user.

$$max\ (c - y_i(\vec{w} \cdot \mathbf{x}_i - b)), \quad \forall\ c = \begin{cases} 0 & \text{if (8) holds true} \\ 1 & \text{else} \end{cases} \tag{30}$$

The optimization problem with the soft-margins approach is represented in Equation [31]. The variable λ controls the penalty magnitude for observations being on the wrong side of the margin. λ therefore, represents

the trade off between margin size and the observation being correctly classified. As the λ approaches zero, the problem converges into the hard margin scenario.

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(c - y_i(\vec{w} \cdot \mathbf{x}_i - b))\right] + \lambda||\vec{w}||^2 \tag{31}$$

Multiclass classification The classification model in itself is natively used for binary stratification, that is 2 classes only. However, as may be the case in many classification scenarios, data has more then 2 dimensions. For the purpose of Multiclass classification using SVMs, a process known as *class-reduction* is implemented [1]. This is done by either a 'one-vs-all' or 'one-vs-one' approach. With the former method, classifiers for each given class c are trained to differentiate it from the rest of the set. The testing data is then inputed into each of the classifiers and the highest scoring one (having the largest distance from the decision boundary) determines the class for each observation. This is also known as the *winner-takes-all* approach. Using the latter method consists of training classifiers for every combination of two classes out of all the classes. In the case where there are n classes, $\frac{n(n-1)}{2}$ classifiers will be trained as in [32] a max-wins voting strategy, whereby every observations is denoted to one of two classes by each classifier. Afterwards, the final classification is determined by a tally of the votes on each new data point.

$$\begin{aligned} \text{Classes:} \quad & \mathbf{c} = (c_1, c_2, c_3, c_4) \\ \text{Classifiers:} \quad & \mathbf{Cl.} = (Cl_{12}, Cl_{13}, Cl_{14}, Cl_{23}, Cl_{24}, Cl_{34}) \end{aligned} \tag{32}$$

Strengths and Weaknesses Among the advantages of SVMs one can mentions several factors. Firstly, the usage of different kernels - especially user-specified ones, allows for expert knowledge to be integrated into the classification problem. Secondly, the optimization space for the categorization problem is convex, thus making it easily and usually swiftly solvable with appropriate solving algorithms. And thirdly, the usage of soft-margins, or penalizing for error in classification makes the user more aware of the imminent over-fitting, which is the bane of classification problems. This awareness encourages the user to take a more cautious

approach and avoid over-specification. The disadvantages include but are not limited to the following. The choice of kernel is for the user to decide, which could add to the element of human error, especially with dilettante users. Secondly, the problem of Multiclass classification is not solved directly, but rather is adopted for, by creating a multitude of partial SVMs, making it not the most effective tool for such scenarios. Finally, the algorithm tends to be memory intensive during the training phase. Because a value needs to be calculated for each pair of n points, a matrix of size n^2 must be constructed. For example, when training with a dataset of size $n = 1,000$ - the kernel values matrix will be $n^2 = 1,000,000$ large. In optimized modules though, you can get away with only calculating half of the matrix, since its symmetrical along its diagonal.

3.2.4 Artificial Neural Networks

An Artificial Neural Network (ANN) [11] is a mathematical computational model, whose development was inspired by cognitional processes in a biological brain. AN Networks typically consist of numerous interconnected input and output information units. The form of connectivity between these units embodies the connection strength, in a fashion similar to how neurons are linked in the brain. ANNs are predominantly used in Cognitive Sciences and related applications, of which Artificial Intelligence is a prominent example. Common tasks for such systems are character and facial recognition, financial markets prediction and text mining among other uses.

Intuition A neural network is composed of linked processors called Perceprtons (combination of preception and neurons). Each Perceprton is capable of executing simplistic mathematical operations, however when combined into networks, they are capable of elaborate and sophisticated problem solving. Any given Perceptron receives inputs i with according weights w. These weighted inputs (products) are then summed up. When said sum exceeds a threshold value T, the output O is equal to one and zero otherwise, as in [33]. This operation mimics the operation of a biological neuron, which releases an electric signal when its agitation transcends a certain limit.

$$\langle i, w \rangle = \sum_j i_j w_j = \begin{cases} 1 & , \sum_j i_j w_j \geq T \\ 0 & , \sum_j i_j w_j < T \end{cases} = O \tag{33}$$

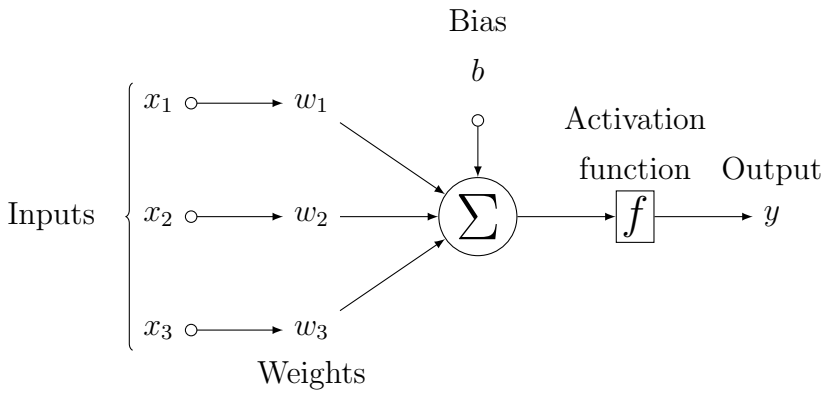
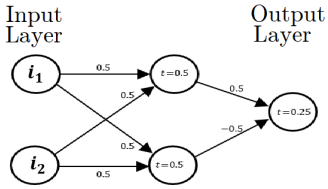


Figure 5: Structure of a Perceptron with 3 input nodes x_i and output y. Output 1 when threshold is surpassed and zero otherwise.

The organization of Perceptrons is crucial to the networks operation. The input weights are the ones, which determine the pattern to be recognized [3]. Therefore, the prime objective when constructing an ANN is determining proper values for the weights. The Perceptrons are used as logical gateway, which can carry out the "AND" and "OR" operations. A major obstacle in the early days of ANNs was the inability to construct a "XOR" (exclusive OR) logical gate, irregardless of the weights on the inputs. This difficulty was later overcome with the introduction of multi-layered ANN.

Multi-layered ANNs consist of several layers of Perceptrons. All layers except the first (input layer) and the last (output layer) receive their inputs from the lower level of Perceptrons and output into the higher layer, which in turn uses this output as its input. It was demonstrated that a "XOR" gateway could be constructed using a 3-layered network [6].



Input_1	Input_2	Output
0	0	0
1	0	1
0	1	1
1	1	0

Figure 6: A three layer Network, creating a XOR gateway.

Classification The type of network used most commonly for classification is a Multilayer Perceptron Network (MLP). This falls into the category of feedforward ANNs. A feedforward network is one, which prohibits cycles between its nodes. Thus each layer only outputs information to the next layer, and not to the ones which came before. Typically, a non-linear sigmoid function is used for activation rather than a binary step function as described in [5]. A sigmoid function [34] (often called a Soft-Step) or another continuous function are usually used, since ultimately the algorithm will have to calculate gradients to maximize or minimize an optimization function. Therefore this function should be continuous, differentiable and Real-valued.

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{34}$$

The network is than optimized through the process of **Backpropogation** of errors. During the backwards propagation the initial weights of all the axons (input nodes of a neuron) weights are determined at random. Subsequently, the optimization problem is set by measuring the aggregate error terms in classification (misclassification). This error is calculated as a function of all afore mentioned weights. Next, the aggregate error function must to be minimized by observing the contribution of each weight to the error and reducing it. Finally, the gradient (derivative in vector space) is derived and the weights are adjusted by Δw respectively as shown in [7]. This process is repeated, pending we no longer get improvement (lower error terms) or the error reduction is lower than a predetermined threshold.

Stochastic Gradient Descent In the basic variant of ANN's, an objective function is optimized by tuning the weights in a manner which reduces the error term gradually with each iteration. In every repetition the weights are adjusted by Δw according to the gradient for each weight over all the observations. Every repetition requires the calculation of the function's derivatives with respect to every data point and their summation within every iteration. This base method is also referred to as Batch Gradient Descent (BGD). This means, that in order to tune a weight w_j we must go through all its derivatives in the gradient and sum them up. Applying this technique requires us to derive the function $m \cdot n$

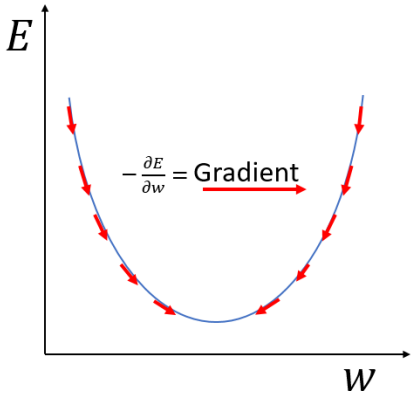


Figure 7: A descent down the error function, taking gradients of w to calculate the steps.

times in each step of the algorithm, with m and n being the number of data points and the set size of features respectively. The problem with this procedure is its computational costliness which increases as the data set grows larger. Therefore, the standard BGD approach scales poorly. We can hence modify the algorithm using a method called Stochastic Gradient Descent (SGD) to address scalability issues.

To carry out SGD we first shuffle the training dataset to avoid any inherit order or structure present in the dataset. Subsequently, we derive each of the weight modification and adjust after each derivation before continuing to the next data point. Finally, we repeat this process for each data point. Noteworthy is that we are now adjusting after each derivation instead for all derivations of a data point. Therefore, the descent tempo is now $(m \cdot n)$ $\frac{\text{improvements}}{\text{algorithm round}}$ as compared to the tempo (n) $\frac{\text{improvements}}{\text{algorithm round}}$ of BGD. Hence, SGD progresses m times as fast as BGD. Additionally, the loop over all the data points can be repeated more than once for better results.

Since each adjustment is being made for a single data point at a time, it is much more likely that the adjustment is not actually an improvement and possibly takes us further away (or not nearer) from the global minimum. Nonetheless, the algorithm does converge to the extremum and faster at that, then the BGD approach. Figure [8] illustrates graphically the convergence of both SGD and BGD. The stochastic approach is clearly recognizable with its "squiggly" line. One notable problem with this algorithm is that it might continuously overshoot the local minimum due

to its high variance.

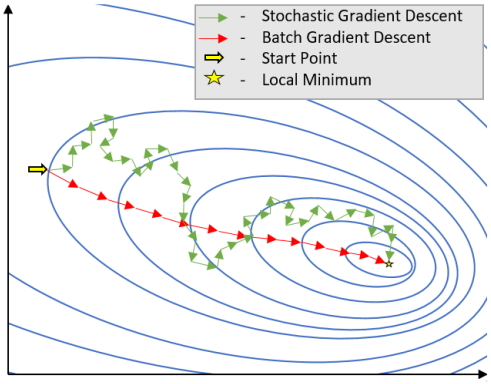


Figure 8: A comparison of the Batch and Stochastic Gradient Descent Approaches

In the example above, an SGD approach is described, in which we fit the weights for a single observation at a time. Another possible modification is to take minute batches of observations and fitting the algorithm on them instead of the entire dataset. This approach is also known as Mini-Batch Gradient Descent and combines the best of both approaches, since it simultaneously addresses the overshooting complication of SGD and is speedier than BGD. Thus, by adjusting the batch size one can control the trade-off between speed and accuracy, where a larger batch results in a more accurate step, but the computation time will also increase.

The mechanism of Stochastic Gradient Descent, at its core, is extremely inaccurate on a single step basis. This is due to the probability of improvement being much lower when compared to Batch Gradient Descent. However, this is compensated for with a more rapid improvement rate, making the accumulated advancement catch up and overtake that of the normal scenario. The heightened convergence pace is what makes this approach more suitable for classification with extensive training data sets.

Adaptive Moment Estimation ADAM is another approach often used in tandem with SGD. Using ADAM we calculate the first (mean) and second (variance) moment of the gradients and modify the step size for each parameter (feature) in accordance with its frequency. Therefore, the update size is usually more significant for infrequent parameters and more fine for frequently occurring parameters.

3.2.5 Decision Trees and Random Forests

Another popular type of classifiers stems from the idea of decision trees [[15]]. That is, a flow chart consisting of decision intersections (called nodes) flows from a starting root (the tree stem). The data is further segmented in each following junction until a state is achieved, in which ideally all segments are "pure", containing observation belonging to a single class.

ID3 Algorithm This procedure of Decision Tree induction is known as the ID3 Algorithm [[14]]. More formally, the design functions as follows. We start, as is common in classification problems, with a data training dataset. Each data point (observation) in the set has an assortment of attributes (features) and is assigned to a certain class. At each stage of the tree construction, including the inception stage, a node is created. This node receives as input a group of observations which belong to 2 or more classes. The node than selects a feature of the data and divides the data according to this feature. The feature selection consideration will be explained subsequently. For each possible value of the previously selected feature, a node is created. If the data inside a node belongs to a single class, it will not be segmented any further and the node will be denoted as a *leaf*. If, on the other hand, the new node contains data points from different classes, we repeat the process of feature selection and segmentation accordingly. Finally, we want to reach a scenario in which, all leaves of the tree contain observations belonging to a single class each. The contents of such leaves may also referred to as *pure subsets*.

Entropy The priorly mentioned decision rule for feature selection, upon which the data will be split in a given node, can be determined by 2 approaches. Both have at heart the same principle, this being the amount of certainty gained from each split. Hence the goal is the maximization of certainty, that after a given split a data point belongs to a distinct class. For example, a feature would be considered a bad feature for splitting, if at a given node it splits the data in a manner where all subsets have a relatively similar probabilities of belonging to the different classes. In such an example, no progress was made in the direction of segmenting

the data completely. The closer the separation of the data brings us to pure subsets, the higher its ranking as decision rule at a given node will be. This means, that the lower the entropy values are, the purer the resulting subsets will be.

One measure which helps quantify the quality of this segmentation rule is **Entropy**. The calculation of Entropy is demonstrated in [equation 35], with S being the subset passed to the node as input, $i \in \{1 : c\}$ denoting the different classes and p_i the percentage of data point corresponding to class i.

$$H(S) = \sum_{i=1}^{c} -p_i \cdot log_2(p_i) \tag{35}$$

Gini An alternative metric to entropy is the Gini Impurity coefficient. The coefficient denominates the probability of a randomly selected observation to be incorrectly labeled, if the label was set at random according to the actual distribution of labels in the dataset. We compute Gini by summing the probabilities of a given item belonging to a certain class i multiplied by the probability of misclassification. The Gini impurity is demonstrated in [36], with i denoting a given class, out of c classes.

$$\begin{aligned} I_G(p) &= \sum_{i=1}^{c} p_i(1 - p_i) = \sum_{i=1}^{c} (p_i - p_i^2) \\ &= \sum_{i=1}^{c} p_i - \sum_{i=1}^{c} p_i^2 = 1 - \sum_{i=1}^{c} p_i^2 \\ &= \sum_{i \neq k} p_i p_k \end{aligned} \tag{36}$$

Information Gain Entropy allows us to further calculate the "Information Gain" from each split. This gain is calculated in [37], with S being the set of input examples, v a possible value of the attribute A and S_v the subset of S in which the A attribute of the examples is equal to v. The gain is therefore a weighted average of the Entropies, with respect to the prevalence of a given value v in each subset. Thus Information Gain also assign important to the progress made from a new node. So a node which classifies purely few items might be outweighed by one which does not split into completely pure subsets, but splits correctly much more items.

$$Gain(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{S} H(S_v) \tag{37}$$

Pruning After the tree is built, it undergoes the process of *pruning*. That is branches that add very little information to the classification are removed. The algorithm is considered efficient and fast since after pruning the amount of features actually used from a dataset is small. In addition, the algorithm is proficient in filtering noise and selecting the best features. Better data attributes are bound to have higher Information Gain values are therefore more likely to be used in decision nodes.

Random Forest This algorithm builds a boot-strap system on the concept of Decision Trees. The algorithm [4] builds$K$ different Decision Trees. Each Decision Tree T is built using a randomly picked subset S of the complete dataset. Each such tree is trained using the previously mentioned ID3 algorithm. However each tree does not use all the datasets attributes D but rather ones selected at random $d << D$. The trees are not pruned.

Thus, each of the K trees is only suitable for a random subset S of the dataset and examining only a subset d of attributes. Each tree is not pruned, therefore it classifies its training data perfectly. When classifying novel data, the observation gets classified using all K tree, and the most commonly appearing classification decides the class. The process is therefore a voting classification using a *Forest* of Decision Trees.

4 Application

4.1 Information Procurement

The main theme of the collected data would concentrate around the topic of Internet sales platforms also known as E-Commerce. Several aspects make the topic beneficial for this research. Firstly, the very nature of E-Commerce. By nature of E-Commerce Platforms, I refer to the fact, that such companies are almost exclusively web-based. It is therefore probable that most of the company's marketing efforts as well as overall news relating to such a company, would circulate first and foremost in the Internet. Secondly, having all company-relevant news attainable foremost from the web, means that such news would seep to social media faster in comparison to news, which are usually covered initially by traditional media such as television and Newspapers.

Search Terms

The data was collected in the form of relevant Tweets from the Twitter Stream API. A Tweet would be considered relevant if it contained a search parameter related contextually to E-Commerce. The initial efforts were concentrated around the web-store Amazon. Amazon appears to the most fruitful search parameter in terms of the quantity of Tweets relating to it. Additional search words that were tested were **Alibaba, Zalando** and **Groupon**. The widespread mention of Amazon in Tweets is however somewhat over-inflated due to the extensive use of Amazon gift cards. Amazon gift-cards have become prominent due their variety of uses. A few examples of common practices involving Amazon gift cards are rewarding users for services, such as polls and questionnaires, enticing people to take part in events or groups, and being offered as general rewards in competitions and games. The plethora of uses, facilitates Amazon gift cards to be viewed as a sort of pseudo-currency in the Internet. In turn this means, that Amazon could be mentioned in a Tweet, despite the context only indicating the Gift-card and being completely unassociated to the E-Commerce platform whatsoever.

Collecting the Data

The gathering of Tweets was executed using a program written by me in the Python programming language. The main module being used in the program was a Twitter Streaming API called Tweepy. Tweepy is an open source interface, which allows communicating with the Twitters servers and sending queries requesting specific information from Twitter's databases. The interface allows for two main type of queries, Rest and Streaming. The former allows looking up information posted on Twitter in the past whereas the latter, as the name implies connects to an active data stream containing a narrowed down flow of Tweets being actively published by Twitter users. Both types of API's are being offered for free to a certain extent, whereas almost unrestricted versions of the same API are offered as a proprietary fee-based product of Twitter. The free version of the REST API is restricted to only looking up Tweets posted in the last two to three weeks. And the gratis version of the Streaming API is restricted to a fire-hose narrowed down to about 15% of the total Bandwidth of all current Tweets. The Tweets from the Twitter servers come in form of JSON strings, which allows for embedding other JSON objects in them, which allows for multi-level storage of Tweet properties. For example, one of the JSON objects integrated in each Tweet JSON object is the USER object for the Tweet-poster. The USER object in turn contains all data publicly available in Twitter about a Twitter account such as, location, date of registration, homepage etc. An additional object of interest is the ENTITIES JSON object, which contains all outside references from the Tweet's text such as, URLs, Multimedia, References to other Tweets or other users. This structure greatly eases the construction an analysis of a Tweet and its features, since most of the necessary data is available from the Tweet self and no further queries about the Tweets and its posting-user are necessary.

elaborate about stages of data collection:

1. Initial data: full of duplicates
2. Filtered data - fewer duplicates and Spam, but still rather very one-subject-centered. Leads to overfitting when classifying
3. still untried - collect up to x (400) Tweets per day, for a duration of 2 weeks

Data cleansing

It was observed that numerous Tweets were being posted more than once and in several occurrences even hundreds of times. These duplicates were being primarily posted by bots, as was evident from a short observation of user profiles belonging the Tweets original posters. Evidently, additional effort was being made by the programmers of the bots to try and mask them by slightly altering the content of the Tweets, or the user account. This was usually done by changing or adding characters to the text, which carry no lingual significance in themselves. Moreover, in furtherance of increasing the bots' credibility as an actual people (Fig. 1), often times entire nets of such bot could be observed, wherein the bots would maintain friendship and following connections among themselves. This in turn, further adding to each of them having a multiplicity of friends and followers contributing to their veil of disguise as real human users of Tweeter. Upon closer observation such accounts reveal their true essence, since most of the content propagated by them is commercial in nature and is repeated verbatim time and again across many of the related accounts *followers* and *friends*, it would be safe to assume that no actual people are behind them.

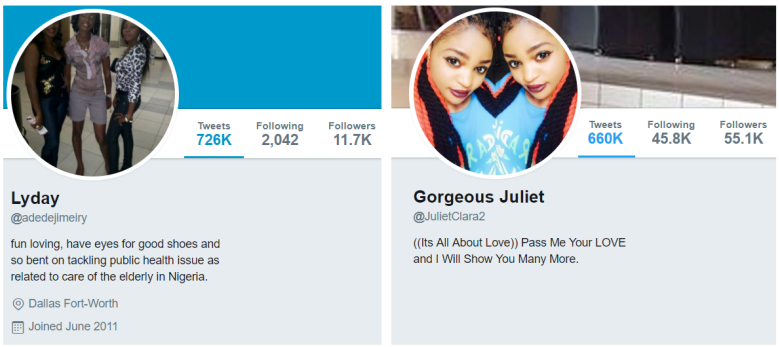


Figure 9: Twitter accounts, which present themselves as actual people

Several precautions were undertaken to try and filter out such bots. A passive precaution which was implemented, was blacklisting users, which had priorly been observed posting Tweets, which were verbatim copies of other Tweets within the same query. The suspected users were added to a suspect database and content originating from them was ignored in future queries. Another action made with the same purpose in mind was a retrospective cleanup of the collected Tweets, based on their content similarity. After closing a collection query, a maximum similarity measure for each Tweet in relation to all other recorded Tweets was calculated using a simple . Following, Tweets which were found to have a maximal similarity score to other previously captured Tweets higher than a predetermined threshold were classified as non-unique copies and were disregarded. As a measure of similarity, the Sørensen-Dice coefficient[17] (Fig. 2) was implemented using the *SequenceMatcher.ratio()* function from the difflib Python module also try Levenshtein . A round of cleanup using this procedure would usually reduce a data set by from one quarter and up to one half of its original size.

$$QS_{XY} = \frac{2|X \cap Y|}{|X| + |Y|}, \qquad QS_{XY} \in [0:1]$$

* $|X|$ and $|Y|$ are the numbers of elements in given Tweets X and Y accordingly

** QS ranges from 0 (completely different) to 1 (identical)

Figure 10: Sørensen-Dice coefficient

4.2 Building the Datasets

Once a list of labeled Tweets is obtained, the next stage is constructing a feature-set to be later passed on as input for training an ML Classifier. The features contained if the feature sets describe certain aspects and characteristics of a Tweet and its owner. Two main approaches to feature sets are found in the literature, *Word-Based* also often called *Bag-of-Words* as can be observed in **insert citation** and *Descriptive* **insert citation**.

4.2.1 Word-Based Features

The former approach simply converts the entire text corpus to a frequency charts of all the words contained within. Words are then selected to act as features in incoming data, which is to be classified. The features are hence a variable list (usually of several thousands in length), where each variable is a boolean representation, indicating the presence or absence of a certain word. Usually the words undergo preprocessing as is common in Natural Language Processing prior to being used as features. The corpora are segmented to lists of words, often omitting articles, proposition and punctuation. Such grammatical structures are critical in human speech and writing ind order to convey ones meaning clearly and explicitly, however for the purposes of more ambiguous classification as in our case, such nuances are avoided for the sake of simplicity. Words are then *stemmed* or *lemmatized*, meaning their are reverted to their grammatical stem - dropping all prefixes and suffixes. This eases the enumeration of words, since it is preferable that the same words in different inclinations would be counted as the same. For example, the word pair *eating* and *ate* would be reverted to their stem *eat*, as well as *apple* and *apples* would be considered as one and the same. Finally, words would be assigned their part of speech (noun, verb, adjective ..) and could be either ignored or incorporated into the feature set, according to the conceived importance of a given part of speech.

4.2.2 Descriptive Features

The latter approach mentioned is based of more generalized view of the Tweet, instead of concentrating on the actual textual content. Descrip-

tive features are aimed at describing the Tweet implicit properties, such as attitude, sentiment, seriousness and trustworthiness. These features detect presence of different symbols, their frequency and consecutiveness. Additionally, unlike the Web-Based approach, non-textual objects such as multimedia, links and mentions of other users and Tweets are also taken into account. Furthermore, features of a Tweets owner are included alongside. Since Tweeters API provides a complete user profile incorporated inside the Tweet data object itself, constructing features describing the user is done simultaneously to features describing the content of the Tweet itself. This approach might be viewed as an *indirect* one, since less obvious properties of the Tweet are used to characterize it.

Descriptive features could be segregated into three distinctive groups. The first will be referred to as text-based features. As the name suggests, the features will mostly denote the presence or lack of specific characters such as emoticons and signs in the Tweets text. Whether a Tweet contains combinations or sequences of certain symbols as well as ratios defining the text also befit this category.

The second tier of features describe any special *Entities* (Tweeter's nomenclature) contained within a Tweet. *Entities* refer to non-textual contents of a Tweet such as media (in form of pictures, sound or videos), URL's linking to external websites, Mentions or ReTweets (Referring to other Tweets or to Tweeter user profiles) and finally Hashtags. A word or phrase preceded by the Hashtag symbol # indicate the association of web content (such as a Tweet or other micro-blogging post) to a specific theme such as an event, news, gossip or any other tidbit [19]. Hashtags are used primarily to simplify looking up Tweets or other social media content by technically associating them with the *hashtagged* topic.

The third tier - subject

5 Results

Place holder

- NOTE: Random Forest is not suitable for Bag-Of-Words because of sparse data

6 Genaral Use Cases

placeholder

References

[1] Aly, M. Survey on multiclass classification methods. vol. 19.

[2] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit.* O'Reilly Media, Inc., 2009.

[3] Bishop, C. M. *Neural networks for pattern recognition.* Oxford university press, 1995.

[4] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and regression trees.* CRC press, 1984.

[5] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2*, 2 (1998), 121–167.

[6] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[7] Demchenko, Y., Grosso, P., De Laat, C., and Membrey, P. Addressing big data issues in scientific data infrastructure. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on* (2013), IEEE, pp. pp. 48–55.

[8] Guyon, I., Boser, B., and Vapnik, V. Automatic capacity tuning of very large vc-dimension classifiers. 147–155.

[9] Kim, E. So, what is a kernel anyway? `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`.

[10] Klier, M., and Heinrich, B. Datenqualität als erfolgsfaktor im business analytics. *Controlling 28*, 8-9 (2016), pp. 488–494.

[11] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[12] Mitchell, T. *Machine Learning.* McGraw-Hill, 1997.

[13] 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., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of*

Machine Learning Research 12 (2011), 2825–2830.

[14] Quinlan, J. R. Induction of decision trees. *Machine learning 1*, 1 (1986), 81–106.

[15] Quinlan, J. R. *C4. 5: programs for machine learning*. Elsevier, 2014.

[16] Rish, I. An empirical study of the naive bayes classifier. 41–46.

[17] Sørensen, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr. 5* (1948), pp. 1–34.

[18] Streaming APIs. Twitter Developer Documentation. `https://dev.twitter.com/streaming/overview`. Accessed: July 13. 2017.

[19] "tweet". *https://www.merriam-webster.com*. Merriam-Webster, Accessed: June 20. 2017.