Dimitris Katsavelis

“DATA MINING WITH THE USE OF MACHINE LEARNING ALGORITHMS ON DATASET THAT CONTAINS INVENTORY-ACTIVE PRODUCTS”

Master of Science
Academic Year: 2020-2021

Scientific Supervision by
Professor Tsadiras Athanasios
March, 2021
# Table of contents

Table of contents .......................................................................................................................... 2  
Acknowledgements ....................................................................................................................... 4 
Abstract .......................................................................................................................................... 5 
List of Figures .................................................................................................................................. 6 
List of Tables .................................................................................................................................... 6 
List of Graphs .................................................................................................................................... 6 
CHAPTER 1 – Introduction .................................................................................................................. 7  
1.1 Introduction to knowledge discovery in supply chain management .......................................... 7 
1.2 Purpose of the thesis ................................................................................................................... 7 
1.3 Outline of the thesis structure ...................................................................................................... 7 
CHAPTER 2 – Literature review ......................................................................................................... 8  
2.1 Data mining in Supply Chain Management .................................................................................. 8 
2.2 Machine learning and data mining ................................................................................................ 10 
2.3 Description of machine learning algorithms ................................................................................ 12  
2.3.1 Decision tree algorithms ......................................................................................................... 12  
2.3.1.1 J-48 (or C4.5) ...................................................................................................................... 13 
2.3.1.2 Random Tree ..................................................................................................................... 14 
2.3.1.3 Random Forest .................................................................................................................. 14 
2.3.1.4 Simple Cart ........................................................................................................................ 15 
2.3.1.5 Best First Tree ................................................................................................................... 16 
2.3.1.6 Logistic Model Tree ............................................................................................................ 16 
2.3.1.7 Decision Stump .................................................................................................................. 17 
2.3.1.8 Hoeffding Tree .................................................................................................................. 18 
2.3.2 Bayesian algorithms ................................................................................................................ 19  
2.3.2.1 Naïve Bayes ....................................................................................................................... 20 
2.3.2.2 Naïve Bayes Updateable .................................................................................................... 21 
2.3.2.3 Bayesian Network ............................................................................................................. 22 
2.3.2.4 Naïve Bayes Multinomial Text .......................................................................................... 23 
2.3.3 Instance-based algorithms ......................................................................................................... 24  
2.3.3.1 K-Nearest Neighbors (K-NN) ............................................................................................ 24 
2.3.3.2 K-Star (K*) ....................................................................................................................... 25 
2.3.4 Neural Network ....................................................................................................................... 27 

2
Aristotle University
MSc in Logistics & Supply Chain Management

2.3.4.1 Multilayer Perceptron................................................................. 27
2.4 Confusion Matrices ........................................................................ 28
2.5 Evaluation measurements ............................................................... 29
2.6 Random Under-sampling ................................................................. 30
2.7 Cross-validation ........................................................................... 31
2.8 WEKA Tool .................................................................................. 33

CHAPTER 3 – Methodology .................................................................. 38
3.1 General description of the methodology .......................................... 38
3.2 Exploratory data analysis and visualization ..................................... 39
3.3 Algorithms evaluation on WEKA ..................................................... 46
  3.3.1 Presentation of confusion matrices ............................................. 46
    3.3.1.1 Tree-based algorithms ......................................................... 46
    3.3.1.2 Instance-based algorithms ............................................... 47
    3.3.1.3 Neural Network ................................................................. 48
  3.3.2 Results of executed algorithms with the use of Cross-validation .... 48
    3.3.2.1 Tree-based algorithms ....................................................... 49
    3.3.2.2 Bayesian algorithms ......................................................... 49
    3.3.2.3 Instance-based algorithms ................................................. 50
    3.3.2.4 Neural Network ................................................................. 50

CHAPTER 4 – Results evaluation and conclusion .................................... 51
4.1 Evaluation of results ..................................................................... 51
4.2 Conclusion .................................................................................... 52

Bibliography ...................................................................................... 53

APPENDIX .......................................................................................... 53
Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Professor Athanasios Tsadiras, for the constant communication. Every time I needed help and guidance was there with helpful advice and was answering all my questions. The comments on my thesis were precise and the feedback he provided helped me elevate my work.

Most importantly, I would like to show my appreciation to my parents. Their support through this difficult process was the key to help me successfully complete my thesis and my master’s degree, also. Million thanks aren’t enough to express my gratitude and love to you.

Furthermore, I would like to extend my gratitude to all of the professors of the master’s degree for the knowledge, the experience and the helpful advice they provide throw this difficult process and for their help to provide me an internship that turn out to be my first job in this field. Last but not least, I would like to thank my colleagues for the support and the memories we made.
Abstract

In this dissertation, the scientific field of supply chain management is introduced as well as the field of data mining is presented. The machine learning algorithms are used on a dataset that contains inventory-active products, to achieve knowledge discovery. More specifically, the machine learning algorithms are used for predicting if a product went on backorder. It is important to mention that the statistical techniques of “Under-sampling” and “Cross-validation” are used to improve the results. All these results that are produced from the executed algorithms are presented and evaluated, along with confusion matrices and other essential evaluation measurements.
List of Figures

Figure 1 Representation of Decision tree ................................................................. 13
Figure 2 Algorithm for Hoeffding tree ................................................................. 19
Figure 3 A multilayer feed-forward neural network ............................................ 27
Figure 4 Representation of Confusion Matrix ....................................................... 29
Figure 5 Representation of 10-fold cross validation .............................................. 33
Figure 6 "Waikato Environment for Knowledge Analysis" provides 5 applications ........ 33
Figure 7 Weka representation of "Preprocess" tab ............................................... 35
Figure 8 "Classify" tab choosing machine learning algorithms ............................. 36
Figure 9 "Classify" tab overview ............................................................................ 36
Figure 10 Representation of dataset before and after application of random undersampling ........ 41
Figure 11 Dataset visualization regarding the distribution of the numeric variables .......... 42

List of Tables

Table 1 Dataset explanation with definitions and data type of each variable ............... 39
Table 2 Representation of dataset in arff format .................................................. 40
Table 3 Confusion Matrices of Tree-based algorithms ...................................... 46
Table 4 Confusion Matrices of Bayesian algorithms ........................................... 47
Table 5 Confusion Matrices of Instance-based algorithms ................................. 47
Table 6 Confusion Matrix of Neural Network ....................................................... 48
Table 7 Results after the execution of tree-based algorithms with the use of Cross Validation 49
Table 8 Results after the execution of Bayesian algorithms with the use of Cross-Validation ......... 49
Table 9 Results after the execution of Bayesian algorithms with the use of Cross Validation .................. 50
Table 10 Results after the execution of Neural Network with the use of Cross Validation .......................... 50
Table 11 Ranking of algorithms based on F-measure and AUC-ROC .................... 51

List of Graphs

Graph 1 Random undersampling graph .................................................................. 31
Graph 2 Representation of methodology ................................................................ 38
Graph 3 Correlation matrix with the use of WEKA tool ........................................... 44
CHAPTER 1 – Introduction

1.1 Introduction to knowledge discovery in supply chain management

Data mining covers all suspects in supply chain management. The use of data mining in supply chain has become even bigger over the years. More and more supply chains rely on the information that data mining applications provide to understand the results and make predictions or understand where they are struggling and change their perspective. It is become critical for a supply chain manager to understand and know how to use data mining applications to provide results that could help him decide.

1.2 Purpose of the thesis

The purpose of this thesis is to understand how data mining applications work and how they can help with supply chain problems and optimize the decision-making process for a supply chain manager. We are going to implement the methods we learn in the university classes and the data mining methods and combine them in a real situation problem to create a solution. It is a new different chapter for us, because there is no knowledge background in the field of data mining but in order to stay in touch with the market, we need to learn the data mining process that has become a crucial part in the business.

1.3 Outline of the thesis structure

In this dissertation, there will be an in-depth explanation of the ways that data mining can be used along with Supply Chain Management. Moreover, the definitions of machine learning and data mining will be presented, along with all the machine learning algorithms that were used and executed in this dissertation. Presented in the following categories:

1. Decision Tree (J-48, Random Tree, Random Forest, BFTree, Simple CART, LMT, Decision Stump, Hoeffding Tree)
2. Bayesian (Naïve Bayes, Naïve Bayes Updateable, Bayesian Network, Naïve Bayes Multinomial Text)
3. Instance-based (K-Nearest Neighbors, K-Star)
4. Neural Network (Multilayer Perceptron)
In Chapter 3, there will be an analysis of the methodology that was used to realize this project. More specifically, we will go through a brief description of the methodology and the dataset that was selected, and then we will present the results.

In Chapter 4, there will be an evaluation of results and we will present you the conclusion, with all the useful information that is discovered from the project.

CHAPTER 2 – Literature review

2.1 Data mining in Supply Chain Management

The use of data mining is growing in every business that wants good and comprehensive analyses of its data. So, it is inevitable that supply chain management will include data mining tools in their business. As the years passed the use of data mining in supply chain management was growing rapidly and by now, we can find it in almost every aspect of the supply chain. But to understand where we use data mining in supply chain, we first have to give a definition.

“DM is another subdomain of artificial intelligence and can be defined as a process that aims to generate knowledge from data and presents findings comprehensively to the user. Generating knowledge in the context of DM can be translated to the discovering of new and non-trivial patterns, relations and trends in data useful to the user. DM as a process involves, in essence, the collection and selection of data, the pre-processing of data, data analysis itself including the visualization of results, interpretation of findings, and the application of knowledge.” (Schuh et al., 2019)

As mentioned earlier data mining is now used probably in every aspect inside the supply chain but the more characteristics are the ones below:

- Forecasting
- Product development
- Inventory management
- Supplier selection
- Logistics
- Risk analyses
Forecasting is a field that plays a significant role inside the supply chain and because of that is a field that has been studied quite comprehensively. Data mining can provide a more accurate forecasting using certain tools. Quite close to forecasting is prediction. The difference between those two is that forecast can provide a variety of possible outcomes, but prediction is the statement that one outcome is the correct one. Data mining tools can provide also predictions using certain procedures that we are going to mention in this paper. Finally, data mining tools can provide supply chain managers the advantage of identifying and manage risk inside the supply chain.

Product development is also a key area in many businesses and many supply chains. Many brands spend a lot of money in research and development to develop their product and their sales or their place in the market. Before a product was created a lot of companies perform marketing studies. In these studies, they use data mining techniques to get their results and understand what the customer wants. There is also a research that simultaneous management of supply and demand can foster mass customization in real-time. (Tien, 2006)

Inventory management is also a well-studied field in the supply chain literature. There are also a lot of papers that deal with inventory management using data mining tools and techniques. Most of them propose an automated inventory system that controlled from the results that big data analytics could provide. There are also a lot of research in this part of the supply chain that provide answers using classification modeling as the analyses of a dataset that we provide in the next chapters of this thesis. Finally, many papers deal with the inventory problem using simulation because of the presence of variance in inventory. (Tiwari et al., 2017)

Supplier selection is, as the others before, a common studied in the supply chain management. The difference is that in this field the common tool to use in the analyses is data mining. The try to combine data mining techniques with the knowledge of supply chain management to achieve better results. The use is not limited only in the selection of the best supplier but also provide feedback in supplier performance (Wang et al., 2016). Some papers using specific multiple criteria aspects compare to others that utilizing fuzzy and related
approaches. Finally, in supplier selection we see all the gamut of data mining techniques due to plenty papers that provide different analyses.

Supply chains often outsource some of the operations as it is providing better service and it is more efficient. This provide many advantages such as focusing on the product but also increase the risk inside the supply chain. The risk of outsourcing in the supply chain is a well-studied field. There are some papers that provide analyses using data mining techniques but because of the variation in the data simulation may be a better method to analyze it.

Logistics is also a key field in supply chain management. In logistics and 3pl companies the use of big data and the results that can provide throw a comprehensive analysis are critical to their business success. Studies show that most successful 3pl companies invest a lot to big data analyses and data mining techniques to ensure visibility from the start of the process till the end and create flexible supply chain strategies. (Burnson, 2013)

As we can see now almost all of the functions inside a supply chain are analyzed and investigate with data mining tools and techniques to improve their production, cost and other key areas and provide better results combined with the knowledge the supply chain management can provide. Big data analyses is key to 3pl business and logistics companies around the world. More and more companies nowadays understand the significant of data mining and the role that can play in their business and invest a lot of effort and money in providing comprehensive analyses.

2.2 Machine learning and data mining

According to (Witten & Frank, n.d.), the cooperation of computing and communication Has created a society that works with information. As they state “Yet most of the information is in its raw form: data. If data is characterized as recorded facts, then information is the set of patterns, or expectations, that underlie the data.”.

There is a vast quantity of information that exists inside the databases—information that is possibly important but has not yet been identified. The task of data science is to discover it.

At this point, to give a better understanding to the reader there will be a brief explanation of the definitions of machine learning and data mining.
Arthur Samuel supports that “machine learning is the field of study that gives computers the ability to learn without being explicitly programmed”.

(Witten & Frank, n.d.) claim that machine learning supplies the technical foundation of data mining. It is used to take out information from the raw data, this information ease demonstrated in an understandable form and can be used for numerous reasons. The input on machine learning algorithms is a collection of registrations. These registrations are either classified, associated, or clustered. Each registration is a single, solitary case of the idea to be learned. Also, each registration is identified by the values of a collection of prearranged columns. The registrations can be either arithmetic (continuous) or nominal (discreet). The representation of it in a table can be shown as the registrations to be the rows and the attributes to be the columns.

This dissertation is using a category of machine learning called “Supervised Learning” and more specifically “Binary Classification” because the problem we are facing is to predict if the products went on backorder or not.

According to (Kolosova & Berestizhevsky, 2021), at times supervised learning is analogized with the work of the statisticians. The term “supervision” indicates that the response variable (or labels) are given, and the word “learning” specifies any technique or process that can be evaluated for its result. When the labels are accessible, there is the great advantage that the analysis of the model built can be verified. The purpose of classification is to allocate entities of a population to one of the various types with the use of a collection of attributes recorded on each entity. To be more precise, in the occasion of insurance underwriters, the collection of attributes might consist of the insurance application as well as the attribute of the insurance product that is proposed as a response to the application. These entities are predetermined and can either be in order or not. The insurance product can have two situations: either be accepted or rejected. The machine learning classifiers are using training data sets that contain columns and classes, which help the classifier learn and build the model.

In their book, (Han et al., 2011) support that data mining is suggested as producing no leads from vast quantities of data. The exact explanation of the term data mining is actually “producing knowledge by mining data”. So, “knowledge mining” is an easy alternative that may
not give the exact meaning. The word “mining” refers to the fact that the output of the procedure will produce invaluable results. There are synonyms to the word data mining such as knowledge mining from data, knowledge extraction and the most known old-fashioned Knowledge Discovery from Data (KDD). It is important to mention that the process of data mining produces a valuable knowledge discovery.

2.3 Description of machine learning algorithms

In this subsection, all the algorithms that were used in the analysis will be described in the next sections.

2.3.1 Decision tree algorithms

According to (Witten & Frank, n.d.), “a “divide-and-conquer” method to the problem of learning from a set of independent instances leads naturally to a style of representation called a decision tree (or tree-based). Nodes in a decision tree involve testing a particular attribute. Usually, the test at a node compares an attribute value with a constant. However, some trees compare two attributes with each other, or use some function of one or more attributes. Leaf nodes give a classification that applies to all instances that reach the leaf, or a set of classifications, or a probability distribution over all possible classifications. To classify an unknown instance, it is routed down the tree according to the values of the attributes tested in successive nodes, and when a leaf is reached the instance is classified according to the class assigned to the leaf.” (Figure 1)
2.3.1.1 J-48 (or C4.5)

(Wu et al., 2008) claim that “Given a set S of cases, C4.5 first grows an initial tree using the divide-and-conquer algorithm as follows:

- If all the cases in S belong to the same class or S is small, the tree is a leaf labeled with the most frequent class in S.
- Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets S1, S2,... according to the outcome for each case, and apply the same procedure recursively to each subset.” and that

- “Pruning is carried out from the leaves to the root. The estimated error at a leaf with N cases and E errors is N times the pessimistic error rate as above. For a subtree, C4.5 adds the estimated errors of the branches and compares this to the estimated error if the subtree is replaced by a leaf; if the latter is no higher than the former, the subtree is pruned. Similarly, C4.5 checks the estimated error if the subtree is replaced by one of its branches and when
this appears beneficial the tree is modified accordingly. The pruning process is completed in one pass through the tree.”

2.3.1.2 Random Tree

As (Kalmegh, 2015) supports “Random trees have been introduced by Leo Breiman and Adele Cutler. The algorithm can deal with both classification and regression problems. Random trees is a collection (ensemble) of tree predictors that is called forest. The classification works as follows: the random trees classifier takes the input feature vector, classifies it with every tree in the forest, and outputs the class label that received the majority of “votes”. In case of a regression, the classifier response is the average of the responses over all the trees in the forest.”

Also, (Ali et al., 2012) state that “A random tree is a tree constructed randomly from a set of possible trees having K random features at each node. “At random” in this context means that in the set of trees each tree has an equal chance of being sampled. Or we can say that trees have a “uniform” distribution. Random trees can be generated efficiently and the combination of large sets of random trees generally leads to accurate models. There has been an extensive research in the recent years over Random trees in the field of machine Learning.

2.3.1.3 Random Forest

In their work, (Ali et al., 2012) claim that “Random Forest developed by Leo Breiman is a group of un-pruned classification or regression trees made from the random selection of samples of the training data. Random features are selected in the induction process. Prediction is made by aggregating (majority vote for classification or averaging for regression) the predictions of the ensemble. Each tree is grown:

- By Sampling N randomly, if the number of cases in the training set is N but with replacement, from the original data. This sample will be used as the training set for growing the tree.
- For M number of input variables, the variable m is selected such that m<<M is specified at each node, m variables are selected at random out of the M and the best split on these m is used for splitting the node. During the forest growing, the value of m is held constant.
- Each tree is grown to the largest possible extent.
No pruning is used. Random Forest generally exhibits a significant performance improvement as compared to single tree classifier such as C4.5. The generalization error rate that it yields compares favorably to Adaboost, however it is more robust to noise.”

2.3.1.4 Simple Cart

(Kalmegh, 2015) contends that “Simple Cart method is CART (Classification And Regression Tree) analysis. CART is abbreviated as Classification and Regression Tree algorithm. It was developed by Leo Breiman in the early 1980s. It is used for data exploration and prediction also. Classification and regression trees are classification methods which in order to construct decision trees uses historical data. CART uses learning sample which is a set of historical data with preassigned classes for all observations for building decision tree. Simple Cart (Classification and regression tree) is a classification technique that generates the binary decision tree. Since output is binary tree, it generates only two children. Entropy is used to choose the best splitting attribute. Simple Cart handles the missing data by ignoring that record. This algorithm is best for the training data. Classification and regression trees (CART) decision tree is a learning technique, which gives the results as either classification or regression trees, depending on categorical or numeric data set.

According to (Wu et al., 2008) “CART splitting rules are always couched in the form An instance goes left if CONDITION, and goes right otherwise, where the CONDITION is expressed as “attribute Xi <= C” for continuous attributes. For nominal attributes the CONDITION is expressed as membership in an explicit list of values. The CART authors argue that binary splits are to be preferred because (1) they fragment the data more slowly than multi-way splits, and (2) repeated splits on the same attribute are allowed and, if selected, will eventually generate as many partitions for an attribute as required. Any loss of ease in reading the tree is expected to be offset by improved performance. A third implicit reason is that the large sample theory developed by the authors was restricted to binary partitioning. The CART monograph focuses most of its discussion on the Gini rule, which is similar to the better known entropy or information-gain criterion. For a binary (0/1) target the “Gini measure of impurity” of a node t is

\[ G(t) = 1 - p(t)^2 - (1 - p(t))^2 \] (1)
where \( p(t) \) is the (possibly weighted) relative frequency of class 1 in the node, and the improvement (gain) generated by a split of the parent node \( P \) into left and right children \( L \) and \( R \) is:

\[
I(p) = G(p) - q \cdot G(L) - (1 - q) \cdot G(R)
\]  

(2)

Here, \( q \) is the (possibly weighted) fraction of instances going left.

### 2.3.1.5 Best First Tree

In their work, (Rebecca Jeya Vadhanam et al., 2016) support that “In Best-First (BF) decision tree algorithm, the tree expands by selecting the node which maximizes the impurity reduction among all the existing nodes to split. In this algorithm, the impurity could be measured by the Gini index and information gain. BF tree are constructed in a divide-conquer method similar to the standard depthfirst decision trees. The basic step for constructing the best-first tree is given below:

- Select an attribute to place at the root node and make some branches for this attribute based on some criteria.
- Split training instances into subsets, one for each branch extending from the root node.
- Constructing process continues until all nodes are pure or a specific number of expansions are reached.”

### 2.3.1.6 Logistic Model Tree

As (Landwehr et al., 2003) point out “A logistic model tree basically consists of a standard decision tree structure with logistic regression functions at the leaves, much like a model tree is a regression tree with regression functions at the leaves. As in ordinary decision trees, a test on one of the attributes is associated with every inner node. For a nominal (enumerated) attribute with \( k \) values, the node has \( k \) child nodes, and instances are sorted down one of the \( k \) branches depending on their value of the attribute. For numeric attributes, the node has two child nodes and the test consists of comparing the attribute value to a threshold: an instance is sorted down the left branch if its value for that attribute is smaller than the threshold and sorted down the right branch otherwise. More formally, a logistic model tree consists of a tree structure that is
Aristotle University
MSc in Logistics & Supply Chain Management

made up of a set of inner or non-terminal nodes \( N \) and a set of leaves or terminal nodes \( T \). Let \( S \) denote the whole instance space, spanned by all attributes that are present in the data. Then the tree structure gives a disjoint subdivision of \( S \) into regions \( S_t \), and every region is represented by a leaf in the tree:

\[
S = \bigcup_{t \in T} S_t, S_t \cap S_{t'} = \emptyset \quad \forall \alpha \neq t'
\]  

(3)

Unlike ordinary decision trees, the leaves \( t \in T \) have an associated logistic regression function \( f_t \) instead of just a class label. The regression function \( f_t \) takes into account a subset \( V_t \subseteq V \) of all attributes present in the data (where we assume that nominal attributes have been binarized for the purpose of regression), and models the class membership probabilities as

\[
Pr(G = j|X = x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}
\]  

(4)

where

\[
F_j(x) = a_0^j + \sum_{v \in V_t} a_v^j \cdot v,
\]  

(5)

or, equivalently,

\[
F_j(x) = a_0^j + \sum_{k=1}^{m} a_{v_k}^j \cdot v_k
\]  

(6)

if \( a_{v_k}^j = 0 \) \( \forall \alpha \not\in V_t \). The model represented by the whole logistic model tree is then given by

\[
f(x) = \sum_{t \in T} f_t(x) \cdot I(x \in S_t)
\]  

(7)

where \( I(x \in S_t) \) is 1 if \( x \in S_t \) and 0 otherwise."

2.3.1.7 Decision Stump

As (Sugiyama, 2016) supports, “As an example of a weak learning algorithm, let us consider a decision stump classifier, which is a depth-one version of decision trees. More specifically, a decision stump classifier randomly chooses one of the elements in the \( d \)-dimensional input vector \( x = (x^{(1)},...,x^{(d)})^T \) and classification is performed by thresholding the chosen element. This means that the decision boundary is parallel to one of the coordinate axes.
The decision stump may be a poor classifier in terms of the classification accuracy because of its low degree of freedom. Nevertheless, at least, it has advantage in computation costs because there only exist $n + 1$ solutions for $n$ training samples. Indeed, the global optimal solution can be easily obtained by just sorting the $n$ training samples along the chosen axis and find the best interval that minimizes the classification error."

2.3.1.8 Hoeffding Tree

As (Srimani & Patil, 2015) explain “One of the basic algorithm for stream data classification is Hoeffding tree algorithm. It is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams assuming that the distribution generating examples does not change over time. It produces decision trees which are similar to traditional batch learning method. Hoeffding trees and decision trees are asymptotically related. HT algorithm is based on a simple idea that a small sample can be often sufficient to choose an optimal splitting attribute. The key point to be noted here is traditional batch learning methods and also generate decision trees based on splitting attributes. Mathematically, it is proved that HT algorithm uses Hoeffding bound. To understand the meaning of Hoeffding bound few assumptions are made. Suppose we make ‘N’ independent observations of a random variable ‘$r$’ with range ‘R’, where ‘$r$’ is an attribute selection measure. In case of Hoeffding trees ‘$r$’ is information gain and if we compute the mean value of $r$ ‘$r_{mean}$’ of this sample the Hoeffding bound states that the true mean of ‘$r$’ is at least $1 - \delta$ where $\delta$ is user specified and Performance analysis of Hoeffding trees in data streams

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}$$

(8)

Hoeffding bound is independent of probability distribution. Main advantages of HT algorithm are:

1. it is incremental in nature
2. achieves high accuracy using small sample
3. multiple scans on same data are never performed.
But the main disadvantage is HT cannot handle concept drift because once the node is created it can never be changed. The algorithm spends a great deal of time with attributes that have nearly identical splitting quality. In addition, the memory utilisation can be further optimised. The Hoeffding tree algorithm is presented in the next Figure.

Figure 2 Algorithm for Hoeffding tree

2.3.2 Bayesian algorithms

In their work, (Wu et al., 2008) claim that “Given a set of objects, each of which belongs to a known class, and each of which has a known vector of variables, our aim is to construct a rule which will allow us to assign future objects to a class, given only the vectors of variables describing the future objects. Problems of this kind, called problems of supervised classification, are ubiquitous, and many methods for constructing such rules have been developed. One very important one is the naive Bayes method—also called idiot’s Bayes, simple Bayes, and independence Bayes. This method is important for several reasons. It is very easy to construct, not needing any complicated iterative parameter estimation schemes. This means it may be readily applied to huge data sets. It is easy to interpret, so users unskilled in classifier technology can understand why it is making the classification it makes. And finally, it often does surprisingly well: it may not be the best possible classifier in any particular application, but it can usually be
relied on to be robust and to do quite well.” So, the Bayesian algorithms used in this thesis are the following:

2.3.2.1 Naïve Bayes

As (Wu et al., 2008) state “For convenience of exposition here, we will assume just two classes, labeled $i = 0, 1$. Our aim is to use the initial set of objects with known class memberships (the training set) to construct a score such that larger scores are associated with class 1 objects (say) and smaller scores with class 0 objects. Classification is then achieved by comparing this score with a threshold, $t$. If we define $P(i|x)$ to be the probability that an object with measurement vector $x = (x_1, ..., x_p)$ belongs to class $i$, then any monotonic function of $P(i|x)$ would make a suitable score. In particular, the ratio $P(1|x)/P(0|x)$ would be suitable. Elementary probability tells us that we can decompose $P(i|x)$ as proportional to $f(x|i)P(i)$, where $f(x|i)$ is the conditional distribution of $x$ for class $i$ objects, and $P(i)$ is the probability that an object will belong to class $i$ if we know nothing further about it (the ‘prior’ probability of class $i$). This means that the ratio becomes:

$$\frac{P(1|x)}{P(0|x)} = \frac{f(x|1)P(1)}{f(x|0)P(0)}$$

(9)

To use this to produce classifications, we need to estimate the $f(x|i)$ and the $P(i)$. If the training set was a random sample from the overall population, the $P(i)$ can be estimated directly from the proportion of class $i$ objects in the training set. To estimate the $f(x|i)$, the naive Bayes method assumes that the components of $x$ are independent, $f(x|i) = \prod_{j=1}^{p} f(x_j|i)$, and then estimates each of the univariate distributions $f(x_j|i), j = 1, ..., p; i = 0,1$, separately. Thus the $p$ dimensional multivariate problem has been reduced to $p$ univariate estimation problems. Univariate estimation is familiar, simple, and requires smaller training set sizes to obtain accurate estimates. This is one of the particular, indeed unique attractions of the naive Bayes methods: estimation is simple, very quick, and does not require complicated iterative estimation schemes. If the marginal distributions $f(x_j|i)$ are discrete, with each $x_j$ taking only a few values, then the estimate $\hat{f}(x_j|i)$ is a multinomial histogram type estimator (see below)—simply counting the proportion of class $i$ objects which fall into each cell. If the $f(x_j|i)$ are continuous, then a
common strategy is to segment each of them into a small number of intervals and again use multinomial estimator, but more elaborate versions based on continuous estimates (e.g. kernel estimates) are also used.

Given the independence assumption, the ratio in (9) becomes:

\[
\frac{P(1|x)}{P(0|x)} = \frac{\prod_{j=1}^{p} f(x_j|1)P(1)}{\prod_{j=1}^{p} f(x_j|0)P(0)} = \frac{P(1)}{P(0)} \prod_{j=1}^{p} \frac{f(x_j|1)}{f(x_j|0)}
\]

(10)

Now, recalling that our aim was merely to produce a score which was monotonically related to \(P(i|x)\), we can take logs of (10)—log is a monotonic increasing function. This gives an alternative score

\[
\ln \frac{P(1|x)}{P(0|x)} = \ln \frac{P(1)}{P(0)} + \sum_{j=1}^{p} \ln \frac{f(x_j|1)}{f(x_j|0)}
\]

(11)

If we define \(w_j = \ln \left( \frac{f(x_j|1)}{f(x_j|0)} \right)\) and a constant \(k = \ln \left( \frac{P(1)}{P(0)} \right)\) we see that (11) takes the form of a simple sum

\[
\ln \frac{P(1|x)}{P(0|x)} = k + \sum_{j=1}^{p} w_j
\]

(12)

so that the classifier has a particularly simple structure.”

2.3.2.2 Naïve Bayes Updateable

(Ren et al., 2014) claim that “the incremental learning of Naïve Bayesian classifier is actually a recursive Bayesian estimation of parameters. Its advantage is that it can preserve the information in initial training data in the form of parameters. During the process of incremental learning, the system does not need to visit the raw data, but only needs to access the two saved statistic parameters. According to the information in new training set, the system amends and resaves the two statistic parameters. So, the incremental learning formula of Naïve Bayesian classifier is deduced by the statistical knowledge. Suppose D is the current training set, T is the new testing set, \(x_p = (A_1, A_2, ..., A_3, ..., A_4) \in T\) is the new instance for amending, and \(c_p\) means
its class label. \( \theta_j = P(c = c_j) \) is class prior probability of class label \( c_j \). Then the amending formula of class prior probability is:

\[
\theta_j = \begin{cases} 
\frac{s}{s+1} \theta_j + \frac{1}{s+1}, & \text{if } \tau \alpha c_p = c_j \\
\frac{s}{s+1} \theta_j, & \text{if } \tau \alpha c_p \neq c_j
\end{cases}
\] (13)

Where, \( s = |D| + |T| \)

\( |D| \): represents number of samples in training set \( D \),

\( |T| \): represents number of samples in new testing set \( T \).

Suppose \( \theta_{ikj} = P(A_i = \alpha_k | c = c_j) \) is class prior probability of attribute \( A_i \) of value \( \alpha_k \) in class \( c_j \). Then the amending formula of class conditional probability is:

\[
\theta_{ikj} = \begin{cases} 
\frac{m}{1+m} \theta_{ikj} + \frac{1}{1+m}, & \text{if } \tau \alpha c_p = c_j \text{ και } A_i = \alpha_k \\
\frac{m}{1+m} \theta_{ikj}, & \text{if } \tau \alpha c_p = c_j \text{ και } A_i \neq \alpha_k \\
\theta_{ikj}, & \text{if } \tau \alpha c_p \neq c_j
\end{cases}
\] (14)

\( m = |A_i| + \text{count}(c_j) \) (15)

\( |A_i| \): represents the numbers of values for attribute \( A_i \),

\( \text{count}(c_j) \) represents number of samples whose class label is \( c_j \)

Then the incremental amending of Naïve Bayesian classifier is completed after the modification of the two statistical parameters according to the (13), (14) and (15).”

2.3.2.3 Bayesian Network

In his work, (Bouckaert, 2004) contends that “Let \( U = \{x_1, \ldots, x_n\} \), \( n \geq 1 \) be a set of variables. A Bayesian network \( B \) over a set of variables \( U \) is a network structure \( B_S \), which is a directed acyclic graph (DAG: Directed Acyclic Graph) over \( U \) and a set of probability tables \( B_p = \{p(u|pa(u)) | u \in U\} \), where \( pa(u) \) is the set of parents of \( u \) in \( B_S \). A Bayesian network represents a probability distributions:

\[
P(U) = \prod_{u \in U} p(u|pa(u))
\] (16)
The classification task consists of classifying a variable $y = x_0$ called the class variable given a set of variables $x = x_1, ..., x_n$, called attribute variables. A classifier $h : x \rightarrow y$ is a function that maps an instance of $x$ to a value of $y$. The classifier is learned from a dataset $D$ consisting of samples over $(x, y)$. The learning task consists of finding an appropriate Bayesian network given a dataset $D$ over $U$.

To use a Bayesian network as a classifier, one simply calculates $\arg\max_y P(y|x)$ using the distribution $P(U)$ represented by the Bayesian network. Now note that:

$$P(y|x) = P(U)/P(x)$$

$$\propto P(U)$$

$$= \prod_{u \in U} p(u|pa(u))$$

And since all variables in $x$ are known, we do not need complicated inference algorithms, but just calculate (19) for all class values.”

2.3.2.4 Naïve Bayes Multinomial Text

(Xu et al., 2017) state that “As a matter of fact, multinomial NB classifier is a generative model. It assumes that a corpus of documents is generated by selecting a class $c_m \sim \theta$ for a document $m$ then generating each word of that document independently according to a class-specific distribution $w_{mn} \sim \varphi_{cm}$. The bag-of-words assumption is clearly violated by natural language texts. In multinomial NB classifier, every word $w_i$ gets a say in determining which label $c \in \{1, ..., C\}$ should be assigned to an unseen document $\tilde{w} = (\tilde{w}_1, ..., \tilde{w}_n)$. In order to choose a label $c$ for $\tilde{w}$, multinomial NB classifier begins by calculating the prior probability $Pr(c)$ of each label, which is determined by assuming equiprobable classes, or checking the frequency of each label in the training set. The contribution from each word is then combined with this prior probability, to arrive at a likelihood estimate for each label. This is known as the maximum a posteriori (MAP) decision rule. It can be defined formally as follows.

$$c = \arg\max_c Pr(c) \prod_{n=1}^{N} Pr(\tilde{w}_n | c) = \arg\max_c \theta_c \prod_{n=1}^{N} \varphi_{c, \tilde{w}_n}$$

(20)
Given a training document set \( D = \{(\tilde{w}_m, C_m)\}_{m=1}^M \) and \( \theta_C, \varphi_{cy} \), are usually estimated by a smoothed version of maximum likelihood (ML) as follows. In fact, these are both MAP estimates given uniform Dirichlet priors (see further).

\[
\hat{\theta}_c = \frac{n_c^{(c)} + \alpha}{M + \alpha}
\]

\[
\hat{\varphi}_{c,v} = \frac{n_v^{(c)} + \beta}{n_c + v\beta}
\]

where \( V \) is the number of unique words,

\( n_c^{(c)} \) is the number of documents with class \( c \) in \( D \),

\( n_v^{(c)} \) is the number of times word \( v \) appears in a document of class \( c \) in \( D \), and

\[ n_c = \sum_{v=1}^{V} n_v^{(c)} \]. The smoothing priors \( \alpha \geq 0, \beta \geq 0 \) prevent zero probabilities in further computations. Setting \( \alpha = 1, \beta = 1 \) are called Laplace smoothing, while \( \alpha < 1, \beta < 1 \) is called Lidstone smoothing."

2.3.3 Instance-based algorithms

In their book, (Witten & Frank, n.d.) claim that “Although there are other possible choices, most instance-based learners use Euclidean distance. The distance between an instance with attribute values

\( \alpha_1^{(1)}, \alpha_2^{(1)}, \ldots, \alpha_k^{(1)} \) (where \( k \) is the number of attributes)

and one with values: \( \alpha_1^{(2)}, \alpha_2^{(2)}, \ldots, \alpha_k^{(2)} \) is defined as

\[
\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \cdots + (a_k^{(1)} - a_k^{(2)})^2}
\]

When comparing distances, it is not necessary to perform the square root operation; the sums of squares can be compared directly.”

2.3.3.1 K-Nearest Neighbors (K-NN)
As (Wu et al., 2008) support “One of the simplest, and rather trivial classifiers is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of the test object match one of the training examples exactly. An obvious drawback of this approach is that many test records will not be classified because they do not exactly match any of the training records. A more sophisticated approach, k-nearest neighbor (kNN) classification, finds a group of k objects in the training set that are closest to the test object, and bases the assignment of a label on the predominance of a particular class in this neighborhood. There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance or similarity metric to compute distance between objects, and the value of k, the number of nearest neighbors. To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class labels of these nearest neighbors are then used to determine the class label of the object. Given a training set D and a test object x = (x', y'), the algorithm computes the distance (or similarity) between z and all the training objects (x, y) ∈ D to determine its nearest-neighbor list, Dz. (x is the data of a training object, while y is its class. Likewise, x' is the data of the test object and y' is its class.) Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

\[ y' = \text{argmax} \sum_{(x_i, y_i) \in D_z} I(v = y_i) \]

(24)

where \( v \) is a class label, \( y_i \) is the class label for the ith nearest neighbors,

\( I \): is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

- **Input**: the set of training objects and test object z = (x', y')
- **Process**: Compute \( d(x', x) \), the distance between and every object, (x, y) ∈ D. Select \( D_z \subseteq D \) the set of k closest training objects to z.
- **Output**: \( y' = \text{argmax} \sum_{(x_i, y_i) \in D_z} I(v = y_i) \)

### 2.3.3.2 K-Star (K*)

(Cleary & Trigg, n.d.) suggest that “Let I be a (possibly infinite) set of instances and T a finite set of transformations on I. Each t ∈ T maps instances to instances: \( t: I \to I \). T contains a
distinguished member $\sigma$ (the stop symbol) which for completeness maps instances to themselves ($\sigma(a) = a$). Let $P$ be the set of all prefix codes from $T^*$ which are terminated by $\sigma$. Members of $T^*$ (and so of $P$) uniquely define a transformation on $I$:

$$\bar{t}(\alpha) = t_n(t_{n-1}(\ldots t_1(\alpha)\ldots)), \ \text{όπως} \ \bar{t} = t_1, \ldots t_n$$  \hspace{1cm} (25)

A probability function $p$ is defined on $T^*$. It satisfies the following properties:

$$0 \leq \frac{p(Eu)}{p(\bar{t})} \leq 1$$  \hspace{1cm} (26)

$$\sum_u p(\bar{t}u) = p(\bar{t})$$  \hspace{1cm} (27)

$$p(\Lambda) = 1$$  \hspace{1cm} (28)

As a consequence it satisfies the following:

$$\sum_{\bar{t} \in P} p(\bar{t}) = 1$$  \hspace{1cm} (29)

The probability function $P^*$ is defined as the probability of all paths from instance $a$ to instance $b$:

$$\sum_b P^* (b|a) = 1$$  \hspace{1cm} (30)

$$0 \leq P^* (b|a) \leq 1$$  \hspace{1cm} (31)

The $K^*$ function is then defined as:

$$K^* (b|a) = -\log_2 P^* (b|a)$$  \hspace{1cm} (32)

$K^*$ is not strictly a distance function. For example, $K^*(a|a)$ is in general non-zero and the function (as emphasised by the $|$ notation) is not symmetric. Although possibly counter-intuitive the lack of these properties does not interfere with the development of the $K^*$ algorithm below. The following properties are provable:

$$K^* (b|a) \geq 0$$  \hspace{1cm} (33)

$$K^* (c|b) + K^* (b|a) \geq K^* (c|a)$$  \hspace{1cm} (34)
2.3.4 Neural Network

In this subsection, the neural network that was used in this dissertation will be explained thoroughly:

2.3.4.1 Multilayer Perceptron

As (Arora et al., 2012) suggest “Multilayer Perceptron classifier is based upon backpropagation algorithm to classify instances. The network is created by an MLP algorithm. The network can also be monitored and modified during training time. The nodes in this network are all sigmoid (except for when the class is numeric in which case the output nodes become unthresholded linear units). The backpropagation neural network is essentially a network of simple processing elements working together to produce a complex output. The backpropagation algorithm performs learning on a multilayer feed-forward neural network. It iteratively learns a set of weights for prediction of the class label of tuples. A multilayer feed-forward neural network consists of an input layer, one or more hidden layers, and an output layer. An example of a multilayer feed-forward network is shown in next Figure.

![A multilayer feed-forward neural network](image)

*Figure 3 A multilayer feed-forward neural network*

Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple. The inputs are fed simultaneously into the units making up the
input layer. These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of “neuronlike” units, known as a hidden layer. The outputs of the hidden layer units can be input to another hidden layer, and so on. The number of hidden layers is arbitrary, although in practice, usually only one is used. At the core, backpropagation is simply an efficient and exact method for calculating all the derivatives of a single target quantity (such as pattern classification error) with respect to a large set of input quantities (such as the parameters or weights in a classification rule). To improve the classification accuracy, we should reduce the training time of neural network and reduce the number of input units of the network.”

2.4 Confusion Matrices

In their book, (Han et al., 2012) support that there are four essential terms that have to be understood deeply in order for someone to be able to grasp the meaning of the evaluation measurements that will be used in this dissertation. These terms are:

1. True positives (TP): These are the correctly positive labeled instances that were classified by the classifier.
2. True negatives (TN): These are the correctly negative labeled instances that were classified by the classifier.
3. False positives (FP): These are the negative instances that were incorrectly labeled as positive.
4. False negatives (FN): These are the positive instances that were mislabeled as negative.

These terms are represented in the confusion matrix of next Figure. The confusion matrix is a advantageous technique for indicating how well a classifier can recognize instances of different labels. TP and TN indicate when the classifier is being correct, while FP and FN show when the classifier is being incorrect.
2.5 Evaluation measurements

(Han et al., 2012) also support that the accuracy of a classifier on a examined set is the ratio of the examined set instances that are correctly classified by the classifier. That is,

\[ Accuracy = \frac{TP + TN}{P + N} \] (35)

This is also known as the overall recognition rate of the classifier and it indicates how well the classifier recognizes rows of the various labels. By having a look at a confusion matrix, it is clear if the executed algorithm is confusing two classes. Accuracy is most fruitful when the label distribution is nearly steady. Also there is the error rate or misclassification rate of a classifier, M, which is simply \( 1 - \text{accuracy}(M) \), where \( \text{accuracy}(M) \) is the accuracy of M. This also can be computed as

\[ \text{Error rate} = \frac{FP + FN}{P + N} \] (36)

The sensitivity (recall) and specificity measures can be used, respectively, as well. Sensitivity is also known as the true positive (recognition) rate (the fraction of positive instances that are correctly identified), while specificity is the true negative rate (the fraction of negative instances that are correctly identified). These measures are specified as

\[ Sensitivity = \frac{TP}{P} \] (37)

\[ Specificity = \frac{TN}{N} \] (38)
In order to calculate the F-measure, we need to define the precision first, as well as the recall, which was analyzed above.

\[
\text{Precision} = \frac{TP}{TP+FP}
\] (37.1)

The F-measure is defined as a harmonic mean of precision (P) and recall (R) and the highest possible value is 1.0 showing a perfect ratio of precision and recall. F-measure is defined as (Sasaki & Fellow, 2007):

\[
F = \frac{2PR}{P+R}
\] (37.2)

The AUC-ROC Curve is a performance metric representing how well the classifier can distinguish between the two labels, positive and negative. The closer to 1, the better the performance of the classifier. The AUC-ROC Curve measurement is specified as (Gonçalves et al., 2014):

\[
AUC = \int_0^1 ROC(u)du
\] (38.1)

The ROC curve is defined as a plot of Se(c) versus 1−Sp(c) for \(-\infty \leq c \leq \infty\), or equivalently as a plot of:

\[
ROC(t) = 1 - G(F^{-1}(1 - t))
\]

over \(t \in [0,1]\), where \(F^{-1}(1 - t) = \inf\{x \in R: F(x) \geq 1 - t\}\)

2.6 Random Under-sampling

As (García-Pedrajas et al., 2012) support that the majority of classification techniques experience an imbalance distribution on the training set between the labels. Most of the machine learning algorithms to work efficiently and produce reliable results should be granted with a balanced distribution of the instances among the labels. Facing the class imbalance problem is not an easy task to do and most of the real-world problems face this particular problem of imbalance, some examples are gene recognition, intrusion detection, web mining, etc.
Frequently, the problem of imbalance appears in datasets with binary classification. In this situation there is a label of high interest, that is rarely present in the dataset, along with another label that exists in abundance. In highly imbalanced problems the proportion between the two labels can be sometimes as high as 1:1000 or even more. A lot of scientists have come up with methods and algorithms that propose a groundbreaking solution to this problem of imbalance and improve the results of machine learning algorithms.

(He, 2010) explains that “Random under-sampling removes data from the original data set. In particular, we randomly select a set of majority class examples in \( S_{maj} \), and remove these samples from \( S \), so that \( |S| = |S_{min}| + |S_{maj}| - |E| \). Consequently, undersampling readily gives us a simple method for adjusting the balance of the original data set \( S \).”

2.7 Cross-validation

(Witten & Frank, n.d.) support that the most widespread technique of producing the results of a machine learning algorithm is the 10-fold cross-validation. The selected dataset is split
randomly into 10 pieces in which the labels exist in approximately the same ratio as in the full dataset. Each piece is used for testing purposes when the remaining nine-tenths are given to the algorithm to “learn” (as a training set). Consequently, that process is executed 10 times with a different nine-tenths each time. Each execution produces a result that is being marked down and after it is executed 10 times, it calculates the average error and presents it to the user. The 10-fold cross-validation has been researched extensively and has shown that it is the perfect number to use (see Figure 5).

Furthermore, (Berrar, 2018) asserts that “In the first fold, the first subset serves as a validation set $D_{val,1}$ and the remaining nine subsets serve as training set $D_{train,1}$. In the second fold, the second subset is the validation set and the remaining subsets are the training set, and so on. The cross-validated accuracy, for example, is the average of all ten accuracies achieved on the validation sets. More generally, let $\hat{f}_{-k}$ denote the model that was trained on all but the $k^{th}$ subset of the learning set. The value $\hat{y}_i = \hat{f}_{-k}(x_i)$ is the predicted or estimated value for the real class label, $y_i$, of case $x_i$, which is an element of the $k^{th}$ subset. The cross-validated estimate of the prediction error, $\hat{\epsilon}_{cv}$, is then given as:

$$\hat{\epsilon}_{cv} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}_{-k}(x_i))$$  (39)
In this subsection, there will be a reference to the machine learning tool that was used to realize this project. According to (Frank et al., 2016), the Weka software is a powerful tool for data analysis.

**Figure 5** Representation of 10-fold cross validation.

### 2.8 WEKA Tool

In this subsection, there will be a reference to the machine learning tool that was used to realize this project. According to (Frank et al., 2016), the Weka software is a powerful tool for data analysis.

**Figure 6** "Waikato Environment for Knowledge Analysis" provides 5 applications.
mining and it consists of the following five applications summarily (as seen in the next Figure 6):

1. “Explorer” provides access to all of its Weka utilities with menu selection and form filling. It gives the user the ability to load a dataset and perform a variety of tasks that will be explained later in this section. 2. “Experimenter” is developed to help the user understand the implementation of classification and regression methods 3. “Knowledge Flow” is appropriate for streamed data processing through an advantageous user interface with the ability to “drag and drop” boxes and data sources around the screen. 4. “Workbench” is a unified graphical user interface that merges the aforementioned three applications (Explorer, Experimenter, and Knowledge Flow) into one application, as well as provides the user with the in-depth ability to customizations. 5. “SimpleCLI” produces a simple command-line interface to the user, to be able to type commands (for advanced users only).

After the summary of all the applications that are provided with the Weka software tool, it will follow a brief presentation of the “Explorer” application that was used in depth in this dissertation. Everything in the practical aspect was utilized through this application, so it is extremely important to mention all the capabilities that this application provides.

The “Explorer” application is a well-built implementation that provides the user with six utilitarian tabs: 1. “Preprocess” provides the user the ability to load, prepare, preprocess and perform an initial analysis on the dataset. 2. “Classify” delivers the utilities concerning the classification process, containing a variety of machine learning supervised algorithms (classification or regressions) to execute and produce useful results. 3. “Cluster” caters capabilities concerning the clustering process, containing a variety of machine learning unsupervised algorithms to execute and produce helpful results. 4. “Associate” is beneficial by acquiring knowledge of association rules for the data. 5. “Select attributes” supplies numerous algorithms that can be used to select the most applicable features of the dataset 6. “Visualize” contributes by giving two-dimensional graphs and plots of the dataset.

At this point, after the presentation of the utilities of the “Explorer” application, there will be a brief analysis of the two tabs that are being used mostly in this dissertation. More specifically, these tabs have to do with the data preprocessing as well as with the classification process, where the machine learning algorithms are selected and then executed to be evaluated.
In the “Preprocess” tab, the user can manipulate a chosen dataset, as far as the data preprocessing is concerned, with a variety of numerous and useful filters. On the same tab, the user can find useful information concerning the dataset he has loaded, such as the multitude of instances and attributes. Weka tool gives the capability to the user to use visualization techniques, so with a few mouse clicks, it can present practical and convenient graphs concerning the distributions and the correlations between the variables. This toolkit is accessed through a friendly user interface so that its users can compare different methods and recognize which are the most suitable for the specific case (see Figure 7).

![Figure 7 Weka representation of "Preprocess" tab](image)

As far as the “Classify” tab is concerned, the user can navigate and select through a Variety of Machine learning algorithms dot can be seen on the “Classifier” area, by clicking the “Choose” button (see Figure 8).
After selecting the preferred algorithm, the user can choose through a variety of options on the “Test options” area and by clicking the start button, this selected algorithm is being executed. In the area that is called “Classifier output”, the user can look at useful information when the selected algorithm is executed.
execution of the algorithm completes. This information includes confusion matrices, evaluation measurements that were discussed earlier in this dissertation. (see Figure 9)

As (Witten & Frank, n.d.) state “Weka was developed at the University of Waikato in New Zealand, and the name stands for Waikato Environment for Knowledge Analysis. Outside the university the weka, pronounced to rhyme with Mecca, is a flightless bird with an inquisitive nature found only on the islands of New Zealand.”

Finally, the WEKA machine-learning software provides an easy way, with a friendly user interface that is addressed to people with no previous experience in the field of machine learning and data mining because it allows them to be able to get through the whole machine learning and data mining process without the need to type (not even) a single line of code, something that would be mandatory in any data-related programming language such as R Programming Language and Python. The Weka software tool is developed in Java programming language, is using files in ARFF format, and is available from http://www.cs.waikato.ac.nz/ml/weka.
CHAPTER 3 – Methodology

3.1 General description of the methodology

In this section, the general description of the methodology that was applied to this project will be presented (Graph 2). The dataset that consists of instances of products, that either went on backorder or not, named “bopredict” was firstly converted from .csv to .arff format (with the name “bopredict.arff”) so that it is compatible with the WEKA machine learning tool. Subsequently, the statistical technique of Under-sampling was applied to the dataset (check subsection 2.6 for a brief explanation), so that both labels have the same multitude of rows (13.981 registrations on each label based on column ‘BO’). It has to be mentioned that the multitude of instances of class ‘0’, which is the class with 1.915.954 registrations, after the application of random undersampling it reached 13.981 rows, just as the class the multitude of rows of class ‘1’. Then, the machine learning algorithms that are presented in section 2.4 were executed with the use of the statistical method called “Cross-Validation” with 10 folds (10-fold Cross-Validation can be found in section 2.7), and the results were recorded, to be evaluated. 

Graph 2 Representation of methodology
this point, it is mandatory to highlight that all the registrations that exist in the dataset are unique and there are no duplicates, which would undoubtedly distort the results.

3.2 Exploratory data analysis and visualization

As (R. De Santis, 2019) points out that “The dataset contains historical data for inventory-active products from the previous 8 weeks of the week we would like to predict, captured as a photo of all inventory at the beginning of the week.”

As far as the attributes of the data are concerned, the following figure explains the existence of each column in the dataset. (R. B. De Santis et al., 2018)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Definition</th>
<th>Data type of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKU</td>
<td>Product ID</td>
<td>Integer</td>
</tr>
<tr>
<td>INV</td>
<td>Current inventory level of material</td>
<td>Real</td>
</tr>
<tr>
<td>TIM</td>
<td>Registered transit time (in days)</td>
<td>Integer</td>
</tr>
<tr>
<td>INQ</td>
<td>In transit quantity</td>
<td>Real</td>
</tr>
<tr>
<td>FOR3</td>
<td>Forecast sales for the next 3 months</td>
<td>Real</td>
</tr>
<tr>
<td>FOR6</td>
<td>Forecast sales for the next 6 months</td>
<td>Real</td>
</tr>
<tr>
<td>FOR9</td>
<td>Forecast sales for the next 9 months</td>
<td>Real</td>
</tr>
<tr>
<td>SAL1</td>
<td>Sales quantity for the prior 1 month</td>
<td>Real</td>
</tr>
<tr>
<td>SAL3</td>
<td>Sales quantity for the prior 3 months</td>
<td>Real</td>
</tr>
<tr>
<td>SAL6</td>
<td>Sales quantity for the prior 6 months</td>
<td>Real</td>
</tr>
<tr>
<td>SAL9</td>
<td>Sales quantity for the prior 9 months</td>
<td>Real</td>
</tr>
<tr>
<td>MIN</td>
<td>Minimum recommended amount in stock</td>
<td>Real</td>
</tr>
<tr>
<td>OVRP</td>
<td>Parts overdue from source</td>
<td>Real</td>
</tr>
<tr>
<td>SUP1</td>
<td>Supplier performance in last 1 semester</td>
<td>Real</td>
</tr>
<tr>
<td>SUP2</td>
<td>Supplier performance in last 2 semesters</td>
<td>Real</td>
</tr>
<tr>
<td>OVRA</td>
<td>Amount of stock orders overdue (amount of orders that are late to arrive)</td>
<td>Real</td>
</tr>
<tr>
<td>RSK1</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>RSK2</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>RSK3</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>RSK4</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>RSK5</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>RSK6</td>
<td>General risk flags associated to the material</td>
<td>Binary (0 or 1)</td>
</tr>
<tr>
<td>BO</td>
<td>Product went on backorder (Response variable)</td>
<td>Binary (0 or 1)</td>
</tr>
</tbody>
</table>

Table 1 Dataset explanation with definitions and data type of each variable

As already mentioned, the dataset consists of 23 features, where 16 of them are numerical and 7 of them (including the response variable) are categorical and more specifically binary (0 or
1). The first feature of ‘SKU’ refers to the Product ID, which is unique for each instance. ‘SKU’ and ‘TIM’ attributes are the only integers.

In order to be compatible with the Weka tool, the file should be converted to arff format (from csv). The following figure depicts the arff file after the conversion to arff format. It can be seen clearly that each attribute is declared with the command @attribute, followed by a name and the type of data it contains. Then the data starts after the @data command.

```
@relation bopredict-weka.filters.supervised.instance.SpreadSubsample-M1.0-X0.0-S1

@attribute SKU integer
@attribute INV numeric
@attribute TIM integer
@attribute INQ numeric
@attribute FOR3 numeric
@attribute FOR6 numeric
@attribute XFOR9 numeric
@attribute SALL numeric
@attribute SAL3 numeric
@attribute SAL6 numeric
@attribute SAL9 numeric
@attribute MIN numeric
@attribute RSK1 {0,1}
@attribute OVAP numeric
@attribute SUP1 numeric
@attribute SUP2 numeric
@attribute OVRA numeric
@attribute RSK2 {0,1}
@attribute RSK3 {0,1}
@attribute RSK4 {0,1}
@attribute RSK5 {0,1}
@attribute RSK6 {0,1}
@attribute BO {0,1}

@data
1734327,0.970143,8,0.242536,0,0,0,0,0,0,0,0,0,0,8,0.83,0,0,1,0,0
1557687,0.805557,8,0,0,0,0.075521,0.340084,0.580473,0,0,0.79,0.66,0,0,0,0,0,0,0,0,0
2810432,0.375315,12,0,0,0,0.272956,0.480435,0,0.170598,0.409435,0,0.646272,0,0,0,0.73,0.78,0,0,0,0,0,0
2890195,0.772283,8,0,0,0,0,0.677228,0.257428,0.566641,0.102971,0,0,0.782,0.776896,0,0,0,0,0,0,0,0,0
3169685,0.992278,2,0,0,0,0,0,0.124035,0,0,0,1,0.99,0,0,0,1,0,0
4550200,0,0,0,0.422222,0.625000,0.625000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

*Table 2 Representation of dataset in arff format*
The dataset consisted of 1.929.935 instances and 23 attributes (23rd is the response variable named ‘BO’). More specifically, class ‘0’ contained 1.915.954 registrations, and class ‘1’ contained 13.981 instances, which shows a highly imbalanced situation. After the application of random under-sampling, the under-sampled dataset contains 13.981 for class ‘0’, as well as 13.981 for class ‘1’. The statistical technique of under-sampling reduced the dataset by 1.901.973 instances. The final multitude of instances on the used dataset is 27.962 registrations, split evenly between the two classes (shown in the next figure).

**Figure 10 Representation of dataset before and after application of random undersampling**

At this point, there will be an analysis as far as the distribution of the features is concerned. More specifically, the features that contained binary types were eliminated from the distribution analysis, as well as the attribute with the name ‘SKU’ which contains unique numbers because they don’t produce any additional value to it. So, the features that were selected to be presented in the next figure, that depict the distribution of the attributes for each one separately, are the following: INV, TIM, INQ, FOR3, FOR6, FOR9, SAL1, SAL3, SAL6, SAL9, MIN, OVRP, SUP1, SUP2, OVRA. As far as the ‘INV’ attribute is concerned, it can be seen that there is a high aggregation of instances (12000 registrations) of class ‘1’ close to the value of zero, precisely, between -0.1 and 0.25. On the same attribute but for the class ‘0’ this time, there is an accumulation of instances (approximately 6000 observations) on high values, precisely between 0.9 and 1. The rest of the instances of class ‘0’ are distributed evenly between -0.1 and 0.9. Regarding the feature ‘TIM’, there is an agglomeration of instances on the low values between 0 and 12, showing a few
remarkable pikes on values around 2 and 7, where the multitude of instances touch the 7000 and 13000 respectively.

Figure 11 Dataset visualization regarding the distribution of the numeric variables

As for attribute ‘INQ’, it is clear from the graph above that almost all the instances are accumulated in the value of 0 with minimal exceptions in slightly greater values than 0. This probably explains that there is a small percentage in transit in these instances. Concerning the attribute ‘FOR3’, there is a high collection of instances on the lowest possible value of 0 that touches the 13000, with the majority of observations to be attributed to class ‘0’ and less to class ‘1’. The rest of the instances on this feature follow a normal distribution. Similar observations
take place on both ‘FOR6’ and ‘FOR9’ but with an obvious intensity compared to ‘FOR3’, as far as the part of the normal distribution is concerned. On ‘SAL1’, there is an accumulation of instances on a value of 0 for both classes, while the rest of the observations are found on the value not greater than 0.15. Similar patterns, as far as the distribution of the data points is concerned, with a gradual relocation reaching the so-called “normal distribution” on attributes ‘SAL3’, ‘SAL6’ and ‘SAL9’. The instances of the feature ‘MIN’ are agglomerated on low values between 0 and 0.15. The observations of the attribute “OVRP” are accumulated on a value of 0. ‘OVRA’ shows a similar distribution, with a few observations on low values. Finally, the attribute of ‘SUP1’ shows an amassment on values around 1 for both classes and the rest being distributed evenly until 0.5, with a few not significant exceptions. Similar distribution takes place on ‘SUP2’ with an obvious intensity on high values.

As far as the correlation between the variables is concerned, the Weka tool offers a very helpful feature of visualizing the correlation matrix. So, the next figure represents the correlation matrix and all these essential linear relationships between the features. In the next figure, the green boxes show a strong positive correlation between the two variables, where the red boxes show a strong negative correlation between the two attributes. The strong positive correlation is observed in the following pairs of attributes:

1) Positive correlation between ‘FOR3’, ‘FOR6’, ‘FOR9’ attributes:
   a. ‘FOR3’ with ‘FOR6’ (very strong positive correlation)
   b. ‘FOR3’ with ‘FOR9’ (strong positive correlation)
   c. ‘FOR6’ with ‘FOR9’ (very strong positive correlation)

The above set of correlations, with a first look, seem to be an understandable and logical outcome because all the attributes relate to each other. By definition (as seen in table 1) the three attributes are forecast of sales for the next 3, 6 and 9 months respectively, so, it is comprehensible that the correlation between the forecast sales the next 3 and 6 months, as well
as the next 6 and 9 months, are very strong, where the correlation between the forecast sales the next 3 and 9 months is considered to be strong.

2) Positive correlation between ‘SAL1’, ‘SAL3’, ‘SAL6’, ‘SAL9’ attributes:
   a. ‘SAL1’ with ‘SAL3’ (very strong positive correlation)
   b. ‘SAL3’ with ‘SAL6’ (very strong positive correlation)
   c. ‘SAL6’ with ‘SAL9’ (very strong positive correlation)
   d. ‘SAL3’ with ‘SAL9’ (strong positive correlation)
The above set of correlations produce a similar outcome compared to the previous set of forecast sales because all the attributes relate to each other, as well. This is because (as seen in table 1), the four features are, in reality, recorded sales quantity for the previous 1, 3, 6, and 9 months. So, there is the same information on the three pairs of features a, b, and c contained on both features in each pair. The same situation is observed in pair d as well, but with lower intensity.

3) Positive correlation between ‘SUP1’, ‘SUP2’ attributes:
   a. ‘SUP1’ with ‘SUP2’ (very strong positive correlation)

The above correlation between the attributes of ‘SUP1’ and ‘SUP2’ (as seen in table 1), indicates the performance of the supplier in the last 1 and 2 months, respectively. As seen from the correlation matrix, there is a very strong positive correlation between these two variables. This is a reasonable outcome if someone thinks that the performance of the supplier, most likely will not change easily. The reliable supplier on the first semester will remain the same on the second one, of course with some exceptions that cannot be entirely attributed to them.

4) Negative correlation between ‘FOR3’, ‘FOR6’, ‘FOR9’ and ‘SAL6’, ‘SAL9’ attributes:
   a. ‘FOR3’ with ‘SAL6’ (strong negative correlation)
   b. ‘FOR3’ with ‘SAL9’ (strong negative correlation)
   c. ‘FOR6’ with ‘SAL6’ (very strong negative correlation)
   d. ‘FOR6’ with ‘SAL9’ (very strong negative correlation)
   e. ‘FOR9’ with ‘SAL6’ (strong negative correlation)
   f. ‘FOR9’ with ‘SAL9’ (strong negative correlation)

The pairs that are mentioned above show a negative correlation characterized as either strong or very strong. It is important to mention that having information about recorded sales is considered a strong asset to be able to forecast future sales. To be more precise, when the sales of a product in the last months is decreasing, the forecast sales show an increase. The previous statement is attributed to all pairs of categories 4, with a very strong negative correlation found on pairs c and d.
3.3 Algorithms evaluation on WEKA

In the next subsections, the results of the executed algorithms using the WEKA tool will be presented.

3.3.1 Presentation of confusion matrices

In this subsection, all the confusion matrices of the executed algorithms with the use of cross-validation will be presented divided into four categories (based on the structure of the algorithm). These confusion matrices, which are explained in section 2.4, consist of indications that are necessary to calculate the evaluation measurements presented in the next subsections. These evaluation measurements are analyzed in section 2.5.

3.3.1.1 Tree-based algorithms

The next table shows the confusion matrices of the executed tree-based algorithms.

<table>
<thead>
<tr>
<th>Table 3 Confusion Matrices of Tree-based algorithms</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>J-48</th>
<th></th>
<th>Simple Cart</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11967</td>
<td>2014</td>
<td>12101</td>
</tr>
<tr>
<td></td>
<td>1563</td>
<td>12418</td>
<td>1659</td>
</tr>
<tr>
<td></td>
<td>2088</td>
<td>11893</td>
<td>12322</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Random Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11664</td>
</tr>
<tr>
<td></td>
<td>2088</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12331</td>
</tr>
<tr>
<td></td>
<td>1209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Decision Stump</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11620</td>
</tr>
<tr>
<td></td>
<td>2472</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Best First Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11978</td>
</tr>
<tr>
<td></td>
<td>1623</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LMT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12145</td>
</tr>
<tr>
<td></td>
<td>1698</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hoeffding Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11870</td>
</tr>
<tr>
<td></td>
<td>2092</td>
</tr>
</tbody>
</table>
The next table shows the confusion matrices of the executed instance-based algorithms. These confusion matrices consist of indications that are necessary to calculate the evaluation measurements presented in the next subsections.

### Table 4 Confusion Matrices of Bayesian algorithms

<table>
<thead>
<tr>
<th></th>
<th>Naïve Bayes</th>
<th>Naïve Bayes Updateable</th>
<th>Bayesian Network</th>
<th>Naïve Bayes Multinomial Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10516</td>
<td>10516</td>
<td>10122</td>
<td>12582</td>
</tr>
<tr>
<td></td>
<td>3465</td>
<td>3465</td>
<td>3859</td>
<td>1399</td>
</tr>
<tr>
<td></td>
<td>1860</td>
<td>1860</td>
<td>1620</td>
<td>12583</td>
</tr>
<tr>
<td></td>
<td>12121</td>
<td>12121</td>
<td>12361</td>
<td>1398</td>
</tr>
</tbody>
</table>

### Table 5 Confusion Matrices of Instance-based algorithms

<table>
<thead>
<tr>
<th></th>
<th>K-NN</th>
<th>K-Star</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11599</td>
<td>10249</td>
</tr>
<tr>
<td></td>
<td>2382</td>
<td>3732</td>
</tr>
<tr>
<td></td>
<td>1819</td>
<td>2697</td>
</tr>
<tr>
<td></td>
<td>12162</td>
<td>11284</td>
</tr>
</tbody>
</table>
3.3.1.3 Neural Network

The next table shows the confusion matrix of the executed neural network algorithm. These confusion matrices consist of indications that are necessary to calculate the evaluation measurements presented in the next subsections.

<table>
<thead>
<tr>
<th>Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multilayer Perceptron</strong></td>
</tr>
<tr>
<td>12130</td>
</tr>
<tr>
<td>1721</td>
</tr>
</tbody>
</table>

*Table 6 Confusion Matrix of Neural Network*

3.3.2 Results of executed algorithms with the use of Cross-validation

In this subsection, the results of the executed algorithms with the use of cross-validation will be presented divided into four categories (based on the structure of the algorithm). More specifically, the next four categories consist of tables that represent the aggregate evaluation for each category separately, resulting from the evaluation measurements (analyzed in Section 2.5) calculated from the confusion matrices depicted in the previous section 3.3.1 and theoretically explained in section 2.4). These tables show the percentages of correctly and incorrectly classified instances for each executed algorithm. In other words, the accuracy is the indicating the ability of the algorithm to correctly classify the instances and the error rate is indicating the inability of the algorithm to correctly classify the instances. It is important to mention that the sum of Accuracy and Error Rate must be equal to 100. Another index that is present on the tables is the Sensitivity, which is indicating how correctly the classifier classified the positive instances in a percentage, which is also known as the True Positive Rate. Finally, another index used in the analysis is the Specificity, which shows how correctly the classifier classified the negative instances (also known as True Negative Rate).
3.3.2.1 Tree-based algorithms

Table 1 shows the results of the correctly and incorrectly classified instances for each tree-based algorithm separately, as well as the True Positive Rate (Sensitivity) and True Negative Rate (Specificity) of the instances. Also, the statistical technique of Cross-validation was applied to the dataset.

<table>
<thead>
<tr>
<th></th>
<th>J-48</th>
<th>Random Tree</th>
<th>Random Forest</th>
<th>Simple Cart</th>
<th>Best First Tree</th>
<th>LMT</th>
<th>Decision Stump</th>
<th>Hoefding Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (Correctly Classified Instances)</td>
<td>87.21%</td>
<td>84.25%</td>
<td>89.78%</td>
<td>87.34%</td>
<td>87.03%</td>
<td>87.36%</td>
<td>82.72%</td>
<td>84.97%</td>
</tr>
<tr>
<td>Error Rate (Incorrectly Classified Instances)</td>
<td>12.79%</td>
<td>15.75%</td>
<td>10.22%</td>
<td>12.66%</td>
<td>12.97%</td>
<td>12.64%</td>
<td>17.28%</td>
<td>15.03%</td>
</tr>
<tr>
<td>Sensitivity - True positive rate (TPR)</td>
<td>85.59%</td>
<td>83.43%</td>
<td>88.20%</td>
<td>86.55%</td>
<td>85.67%</td>
<td>86.87%</td>
<td>83.11%</td>
<td>84.90%</td>
</tr>
<tr>
<td>Specificity - True negative rate (TNR)</td>
<td>88.62%</td>
<td>85.07%</td>
<td>91.35%</td>
<td>88.13%</td>
<td>88.39%</td>
<td>87.85%</td>
<td>82.32%</td>
<td>85.04%</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.873</td>
<td>0.842</td>
<td>0.898</td>
<td>0.873</td>
<td>0.87</td>
<td>0.874</td>
<td>0.827</td>
<td>0.85</td>
</tr>
<tr>
<td>AUC-ROC (Area Under the Curve - Receiver Operating Characteristics)</td>
<td>0.895</td>
<td>0.842</td>
<td>0.959</td>
<td>0.928</td>
<td>0.876</td>
<td>0.935</td>
<td>0.824</td>
<td>0.871</td>
</tr>
</tbody>
</table>

Table 7 Results after the execution of tree-based algorithms with the use of Cross Validation

3.3.2.2 Bayesian algorithms

Table 2 presents the results of the correctly and incorrectly classified instances for each Bayesian algorithm separately, as well as the True Positive Rate (Sensitivity) and True Negative Rate (Specificity) of the instances.

<table>
<thead>
<tr>
<th></th>
<th>Naïve Bayes</th>
<th>Naïve Bayes Updateable</th>
<th>Bayesian Network</th>
<th>Naïve Bayes Multinomial Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (Correctly Classified Instances)</td>
<td>80.96%</td>
<td>80.96%</td>
<td>80.41%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Error Rate (Incorrectly Classified Instances)</td>
<td>19.04%</td>
<td>19.04%</td>
<td>19.59%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Sensitivity - True positive rate (TPR)</td>
<td>75.22%</td>
<td>75.22%</td>
<td>72.40%</td>
<td>89.99%</td>
</tr>
<tr>
<td>Specificity - True negative rate (TNR)</td>
<td>86.70%</td>
<td>86.70%</td>
<td>88.41%</td>
<td>10.00%</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.806</td>
<td>0.809</td>
<td>0.803</td>
<td>0.405</td>
</tr>
<tr>
<td>AUC-ROC (Area Under the Curve - Receiver Operating Characteristics)</td>
<td>0.889</td>
<td>0.889</td>
<td>0.897</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 8 Results after the execution of Bayesian algorithms with the use of Cross-Validation
3.3.2.3 Instance-based algorithms

Table 3 shows the results of the correctly and incorrectly classified instances for each instance-based algorithm separately, as well as the True Positive Rate (Sensitivity) and True Negative Rate (Specificity) of the instances.

<table>
<thead>
<tr>
<th>Instance Based</th>
<th>K-NN</th>
<th>K-STAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (Correctly Classified Instances)</td>
<td>84.98%</td>
<td>77.01%</td>
</tr>
<tr>
<td>Error Rate (Incorrectly Classified Instances)</td>
<td>15.02%</td>
<td>22.99%</td>
</tr>
<tr>
<td>Sensitivity - True positive rate (TPR)</td>
<td>82.96%</td>
<td>73.31%</td>
</tr>
<tr>
<td>Specificity - True negative rate (TNR)</td>
<td>86.99%</td>
<td>80.71%</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.85</td>
<td>0.77</td>
</tr>
<tr>
<td>AUC-ROC (Area Under the Curve - Receiver Operating Characteristics)</td>
<td>0.849</td>
<td>0.849</td>
</tr>
</tbody>
</table>

*Table 9 Results after the execution of Bayesian algorithms with the use of Cross Validation*

3.3.2.4 Neural Network

Table 4 depicts the results of the correctly and incorrectly classified instances for the Multilayer Perceptron, as well as the True Positive Rate (Sensitivity) and True Negative Rate (Specificity) of the instances.

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Multilayer Perceptron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (Correctly Classified Instances)</td>
<td>87.23%</td>
</tr>
<tr>
<td>Error Rate (Incorrectly Classified Instances)</td>
<td>12.77%</td>
</tr>
<tr>
<td>Sensitivity - True positive rate (TPR)</td>
<td>86.76%</td>
</tr>
<tr>
<td>Specificity - True negative rate (TNR)</td>
<td>87.69%</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.872</td>
</tr>
<tr>
<td>AUC-ROC (Area Under the Curve - Receiver Operating Characteristics)</td>
<td>0.938</td>
</tr>
</tbody>
</table>

*Table 10 Results after the execution of Neural Network with the use of Cross Validation*
CHAPTER 4 – Results evaluation and conclusion

4.1 Evaluation of results

In this section, the evaluation of the results that were presented in Chapter 3, will take place. The Tree-based algorithms and the Neural Network show very good results. This is probably due to the plethora of instances that are available on the selected dataset (27,962 rows), as well as the oversupply of the existing attributes (22 columns). These algorithms can describe the response variable (‘BO’), with high precision.

More specifically, it can be seen from the table above, as far as the category of tree-based algorithms is concerned, that ‘Random Forest’, ‘LMT’ and ‘Simple Cart’ show very good results. The Random Forest algorithm takes the first place on both the metric of F-measure as well as the AUC metric, which means that there are, in reality, very low false positives and false negatives (F-Measure), while at the same time the ability of the classifier to distinguish between positive and negative classes is pretty high (AUC-ROC).

The ‘Multilayer Perceptron’ took second place and performed better than most of the algorithms with an F-Measure of 0.874 and an AUC-ROC of 0.938. The Bayesian algorithms didn’t perform well on average. The Bayesian Network with an F-score of 0.872 and AUC-ROC of 0.897.
took fifth place. The instance-based algorithms didn’t produce any remarkable results. ‘K-NN’ and ‘K-Star’ the 11th and 12th place respectively.

Finally, it has to be mentioned that the algorithm “Naïve Bayes Multinomial Text” is used only for text classification, so it cannot be connected in any way with the structure of the dataset that is used in this particular dissertation, the only use is to highlight the difference on the results with other algorithms. An ‘AUC-ROC’ of 0.5 indicates a random result classified by the classifier.

4.2 Conclusion

The statistical method of random under-sampling has undoubtedly improved the accuracy of the results because, without it, the instances of the two labels on products that finally went on backorder or not would be quantitatively uneven (greatly imbalanced). Also, the aforementioned statistical method has decreased the duration of the execution of the algorithm significantly, since the data that are used in the analysis are remarkably minimized. The statistical technique of cross-validation has given a useful overview because it has split the dataset ten times and produced a more reliable result after executing each algorithm ten times, with different train and test sets each time.

The tree-based algorithms of ‘Random Forest’, ‘LMT’ and ‘Simple Cart’ performed better than any other categories, this is probably due to the plethora of instances that are available on the selected dataset, as well as the oversupply of the existing attributes. The Bayesian algorithms perform better than the other three categories when the data modeling takes place in poor data environments. This is because these particular algorithms substitute the lack of information with a series of statistical assumptions (Theodoridis, 2015). As far as the neural network is concerned, it has to be mentioned that “Multilayer Perceptron” performed better than most of the algorithms, although this specific algorithm required a long time to build and execute the model on the dataset.
Bibliography


