NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΊΑΣ ΠΛΗΡΟΦΟΡΙΚΉΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Machine Learning Techniques In Categorical Time Series Analysis Of Manufacturing Process

DIPLOMA PROJECT

ΧΑΡΆΛΑΜΠΟΣ ΜΙΧΑΗΛΊΔΗΣ, ΙΣΙΔΏΡΑ ΧΑΡΆ ΤΟΥΡΝΉ

Supervisor : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Athens, July 2016

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΊΑΣ ΠΛΗΡΟΦΟΡΙΚΉΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Machine Learning Techniques In Categorical Time Series Analysis Of Manufacturing Process

DIPLOMA PROJECT

ΧΑΡΆΛΑΜΠΟΣ ΜΙΧΑΗΛΊΔΗΣ, ΙΣΙΔΏΡΑ ΧΑΡΆ ΤΟΥΡΝΉ

Supervisor : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Approved by the examining committee on the July 20, 2016.

.....	.....	.....
Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Γεώργιος Γκούμας Λέκτορας Ε.Μ.Π.	Κώστας Καρπούζης Αναπληρωτής Ερευνητής Ε.Μ.Π.

Athens, July 2016

Χαράλαμπος Μιχαηλίδης, Ισιδώρα Χαρά Τουρνή
Electrical and Computer Engineer

Abstract

The complexity of modern industrial processes and the constant innovations in production monitoring technologies and data collection, strongly outline the need for advancements in production data analysis. Data Mining is a rapidly growing field, aiming in understanding data and extracting previously unknown information, with the use of Machine Learning techniques, in order to optimize production.

Our cooperation with Johnson & Johnson, enabled us to obtain and explore real case production data. The exploitation of them focused on two distinct goals. The first one was the visualization of the data and the graphical representation of all variables characterizing the mixing process, for an improved data overview. The second one was the Machine Learning algorithms' modification and application on the properly pre-processed production data, to look into the possibilities of these techniques in the enterpise space.

Machine Learning, by definition, sets to represent data as objects in space, utilizing labels and distances between them. In this direction, data were grouped into objects and vectorized with various techniques. Classification and Clustering algorithms were parameterized and implemented, investigating unique attributes of the provided data. Distance calculation methods for each agorithm were examined in depth, and each experiment was assessed "using" different evaluation metrics, in order to examine the performance of the algorithms and the result's accuracy, compared to the initial data. Our conclusions indicate that Machine Learning can drive important business decisions, through process quantification, and further research can be done in each specific case.

Key words

Machine Learning, Variable Length Classification, Categorical Time Series, Manufacturing Process, Transition Matrix

Acknowledgements

As this Thesis Project comes to an end, we remember all the people that helped us make it through and we feel the need to express our gratitude to them, because this would not be possible without their guidance and support.

We would like to thank all members of the Computing Systems Laboratory of National Technical University of Athens, especial Professor Koziris who enabled and trusted us to fulfill such a complex Project, involving many stakeholders. Of course we would like to thank Mr. Konstantinou how was our guide in this trip, supervising us patiently throughout the process. We also can not forget the help of Giannis Giannakopoulos whenever needed, regarding the infrastructure where we ran our experiments.

This Project wouldn't be a reality, and wouldn't be so interesting if it wasn't for Michalis Augoulis who connected us with Johnson & Johnson Hellas, providing us with real data which lead us into facing a real-case scenario. Moreover we deeply appreciate the openness and willingness of Johnson & Johnson Hellas' stuff throughout the process.

In addition to the above, kudos for the crucial support and inspiration goes to Mr. Anagnostopoulos and Mr. Cotsikis from the company Mentat. To Mr. Cotsikis for initially believing in our vision and to Mr. Anagnostopoulos for the truly inspiring sessions we had and his crystal-clear way of addressing and solving complex problems.

Last but certainly not least, this is dedicated to our families and friends, who not only helped us the past months but the last 6 years.

Χαράλαμπος Μιχαηλίδης, Ισιδώρα Χαρά Τουρνή,

Athens, July 20, 2016

This thesis is also available as Technical Report CSD-SW-TR-42-14, National Technical University of Athens, School of Electrical and Computer Engineering, Department of Computer Science, Software Engineering Laboratory, July 2016.

URL: `http://www.softlab.ntua.gr/techrep/`
FTP: `ftp://ftp.softlab.ntua.gr/pub/techrep/`

Contents

Chapter 1

Introduction

1.1 Data Analysis in Manufacturing

Modern industrial systems and processes have become very complex. The information technologies' development, and the application of high - performance computers in all branches of an industrial procedure, result in a vast amount of data being produced continuously. An industrial process is being monitored by different sensors at various times, frequencies and resolutions, thus having an increasing complexity in all of its stages, and standing as a challenging problem for nowadays scientific disciplines. The technology advancements have improved production in all its aspects, yet they have made processes complex and nonlinear anymore, so it is rather hard, or impossible, to gather precise and direct information from measurement equipment only. Classical Analysis methods need to become more advanced, in order to lead the process effectively, and new techniques need to be integrated into old production methods or even replace them. [9]

In this direction, industries adopt and develop data analysis and exploitation strategies, in order to increase production quality, process security and human safety. This approach is not new, as computational and business intelligence ideas are being implemented since early 70s, including the application of artificial neural networks and predictive and adaptive system control, in an attempt to help human operators in a production line, and, if possible, eliminate them. This is where data mining comes to the spotlight, as a way to extract valid, previously unknown and comprehensible information from a process, and use it to make important business decisions. Like schematic drawings and mathematical equations were formerly used, to gain insightful knowledge, monitor, understand and optimize procedures, data analysis methods play a similar role, enhancing and boosting currently applied ones. The multidisciplinary nature of the field, which incorporates, between others, machine learning, image processing and statistics, aims in detecting regularities and patterns, invisible to humans or other analysis' methods.

Industries have the ability to use historical process data, identify relations among discrete process steps and input, as well as the determinants of a procedure's performance, and then optimize the factors that prove to have the greatest effect on production. The same rules apply when real - time data are examined, where more complex techniques should be developed, yet the immediate analysis provides the ability for short- term planning and actions, crucial to the improvement and effectiveness of the procedure. Visualization techniques are a crucial tool for pattern detection and prioritize data collection, as they are widely applied here to enhance the result, with the use of distribution graphs and clustering diagrams.

These needs and possibilities were the starting point of the current thesis, which begun as an attempt to identify production monitoring's problems and systemize processes' data analysis and exploitation.

1.2 Problem Motivation

The multinational production environment of Johnson & Johnson Hellas fits in as an ideal field to experiment and implement all of the above. From the initial conversations, the interest from both

sides regarding the project was high. The field of data analysis is vast and the applications in such an environment are tremendous.

In this plant, each production line consists of many machines, one really important is the vessel in which the mixing takes place. In contrast with other production methodologies, this plant produces its products in a batch manner. This means that in each vessel, a great variety of products can be made, depending on the demand of the market. Mixing is a crucial step of production, because this is the part where the main product is made. As it will be explained in the next chapter in more detail, this vessel is being monitored and controlled by a PLC, of which the raw output is the data-set provided to us.

The first need that was communicated to us, was that of the visualization of the production data. The data were logged on a daily basis, but there was no further analysis and utilization of them. Implementing a graphical depiction of data enabled us to have a better understanding of it and proceed with Machine Learning techniques.

The second one had many sub-goals, which included the assessment of the mixing process and the comparison towards a golden standard, having always in mind the optimization of the production process. Each product has the same very strict quality limits which are easily tracked through chemical analysis of the final product. The part, which is difficult to be measured and evaluated, is the execution of the recipe, while making a certain product. This is particularly interesting, because all the products produced, are within the quality limits, but there are no two identical time-series of actions. To provide an order of magnitude, each batch is completed in around 400 actions.

In order to be able to present a case of evaluating and comparing procedures, we initialized our research with the creation of a classification and clustering method. We believe that this was a great first step of evaluating and experimenting on how data-series could be converted and used in machine learning implementations.

The main challenges, which derived during the process of utilization of the given data-set, were the following:

- **Data Cleansing**

  The provided data-series from the PLC were almost perfect from an integrity point of view. On the contrary, the logged data of each vessel were written by humans, which made them pretty inaccurate.

- **Variable Selection**

  In the data-set, for any given moment we had information from 17 different variables, regarding the state of the vessel, which increased their correlation complexity and made it difficult to come to a meaningful conclusion, just by observing their values.

- **Unequal length series comparison**

  One of the most complex problems we had to overcome was the comparison of unequal length data-series. Each batch consists of actions, represented by their codes (MsgNum) in our data-set. Due to the fact that two same products can be produced by different actions, variating in sequence of actions and in their total number, the comparison could not be done one-to-one.

- **Distance Evaluation**

  Most of the implementations of Machine Learning Algorithms manipulate elements that have a vector of attributes. We needed to decide on the elements, of which the distance we would calculate for each application, and also use the proper metrics for our data.

1.3 Proposed and Implemented Solutions

In orer to tackle the above defined challenges and make best use of the given data, we followed the steps stated below, which form our solution:

- **Pre- processing and Data cleansing**:

  Combining the initial PLC dataset and the human-logged data of each process, and, based on known company internal rules, we grouped separate actions together, to create objects representing an independent procedure that takes place. This was either a Production or Cleaning Process. We decided to only use one of the variables of the data in this direction, the Message Number (MsgNum), as after experiments we arrived to the conclusion that it was the best and simplest approach, which eliminated all complexity deriving from the large number of process variables.

- **Visualization**:

  A visualization tool was implemented, to depict all different variables and their values with time, in order to create a graphic and interactive display of the initial data. This was an approach to better understand and correlate the data, and observe probable visible patterns of the initially unmanageable and incomprehensible consecutive timeseries.

- **Unequal length series comparison**:

  We cast out the time relevance of the actions. This was done, because of the repeatability of them and the fact that many of the actions can be performed in consequent iterations, without interfering with the outcome. Having that in mind, our initial approach was to create a normalized frequency vector for all the messages in a batch. We assumed that this was a valid approach, yet we decided to include more information from the initial dataset. The final approach was to construct a normalized transition matrix for each batch. Each cell of this matrix indicates the probability of transitioning from one message code to another. To state it in a simple manner, cell[A][B] indicates the probability of executing the action B after the action A. This approach turned to be really useful and helped us through the Machine Learning part.

- **Distance Evaluation**:

  A way to calculate distance between different 2D objects (matrices) was designed and implemented, in order to compare the objects we created, and find their relative distance. The first approach was to transform the matrix into a vector, and the second one was to create our own distance method and apply the selected algorithms through that. This was implemented for many known distance metrics (Euclidean, Cosine, KL- Divergence, KS-Test, Infinity Norm) and was further used in the Machine Learning algorithm's application. As our experiments evolved, we observed great differences in the results, depending on the distance method we had chosen.

- **Classification**:

  We experimented with two Classification algorithms on the Production objects created after the Data pre-processing, Nearest Centroid Classifier and k- Nearest Neighbours Classifier. Each Product has different attributes (Product Group, Product Cleaning Group it belongs to), already known from the previous labelling, thus the classification was performed on these characteristics. The distance of the objects, needed for classification, was a variable in each execution. For the k-NN algorithm, the value of k was also a parameter examined. The data were trained and a testing data set was used to identify the performance of the training procedure, which was measured by two metrics, Accuracy and Kappa Coefficient. The simulations gave interesting results, with performance measurements of a maximum 85%.

- **Clustering**:

  We customized the k - Means Clustering algorithm, and based again on the Production objects' different attributes, we assigned them to different clusters. The parameterization of the algorithm focused on two things, the selection of the initial centroids and the proper distance metric between the data objects used for clustering. Two metrics were utilized in the evaluation of the experiments, V-Measure and Rand-Index, and the performance of the simulations proved to be quite poor, with a maximum score of 35%.

1.4 Thesis Structure

In Chapter 2, we define the problem we will work on, with all its different parameters, describe all steps taken towards data processing and understanding, and present the data visualization tool created for the purposes of J&J company from the production data.

In Chapter 3, we focus on the theoritical background needed for our Data Mining and Machine Learning applications. We define the above concepts, and analyze the fields of Classification and Clustering, focusing on the algorithms used in the current work. We also present the concepts of Distances and Evaluation in Machine Learning and metrics and techniques used for these purposes.

In Chapter 4, we present the Implementation of our solutions. Classification an Clustering are analytically parameterized and explained, as are Distance and Evaluation metrics applied in the simulations executed.

In Chapter 5, we present the results and diagrams from the experiments run for all Classification and Clustering algorithms, for different initial setups, and their implementations as describes in Chapter.

Finally, in Chapter 6, we summarize our conclusions from the results, and refer to issues we did not have the chance to work on and which could be regarded as future work.

Chapter 2

Data and Problem Description

In this Chapter, we describe all characteristics of the industry problem we deal with in the current thesis, and the initial steps we took in order to utilize the production data provided to us, in an effective way.
In Section 3.1 , we first describe the mixing process we are focusing on, by giving its most significant points. Then, we present all the parameters in the initial dataset, with a screenshot from it for better understanding, and we shortly define the steps followed in its processing and the main challenges tackled.
In Section 3.2, we describe the pre- processing techniques applied, which are divided in three categories, Flattening, Labelling and Object Creation.
Finally, in Section 3.3, we analyze the way the dataset was depicted graphically and present the visualization tool we created for the purposes of the company.

2.1 Problem Description

In this Section, the main parameters and variables of the problem will be presented.

2.1.1 Case Description

The case we studied is a real-life problem. The data have been provided by Johnson & Johnson Hellas, specifically, the data which were used in the Machine Learning Section, are from a single Vessel which is used in the batch production line of a certain factory in Greece. Each Vessel is used for a specific number of actions and can produce a great variety of products. Please find bellow more details for each element.

- **Vessel**

  This vessel has some automation mechanisms but most of the processes are being executed by the operator. There is no use in pointing exactly which the automated functions are and which are being executed by the user. The total data set, explained, can be found in the next sub-section.

- **Mixing/Production**

  The main function of this vessel is the mixing of different materials in order to produce a certain product. The products vary from lotions to oils and creams. Some standard materials are being provided by a network of fluids and other, more rare, are being imported by hand by the operator.

- **Possible Actions**

  The possible actions which can be performed are: Mixing, Heating, Cooling, Import of Fluids from the network, Import of materials by a trap door, Creation of vacuum pressure. Many of them can be performed at the same time, for example, mixing while heating.

- **Recorded Values**

As it will be described in the following subsection, the actions which were performed, are stored by a PLC as a time-series of messages with the appropriate measurements and time-stamps. There is one special characteristic regarding the format of the output CSV, which is the following.

A set of values is being recorded only at the time of a specific action. For every variable we have two values, one is the actual, measured, value of the respective attribute and the other is the Set-Point of the attribute. Meaning that at the time that the Heating process has started, we have two values of temperature. The one is the measured, ex. $50, 87$ and the other is the goal of the heating process, ex. 75. Those two sets are represented in the initial excel with two different rows. Most of the time it is easy to distinguish them, because Set-Point rows have mostly integer values, in contrast to the measurements, which have decimals.

2.1.2 Dataset Description

This is a really important part of our thesis, which is going to help the reader better understand the data pre-processing section. The data-set which was provided to us, is a raw output of the PLC controlling this particular vessel. The format is CSV and the data, which were used in the Machine Learning part represent the total data from a whole year.

- **Data-Set Information**
  - Format: CSV
  - Time period: $16/02/2015 - 13/02/2016$
  - Rows of Data: 132005
  - Filesize: $25Mb$
  - Attributes (Columns):
    - StateAfter
    - MsgNumber
    - Temp
    - Pressure
    - Agitation1
    - Agitation2
    - Homogen
    - Pump Power
    - Raw Meterials
    - TimeString

- **Screenshot from the initial CSV**

	A	B	C	D	E	F	G	H	I	J
1	StateAfter	MsgNumber	Temp	Pressure	Agitation1	Agitation2	Homogen	Pump Power	Raw Materials	TimeString
2	1	552	78.62	0	14.84028	24.75579	0	0.8101852		16.02.2015 09:41:08
3	1	564	78.62	0	14.84028	24.75579	0	0.8101852		16.02.2015 09:41:08
4	1	553	75	-500	15	30	750	60		16.02.2015 09:41:08
5	1	565	75	-500	15	30	750	60		16.02.2015 09:41:08
6	1	552	78.89	1.15741	14.84028	24.75579	0	0.8391203		16.02.2015 09:43:39
7	1	564	78.89	1.15741	14.84028	24.75579	0	0.8391203		16.02.2015 09:43:39
8	0	553	75	-500	15	30	750	60		16.02.2015 09:43:39
9	0	565	75	-500	15	30	750	60		16.02.2015 09:43:39
10	1	522	50.715	1.15741	6.534722	25.62182	0	0.8101852	21.20999	16.02.2015 10:25:43
11	1	523	75	-500	15	31	750	60	500	16.02.2015 10:25:43
12	0	536	77.1	0	14.83333	24.77691	0	0.8101852		16.02.2015 10:25:43
13	0	562	77.1	0	14.83333	24.77691	0	0.8101852		16.02.2015 10:25:43
14	1	537	75	-500	15	30	750	60		16.02.2015 10:25:43
15	1	563	75	-500	15	30	750	60		16.02.2015 10:25:43

Figure 2.1: Initial Data-Set format in CSV format

- **Explanation of important data-set variables**

  - *StateAfter:* This variable can be either 0 or 1. When the value is 1, it means that the action corresponding to the message number is starting. When it is 0, the process is ending.

  - *MsgNumber:* Through a mapping file we can map the message numbers to specific actions performed in the vessel. For example, some of the messages and their meaning can be seen below.

	A	B
1	500	ΑΝΑΚΥΚΛΟΦΟΡΙΑ ΑΠΌ ΠΑΝΩ
2	501	ΑΝΑΚΥΚΛΟΦΟΡΙΑ ΑΠΌ ΠΑΝΩ (Set Points)
3	504	ΑΔΕΙΑΣΜΑ ΣΕ TNT
4	505	ΑΔΕΙΑΣΜΑ ΣΕ TNT (Set Points)
5	522	ΕΙΣΑΓΩΓΗ ΑΠΙΟΝΙΣΜΕΝΟΥ ΝΕΡΟΥ
6	523	ΕΙΣΑΓΩΓΗ ΑΠΙΟΝΙΣΜΕΝΟΥ ΝΕΡΟΥ (Set points)

Figure 2.2: Example MsgNumbers

  - *TimeString:* This is the time-stamp, down to seconds accuracy, at the beginning or end of each action.

- **Splitting**

  Having a continuous time-series of action messages did not provide us any information regarding the beginning and end of each batch. This was an important step, that enabled us to apply the Machine Learning techniques, as explained in the Implementation Chapter.

  The splitting of the data-set into ”chunks” was done by applying certain rules that were explained to us by senior members of the production management department.

- **Labelling**

  Another really important step of the process was the Labelling of the data-set. In the initial data-set there was no information regarding the product that was being produced at any given moment. In order to label the splitted chunks, two files were provided to us.

  The first one was the logging file of the operators, where each one submitted the end time and the product code of each batch. The second one was a mapping file, which enabled us to retrieve based on the product code, two crucial elements of our analysis, the Product Cleaning Group and the Product Group.

  Those two categorizations will play a major role in both classification and clustering.

  - The Product Cleaning Group consists of 5 groups and all the products belong to one of them. The exact distribution can be seen on Figure 2.3

  - The Product Group is a different categorization which consists of 12 groups, like the previously mentioned, all products belong to one of them. The exact distribution can be seen on Figure 2.4

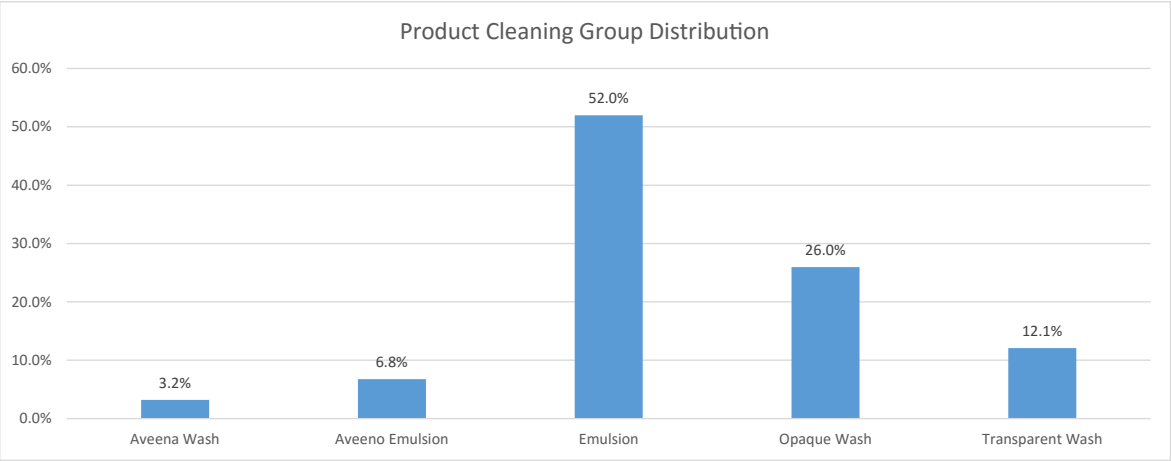

Figure 2.3: Product Cleaning Group Distribution

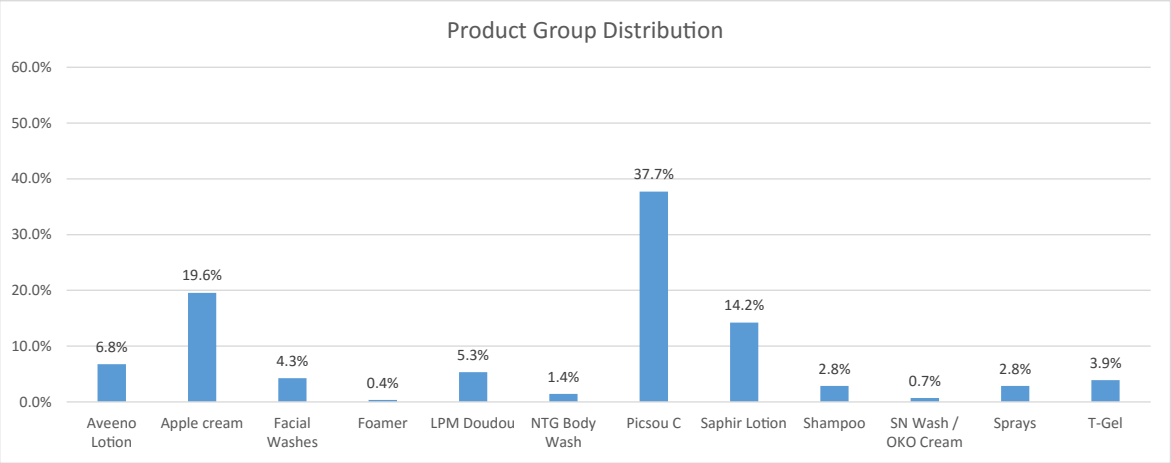

Figure 2.4: Product Group Distribution

2.1.3 Definition of sub-Problems

Two of the most important problems we faced, and tackled, were the following.

- **Variable selection**

  The initial data-set provided us many variables to work with. After some first test,s we decided to move forward utilizing only the message number. In Section 6.2 we mention that a great deal of work can be done in the sector of exploring Machine Learning Techniques using different variables or combination of them.

- **Comparison of non-equal length data-series**

  As one understands, each batch can be produced by different combinations of actions. This is due to the fact that each operator can fine-tune the process according to his beliefs. Each process consists of around 400 messages, the instructions that are being provided by the chemists of the company are not so detailed and the materials used have some tolerance limits. Those factors alow the operator to take initiatives in the production process of each batch.

  This characteristic of the process makes every batch unique, regarding the length of the time-series and the exact order of the actions. As it will become clear in Chapter 4, the compared elements must have same-size attributes in order to be classified or clustered. The idea we

implemented is to construct a transitions matrix for each chunk. This matrix has always size $M \times M$, where M is the number of unique message codes, which is the same for every batch. In our case, $M = 45$.

2.2 Data Pre-Processing

In this section all the methods used for data pre-processing will be analyzed. The role of this part of the thesis is to clean the initial data-set and bring it into the proper format in order to be easier to visualize the data-set as well as to apply the Classification and Clustering.

2.2.1 Flattening - Cleaning

As explained in subsection 2.1.1 the data come in pairs of measurement and Set-Point, so we developed a flattening procedure in order to have one row for each action, containing both all the measurements and all the set-points. Alongside, we cleansed the data from test values that were entered by operators.This part was done in order to help the visualization process.

2.2.2 Labelling

As previously mentioned, Labelling was an important and necessary step, because through the labels we were able to evaluate the performance of both classification and clustering.

The files we used in this step were three: the logging file from the operators, the mapping file between product codes and product attributes , and the chunk series containing all the initial data, splitted into chunks.

The first step was to assign the proper attributes to the chunks found on the log file. This was easy, because we had the unique product code from the log file, and retrieved all the necessary information from the mapping file. The only problem found was the rare case of some typos in the production codes, as found on the log file, which did not enable us to find its attributes. The attributes retrieved were the Production Cleaning Group and the Production Group.

The second step was to assign the products as seen inthe log file with the chunks from the initial data-set. While the task seemed easy to undertake, it proved to be more demanding than expected. The way that this assignment was done, was to find the end time of a certain batch in the log file and then locate, in the data-set, the chunk which had the nearest end time. Three were the main problems in this procedure.

- The inconsistent way the time-stamps were inserted in the log file. This is a process executed by hand which means that there were a lot of typos and non-usable information.
- Because the splitting of the initial data-set was done by non-perfect rules, some chunks were merged together and others split into two.
- The inconsistency between the two lists. In order to make the assignment we implemented an algorithm, which assigned the labels to the nearest chunk. Because many values were found, we decided to use, for the Machine Learning part, only the chunks which had a time difference of less that 7 $hours$, between the end time found in the log and the one found from the data-set.

2.2.3 Object Creation for Machine Learning

The last step of the data pre-processing was to construct a new data-set containing the previously labeled chunks in a usable format. We decided to create the **Chunk class** and make each batch an instance of this class. When all of them were converted, we stored them in a json file, for easier access and to ensure integrity of our procedure.

- **Chunk Class description**

```
class Chunk(object):

  def __init__(self, **entries):
        super(Chunk, self).__init__()

        #The start time of this chunk as found on the initial data-set
        self.start_time = None
        #The end time of this chunk as found on the initial data-set
        self.end_time = None
        #The chunk_type can be either ”Production” or ”Cleaning”
        self.chunk_type = None

        #The product code as found from the log file.
        self.pr_code = None
        #The end time as found from the log file
        self.pr_logged_end_time = None

        #The following attributes were found from the referance file through the
    production code
        self.pr_name = None
        self.pr_group = None
        self.pr_cl_group = None

        #This is used for both classification and clustering
        #This attribute is set to the name of the class/cluster it belongs
        #and is being evaluated at the end of each experiment
        self.cluster = None

        #If this chunk represents the center of a cluster,
        #this variable is set to its name
        self.name = None

        #This is the Transition Matrix of this chunk
        self.TM = None

    #This function is used by create_TM in order to initiate the Transition
    Matrix
    def init_TM(self, num_msgs):
        self.TM = [[0 for _ in range(num_msgs)] for _ in range(num_msgs)]

    #This function creates the Transition Matrix of the chunk
    #Given the message sequence and a message dictionary
    #1)intialises the Transitions Matrix
    #2)for each transition from message A to message B increases by 1 the
    respective cell
    #3)normalises the matri by row
    def create_TM(self, msg_sequence, msgs_dict):
        self.init_TM(len(msgs_dict))
        cur_msg = msg_sequence[0]

        for i in range(len(msg_sequence)-1):
            past_msg = cur_msg
            cur_msg = msg_sequence[i+1]
            self.TM[self.msgs_dict[past_msg]][self.msgs_dict[cur_msg]] += 1

```

```
53          for row_id, row in enumerate(self.TM, 0):
54              row_sum = sum(row)*1.0
55              self.TM[row_id] = [0.0 if row_sum == 0.0 else cell/row_sum for cell
    in row]
```

Listing 2.1: Chunk Class Implementation

- **Chunk List creation and export to JSON**

  Each chunk found in the initial splitted data-set, was converted into a chunk object. The chunks are of two main types ”cleaning chunk” and ”production chunk”. As one can see in the implementation of the chunk class, certain attributes were filled during the creation of each object, and others left blank for further use. One of the most significant attributes of the chunks is the Transition Matrix, which was created from the action message sequence of each chunk, after that was split from the initial data set.

  Once all the message sequences were labeled and converted into chunk objects, we saved that list in order to have a solid data-set for our further experimentation and not to waist time re-running the algorithms. The output was a JSON file, which proved to be very useful later on.

2.3 Data Visualization

In the initial conversation with Johnson & Johnson, it was clearly communicated that a crucial need for them, was the ability to visualize the data from the production process. This task was very difficult and sometimes impossible through the raw output, due to the fact that each batch might be composed from up to 800 rows of data. For this purpose, we created a visualization tool, to graphically display and associate all data variables provided. This tool is already in use and has been proven very helpful in monitoring the production process.

2.3.1 Infrastructure and methodology

The visualization of data variables was implemented in charts with the use of a Javascipt Library. We combined various chart features to create and customize our visualization tool. In order to achieve the visualization of the raw data file, many steps had to be completed.

The steps that the user has to do are, the upload of the CSV file and the beginning of the pre-processing application from the Landing page of the Tool, as seen in the next figures 2.5 . This interface was created with PHP and HTML. From this step on, the process was automated and the user only has to wait a couple of seconds before he is able to see and interact with the chart.

The pre-processing begins with reading the input CSV file, the file is being cleaned from missing values and re-arranged in a more suitable format for the visualization, as explained in flattening Subsection 2.2.1. The previous part was implemented only with the use of Python. After the file is ready in memory, it is stored in a database table. For this part we used the WAMP server, which enabled us to have a mySQL database.

Once everything is ready and stored in the database, we prompt the user to go to the chart page. This page is the end result of our visualization interface, made possible by Highcharts, with the appropriate modifications from the initial templates. You can find more details and screenshots of the tool in the next subsection.

2.3.2 Chart Explanation

- **Description**

Our goal was to visualize all variables in one chart, so that one will be able to view all correlations and dependencies between them, choose which ones to depict in the charts and hide the others, and obtain information for all possible variable combinations. The chart provides the facility of showing or hiding each separate variable, by clicking its name at the bottom of the screen, which then displays or hides both the chart and its respective values' vertical axis. Moreover, the user can zoom in the specific time frame he is interested in, either by typing the exact dates to be drawn in the upper right boxes, or by choosing one of the options in the upper left corner, or, finally, by moving the bar at the bottom of the chart accordingly. Mousing over the chart, at the time points that were in the initial file, there appears a tooltip table, showing the information for the current timestamp, meaning the variables and their values, and also the message with the process description taking place at that specific time. We also added an extra variable, in the form of flags, which is placed at the bottom of the chart and on mouseover displays the message of the process that occurred at that moment. Some screenshots of charts viewable in the tool are presented below.

- **Screenshots**

Below can be found the previously mentioned screenshots, which can help the reader better understand the tool we created for visualizing the data from the production process.

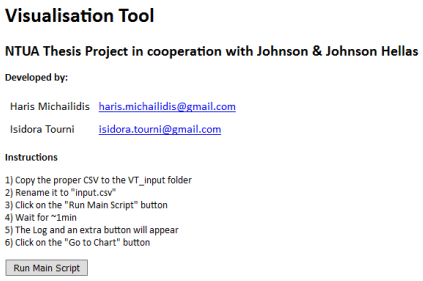

Figure 2.5: Visualization Tool - Landing Page

This screenshot demonstrates the initial page of the tool, where the user can read the instructions. When he has finished placing the file in the correct folder he clicks the ”Run Main Script” button.

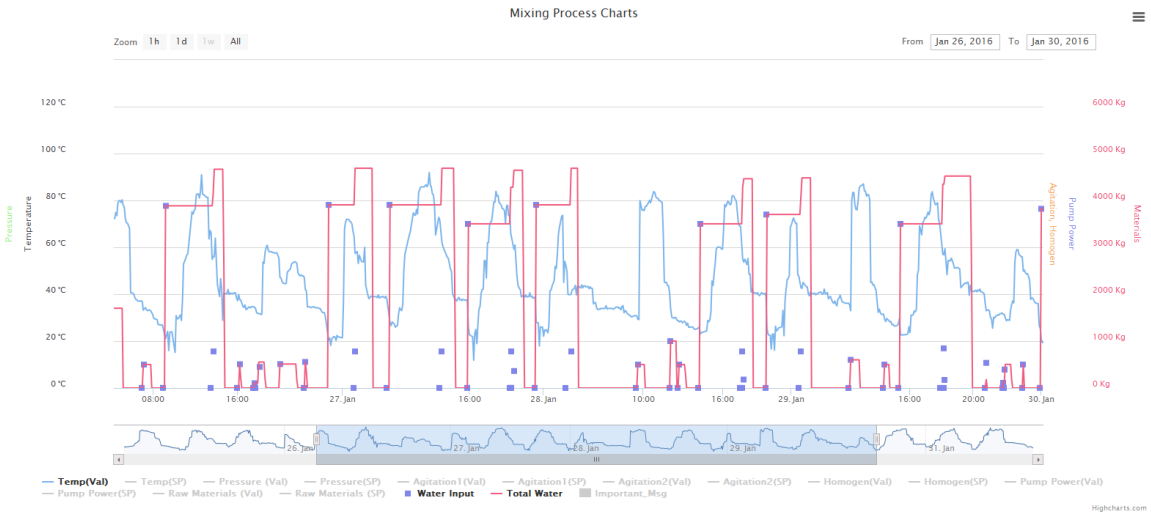

Figure 2.6: Visualization Tool - Temperature, Water - 4 days

In this screen, the user is interested in tracking the temperature and the total water in the vessel. The time window is relatively large, it has an overview of 4 days.

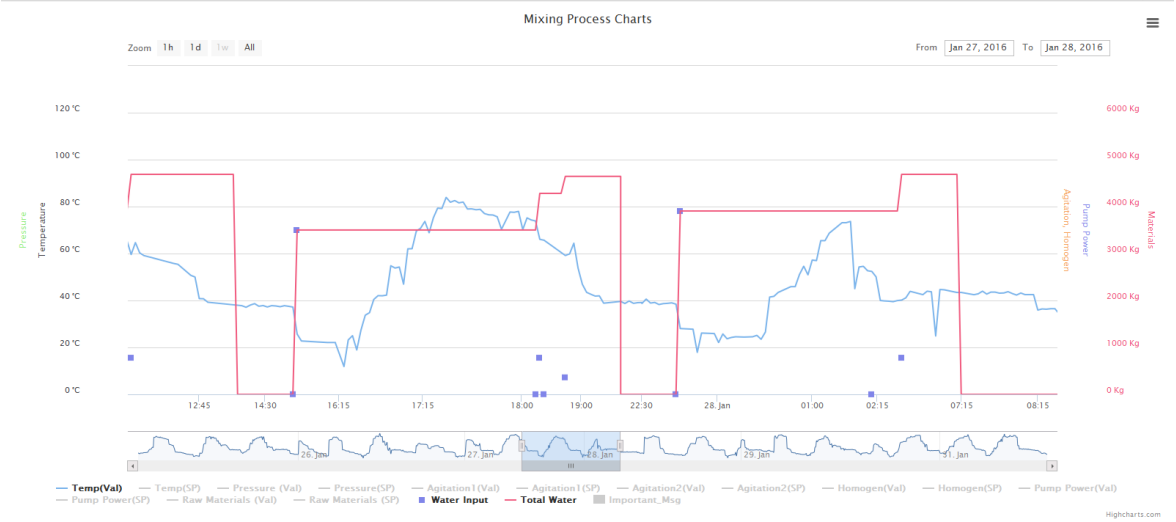

Figure 2.7: Visualization Tool - Temperature, Water - 1 day

This is the same screen as in the previous figure, except for the time window which, in this case, is more narrow and displays only one day of activity.

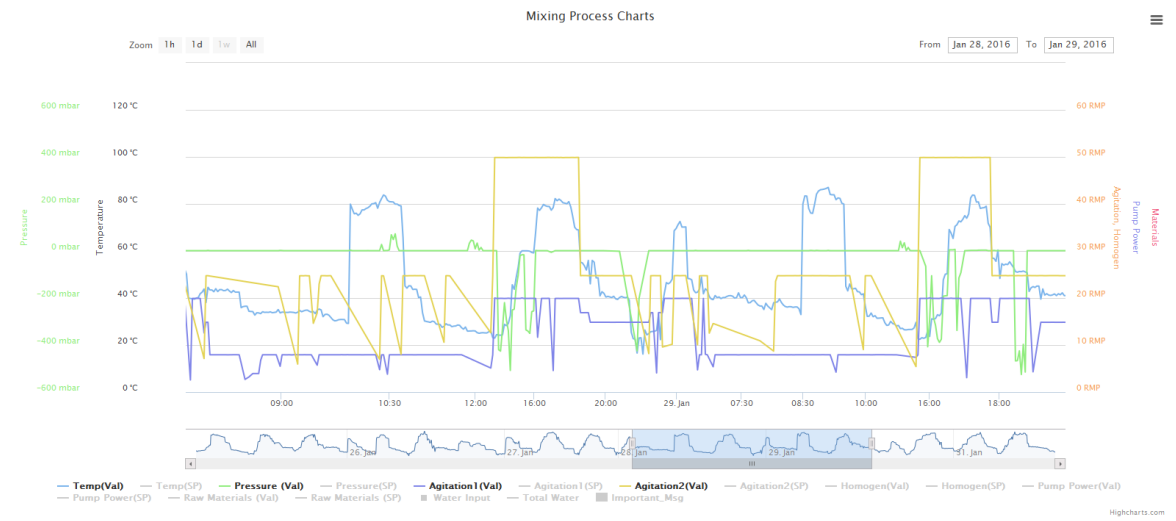

Figure 2.8: Visualization Tool - Four Variables Correlation

In this screen, the user has enabled four separate graphs to be displayed, temperature, Pressure, Agitation 1 and Agitation 2. Through this graph, the user can better understand the relation between those variables during the mixing procedure.

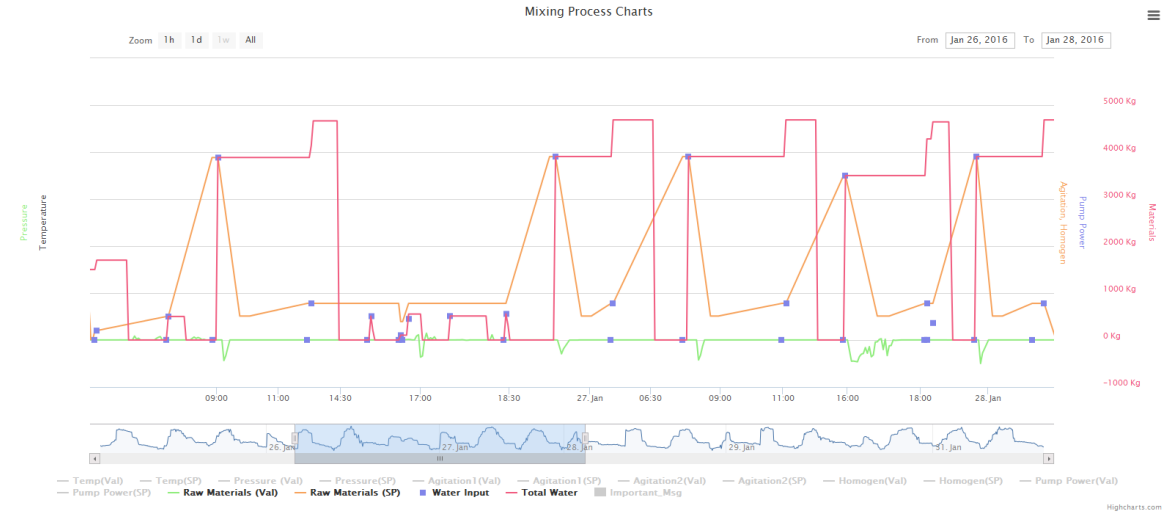

Figure 2.9: Visualization Tool - Raw Materials Addition

One of the most important elements in the mixing process is the water addition. In this graph we can see with purple squares the absolute water addition is each moment. With the red line we observe the sum of all added water, till the moment it is drained from the vessel, where the red line goes to zero.

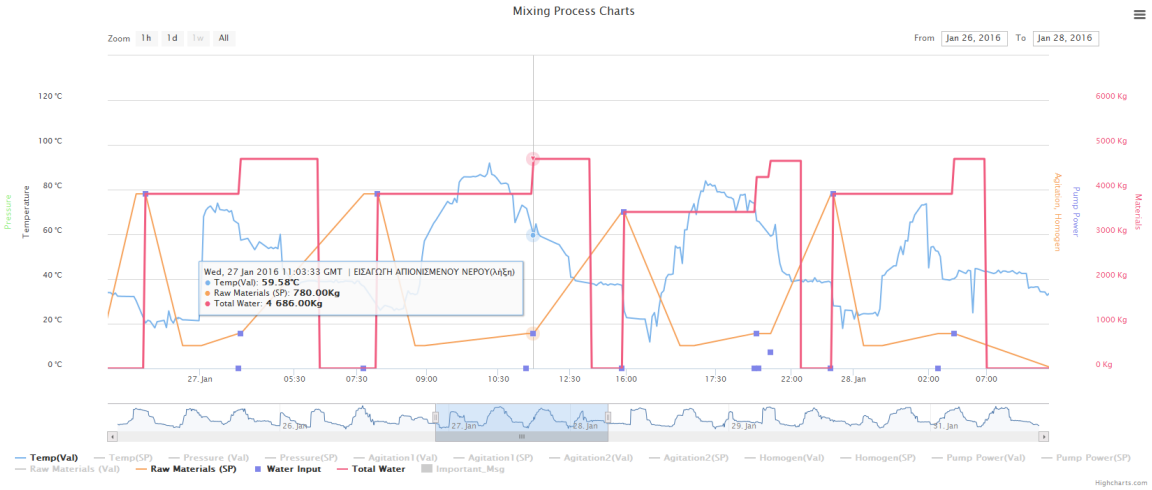

Figure 2.10: Visualization Tool - Action Message Presentation

In order for the user to be able to examine the process in detail, we have implemented a tooltip which includes all the values of the current active variables of the chart for any given moment. The user can hover the mouse over the graph and see the exact values of each variable, along with the action message of that moment.

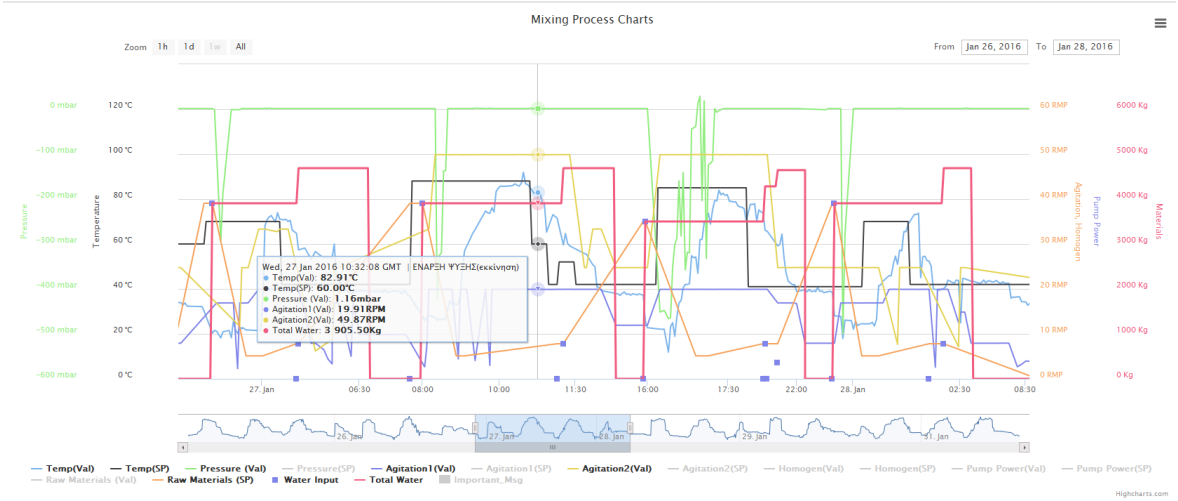

Figure 2.11: Visualization Tool - Variable Values Presentation

As presented in the previous screenshot, in the box that appears aside of the measurements, all the active variables can be displayed, enabling the user to have a crystal clear overview of each moment of the process.

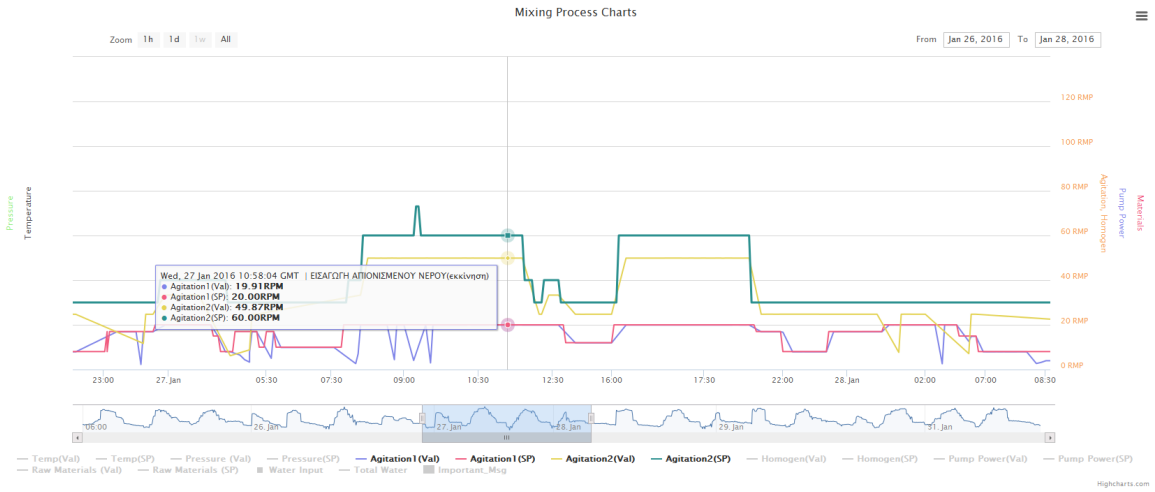

Figure 2.12: Visualization Tool - Values and Set Points

In this screenshot, the concept of set-points is being presented. The set-point is a value, set by the user, which acts as a guide for the process. As one can see, the SP values are always ahead of the measured ones.

Chapter 3

Machine Learning

In this Chapter, we provide the theoritical background of all Machine Lerning techniques applied, and their parameters, and define the concepts that are needed to understand the implementation methods.

In Section 3.1, we introduce the reader to the fields of Data Mining and Machine Learning, and name the different categories they consist of, and the ones we will focus on in the current thesis.

In Section 3.2, we further define Classification Learning and its sub-categories, and analyze the Instance-Based Learning techniques, specifically the k- Nearest Neighbour algorithm, which we apply later on our dataset.

In Section 3.3, we describe the concept of Clustering, naming the groups it is divided into, and dedicating the rest of the section to k- Means algorithm, as a Partitioning Clustering method, which will be further implemented.

In Section 3.4, we explain the role of distance in the previously defined Learning Methods, and analyze some important distance metrics.

Finally, in Section 3.5, we examine some Evaluation approaches and techniques utilized in Machine Learning and theoritically describe the ones which will be further used.

3.1 Data Mining and Machine Learning

[10] [3] The process of discovering patterns in data is widely defined as Data DataMining. It is an automatic or semiautomatic procedure, applied on substantial data quantities, in order to discover desired kinds of combinations and obtain meaningful information. This pattern identification is extremely useful, as it allows us to make nontrivial predictions on new data. [2]

[8] When patterns, that are mined, are represented in terms of a structure, that can be examined, reasoned about and used to inform future decisions, the patterns are called structural, as they are considered to capture the decision structure in an explicit way. Machine Learning is defined as a collection of techniques for finding and describing those structural data patterns, a tool for helping to explain the data and make predictions from it.

Data take then the form of a set of examples or situations, generally characterized as instances, and the output of the process takes the form of predictions and sets of rules about new examples, under given circumstances. So, learning can be considered as having two separate definitions: the aquisition of knowledge and the ability to use it for further purposes. The thing to be learned is defined as concept, and the output produced by a learning scheme is the concept definition.

In Data Mining applications, the learning procedures can be categorized in four different fields:

- **Classification Learning**, where the learning scheme is presented with a set of classified examples, from which it is expected to learn a way of classifying unseen examples
- **Association Learning**, where any association among features is sought, not just ones that predict a particular class value
- **Clustering**, where groups of examples that belong together are sought

- **Numeric Prediction**, where the outcome to be predicted is not a discrete class, but a numeric quantity

In the current thesis, two of the above methods will be thouroughly examined and applied on our data, Classification and Clustering, which are more specific definitions for the two larger categories of Machine Learning: Supervised and Unsupervised Learning.

3.2 Classification

The task of classification covers a wide range of human activity. In its broadest sense, the term involves any decision or forecast made on the basis of currently available information, and a classification procedure is defined as a formal method for repeatedly making such judgments in upcoming situations. In more specific terms, the problem concerns the construction of a procedure that will be applied to a continuing sequence of cases, in which each case must be assigned to a predefined class, depending on observed features or attributes. [1]

Classification is called Supervised Learning, because, in a sense, the scheme operates under supervision, by being provided with the actual outcome for each of the training examples. This outcome is called the class of the example.

Some of the most urgent problems arising in science, industry and commerce, demanding complex and often extensive data, can be regarded as classification or decision problems, such as the preliminary diagnosis of a patient's disease, whilst awaiting definitive test results, in order to select immediate treatment, or the assignment of individuals to credit status on the basis of their financial and personal information.

[5] Supervised learning is one of the tasks most frequently carried out by so-called Intelligent Systems. A large number of techniques have been developed, which are nominally divided in the following fields:

- **Logical and Symbolic techniques**, such as Decision Trees and Learning Rulesets
- **Perception-based techniques**, analyzed in Single Layer Perceptons, Multilayered Perceptons and Radial Basis Function (RBF) Networks
- **Statistics**, which include Naive Bayes Classifiers and Bayesian Networks
- **Instance - Based Learning**
- **Support Vector Machines**

In the next section we will focus on Instance- Based techniques, some of which were applied in the current thesis project.

3.2.1 Instance - Based Learning

[**ApplicationsDataMining**] Instance - Based Learning algorithms are lazy-learning algorithms, as they delay the induction or generalization process, until classification is performed. Once a set of training instances has been memorized, on encountering a new instance, the memory is searched for the training instance that most strongly resembles the new one. In other words, the known instances are being stored and new instances, whose class is unknown, are being related to existing ones. Thus, all the real work is done when the time comes to classify a new instance, rather than when the training set is processed, and so the algorithms require less computation time during the training phase than eager learning algorithms, but more computation time during the classification process.

The above procedure is called the Nearest-Neighbor classification method. The absolute position of the instances within this space is not as significant, as the relative distance between them. Using

a suitable distance method, which ideally minimizes the distance between two similarly classified instances, while maximizing the distance between instances of different classes, the closest existing instance is used to assign the new one to the class. Sometimes, more than one nearest neighbor is used, and the majority class of the closest k - Neighbors (or the distanceweighted average, if the class is numeric) is assigned to the new instance. This is termed the k - Nearest Neighbor method. The selection of k strongly affects the performance of the k - NN algorithm and the result of the classification.

A pseudo-code example for the instance base learning methods is illustrated below. We consider the setup as following: $X : Training\ Data, Y : Class\ Labels\ of\ X, x : Unknown\ sample$

k Nearest Neighbours(X,Y,x)
for i=1 to m **do**
  Compute Distance **d**(X_i,x)
end for
compute set I containing indices for the k smallest distances d(X_i,x).
return majority label for Y_i where i ε I

3.3 Clustering

Clustering techniques apply when the instances are to be divided into natural groups and there is no specified class to be predicted. These clusters presumably reflect a mechanism that causes some instances to bear a stronger resemblance to each other, than they do to the remaining ones. They are considered to be unknown and are inferred from the data, that is why Clustering is is also known as Unsupervised Learning.

The groups that are identified may belong to one of the following different categories, based on the nature of the mechanisms that are thought to underlie in the particular clustering phenomenon:

- **Exclusive**, meaning any instance belongs to only one group
- **Overlapping**, as an instance may fall into several groups
- **Probabilistic**, because an instance may belong to each group with a certain probability
- **Hierarchical**, as a rough division of instances into groups at the top level and each group refined further, perhaps all the way down to individual instances.

However, because these mechanisms are rarely known, as the very existence of clusters is, after all, something that we're trying to discover, the characterisation of them is usually dictated by the clustering tools at our disposal.

A few real examples of the use of clustering involve dividing customers into homogeneous groups as a marketing focused procedure, a weather data collection and analysis for finding new insights into climatological and environmental trends, and bioinformatics' identification of groups of genes with similar patterns of expression, to determine which genes are responsible for specific hereditary diseases.

Due to the fact that the notion of cluster is not precisely defined, many clustering methods have been developed, each of which uses a different induction principle. They are divided into five main groups, as analyzed below:

- **Hierarchical Methods**, which construct the clusters by recursively partitioning the instances in either a top-down or bottom-up fashion. They can be sub- divided in Agglomerative hierarchical clustering and Divisive hierarchical clustering

- **Partitioning Methods**, which relocate instances by moving them from one cluster to another, starting from an initial partitioning. Types of partitioning methods involve Error Minimization Algorithms and Graph - Theoritic Clustering. The simplest algorithm, employing a squared error criterion is the K-means algorithm, which will be further analyzed as it was used on the current problem.

- **Density - Based Methods**, which assume that the points that belong to each cluster are drawn from a specific probability distribution

- **Model - Based Clustering Methods**, which attempt to optimize the fit between the given data and some mathematical models. The most frequently used methods in this category are Decision trees and Neural Networks

- **Grid - Based Methods**, which partition the space into a finite number of cells that form a grid structure, on which all of the operations for clustering are performed

3.3.1 Partitioning Methods- K - Means algorithm

Partitioning clustering methods, as mentioned, partition the data object set into clusters where every pair of object clusters is either distinct (hard clustering) or has some members in common (soft clustering).

The classic and most common technique is called k - Means. It can be also characterized as an Iterative Distance-Based Clustering method. Applying this algorithm, we first specify in advance the parameter k, which represents how many clusters are being sought. Then k points are chosen at random as cluster centers. We also define a maximum number of iterations for the algorithm to run, over which the process is terminated. All instances are assigned to their closest cluster center, according to the ordinary Euclidean distance metric. Next the centroid, or mean, of the instances in each cluster is calculated, and these centroids are considered to be new center values for their respective clusters. Finally, the whole process is repeated with the new cluster centers. Iteration continues until the same points are assigned to each cluster in consecutive rounds, at which stage the cluster centers have stabilized and will remain the same forever. If maximum number of iterations has been reached in the meantime, the algorithm is terminated.

A pseudocode of the above description is presented below, where the setup is the following: $S : Instance\ set,\ k : Number\ of\ Clusters$

k - Means(S, k)
Initialize k cluster centers
while terminatioon condition is not satisfied **do**
  assign instances to the closest cluster center
  update cluster centers based on the assignment
end while
return clusters

This clustering method is simple and effective. It is easy to prove that choosing the cluster center to be the centroid minimizes the total squared distance from each of the cluster's points to its center. Once the iteration has stabilized, each point is assigned to its nearest cluster center, so the overall effect is to minimize the total squared distance from all points to their cluster centers. However, one should take into account that this minimum is a local one, there is no guarantee that it is the global minimum. To increase the chance of finding a global minimum, we often run the algorithm several times with different initial choices and choose the best final result.

3.4 Distances

[7] The notion of distance is the most important basis for Machine Learning, both Supervised and Unsupervised. In the first category, standard distances often do not lead to appropriate results, while in the second one, the calculation of means of objects of known groups is not always a valid method for the correct algorithms' application. By definition, the choice of the distance measure determines whether two objects naturally go together and, therefore, the right choice of the distance measure is one of the most decisive steps for the determination of learning properties. The distance should not only adequately represent the relevant scaling of the data, but also the study target, to obtain interpretable results. Some of the most widely used distance metrics, which were also implemented in the current thesis, are analyzed further below. [6]

3.4.1 Minkowski Metric

The Minkowski metric or L_q norm calculates the distance d between the two objects x and y by comparing the values of their n features. The Minkowski metric, as given in equation 3.1 , can be applied to frequency, probability and binary values.

$$d(x,y) = L_q(x,y) = \sqrt[q]{\sum_{i=1}^{n} \mid x_i - y_i \mid^q} \tag{3.1}$$

The most important special case of the Minkowski metric is for q=2, the **Euclidean distance** or L_2 norm:

$$d(x,y) = L(x,y) = \sqrt{\sum_{i=1}^{n} \mid x_i - y_i \mid^2} \tag{3.2}$$

3.4.2 Cosine Distance

The cosine similarity (or Orchini similarity, angular similarity, normalized dot product) is a similarity on R^n, defined by

$$cos(a) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \tag{3.3}$$

where a is the angle between vectors x and y. In the binary case, it is called the Ochiai- Otsuka similarity. The cosine distance is defined as

$$d(x,y) = 1 - cosa$$

3.4.3 Kullback- Leibler Divergence

The Kullback-Leibler divergence (KL) or relative entropy is a measure, calculated from information theory, which determines the inefficiency of assuming a model distribution, given the true distribution. It is generally used for x and y representing probability mass functions. The equation for its calculation is

$$d(x,y) = D(x \parallel y) = \sum_{i=1}^{n} x_i \star \log \frac{x_i}{y_i} \tag{3.4}$$

3.4.4 Kolmogorov- Smirnov Test

The Kolmogorov Smirnov metric (or Kolmogorov metric, uniform metric) is a metric on probability space P, defined by

$$d(x,y) = \sup_{(x,y)\epsilon R} \mid x - y \mid \tag{3.5}$$

considering again that x, y are different distribution functions. It is used in statistics as measure of goodness of fit.

3.4.5 Infinite Norm

The Infinite (or Uniform or Sup) Norm is the $L_(\infty)$ metric on the set $C_{[a,b]}$ of all real or complex continuous functions on a given segment [a,b]. For vectors x, y it is defined by

$$d(x,y) = \lim_{p\to\infty} \sqrt[q]{\sum_{i=1}^{n} \mid x_i - y_i \mid^q} \tag{3.6}$$

3.5 Evaluation

Evaluation is of crucial value in Data Mining, as the assessment of the application of our methods and the results of our techniques are the key to further conclusions, optimizations and progress. In this section, the concept is analyzed for both Classification and Clustering.

3.5.1 Classification Evaluation

The success rate of the classification learning is usually judged on the test data, for which the true classes are known, and is evaluated using different metrics, giving an objective measure of how well the concept has been learned by the data. Here, two different evaluation methods are being analyzed, Accuracy and Cohen's Kappa, which will be applied in Chapter 5 in the simulations' results.

Another major issue on the classifier's evaluation is its speed. A classifier that is 90% accurate may be preferred over one that is 95% accurate if it is 100 times faster in testing (and such differences in time-scales are not uncommon in neural networks, for example). This parameter is also examined in the experiments presented later in the current thesis.

Accuracy

The reliability of the classification result is represented by the proportion of correct classifications.

Usually, it is the accuracy on the unseen data, when the true classification is unknown, that is of practical importance. The generally accepted method for estimating this is to use the given data, in which we assume that all class memberships are known. Firstly, we use a substantial proportion (the training set) of the given data to train the procedure. This rule is then tested on the remaining data (the test set), and the results compared with the known classifications. The proportion correct in the test set is an unbiased estimate of the accuracy of the rule provided that the training set is randomly sampled from the given data.

There is a slight loss of efficiency here, as we do not use the full sample to train the decision rule, but with very large datasets this is not a major problem.

	1	2	Total
1	p_{11}	p_{12}	p_{1r}
2	p_{21}	p_{22}	p_{2r}
Total	p_{1c}	p_{2c}	1

Table 3.1: Confusion Matrix after Classification

Cohen's Kappa (Kappa coefficient)

[11] The Kappa statistic is a metric that compares an Observed Accuracy with an Expected Accuracy, and is used to evaluate a single classifier or classifiers amongst themselves. It takes into account random chance (agreement with a random classifier), which generally means it is less misleading than simply using accuracy as a metric. Computation of Observed Accuracy and Expected Accuracy is integral to comprehension of the kappa statistic, and is most easily illustrated through use of a confusion matrix.

For a two dimensional confusion matrix, as the previous one, the Kappa metric is equal to

$$k = \frac{p_0 - p_e}{1 - p_e} \tag{3.7}$$

where the Observed Accuracy is

$$p_0 = p_{11} + p_{22}$$

and the Expected Accuracy is

$$p_e = p_{1c}p_{1r} + p_{2c}p_{2r}$$

.

Kappa is always less than or equal to 1. A value of 1 implies perfect agreement and values less than 1 simply less than perfect agreement. In rare situations, Kappa can be negative. This is a sign that the two observers agreed less than would be expected just by chance.

The values and according interpretations of Kappa are summarized below:

- Poor agreement = Less than 0,20
- Fair agreement = 0,20 to 0,40
- Moderate agreement = 0,40 to 0,60
- Good agreement = 0,60 to 0,80
- Very good agreement = 0,80 to 1,00

3.5.2 Clustering Evaluation

Clustering evaluation demands an independent and reliable metric for the assessment and comparison of clustering experiments and results. In theory, the clustering researcher has acquired an intuition for the clustering evaluation, but in practice the mass of data, on the one hand, and the subtle details of data representation and clustering algorithms, on the other hand, make this random judgement impossible. An intuitive, introspective evaluation can only be plausible for small sets of objects, whilst large-scale experiments require an objective method. There is no absolute scheme for the desired assessment, but a variety of evaluation measures from diverse areas such as theoretical statistics, machine vision and web-page clustering that can be applied.

The theoritical definition of various clustering evaluation metrics, which were used during the thesis's experiments, is provided.

V-measure

[4] V-measure is an entropy-based metric, which measures how successfully the criteria of homogeneity and completeness have been satisfied. It is computed as the harmonic mean of distinct homogeneity and completeness scores.

A clustering result satisfies homogeneity if all of its clusters contain only data points, which are members of a single class. A clustering result satisfies completeness if all the data points, that are members of a given class, are elements of the same cluster. The homogeneity and completeness of a clustering solution run roughly in opposition: Increasing the homogeneity of a clustering solution often results in decreasing its completeness. More specifically:

- Homogeneity: In order to satisfy our homogeneity criteria, clustering must assign only those datapoints that are members of a single class to a single cluster. That is, the class distribution within each cluster should be skewed to a single class, having zero entropy. We determine how close a given clustering is to this ideal by examining the conditional entropy of the class distribution given, the proposed clustering.
- Completeness: Completeness is symmetrical to homogeneity. In order to satisfy the completeness criterion, a clustering must assign all of those datapoints that are members of a single class to a single cluster. To evaluate completeness, we examine the distribution of cluster assignments within each class. In a perfectly complete clustering solution, each of these distributions will be completely skewed to a single cluster.

So, V- Measure is given by the following equation

$$V_b = \frac{(1+b) \star h \star c}{(b \star h) \star c}$$

The calculations of homogeneity, completeness and V-measure are completely independent of the number of classes, the number of clusters, the size of the data set and the clustering algorithm used. Thus, these measures can be applied to and compared across any clustering solution, regardless of the number of data points, the number of classes or the number of clusters

Rand - Index

The Rand - Index is a simple criterion used to compare an induced clustering structure (C_1) with a given clustering structure (C_2). Let

- **a** be the number of pairs of instances that are assigned to the same cluster in C_1 and in the same cluster in C_2
- **b** be the number of pairs of instances that are in the same cluster in C_1, but not in the same cluster in C_2
- **c** be the number of pairs of instances that are in the same cluster in C_2, but not in the same cluster in C_1
- **d** be the number of pairs of instances that are assigned to different clusters in C_1 and C_2.

The quantities a and d can be interpreted as agreements, and b and c as disagreements. The Rand Index is defined as:

$$RAND = \frac{a+d}{a+b+c+d}$$

The Rand Index lies between 0 and 1. When the two partitions agree perfectly, Rand Index is 1.

Chapter 4

Implementation

In this Chapter we present the implementation of all the concepts analyzed in Chapter 3 for our experiments, meaning all algorithms and their parameterization.

In Section 4.1, we present the two algorithms used for Classification, Nearest Centroid Classifier and k-Nearest Neighbours Classifier, and the important variables for their execution.

In Section 4.2, we focus on the Clustering procedure, giving a detailed description of the k-Means algorithm used for this purpose.

In Section 4.3, we describe the different distance calculation methods, for all simulations, and present all ways they were implemented and evaluated in both Classification and Clustering.

Finally, in Section 4.4, we explain in detail the way the results were evaluated and the metrics used for this purpose.

4.1 Classification

The objective of our approach was to classify the production chunk objects according to some of their attributes. Specifically, we wanted to categorize them based on two separate parameters, Product Cleaning Group, and Product Group, which are both attributes of each object. The Product Cleaning Group has 5 instances, and the Product Group has 12.

We also experimented with the classification algorithms for the Product Code attribute of each object, but the large number of separate production codes and therefore the small number of elements assigned to each class, resulted in classifier's overfitting. It was not possible to correctly train and test our data in a meaningful way, and the outcome of the process provided no more information than the existing about products' classes.

For the classification, two different algorithms were used, as analyzed previously: a custom Nearest Centroid algorithm and the known well documented k-Nearest Neighbours Algorithm.

Both algorithms are parameterized as following:

- **Classifying attribute:** both algorithms were implemented for two attributes, Product Cleaning Group and Product Group,

- **Split percentage:** We divided the initial product data into two groups, one training set, on which we calculated the separate classes' centers and applied the algorithms to produce our result, and one testing set, which we classified based on the training procedure's derived classes. The training and testing split percentages selected were 80% -20%, 65% -35% and 50% -50% respectively for both algorithms.

- **Distances:** We used the defined distances functions to calculate the distance between the transition matrices of each chunk object from either the class' centers or the nearest chunks, depending on the implementation.

4.1.1 Nearest Centroid Classifier

For our list of chunks with "Production" attribute only, we followed the steps mentioned below, to run the experiments of classification: Selecting the attribute on which we would classify the objects, which was either Product Cleaning Group or Product Group, we initially split the data based on the three training and test dataset percentages, as mentioned in parameterization before. We use the training data to calculate the centers of the algorithm. Focusing on the Transition Matrices of each object in this set, and examining the value of the classification attribute of the object, we found the average transition matrix of all the objects belonging to each instance. This matrix is considered to be the center of the class. Then, for each element in this class, from the initial test set, we calculated its distance from the different centers. Finally we assigned the test element to the class of which the center was nearest.

Depending on the experiment, different methods of distance algorithms were used.

Due to uneven sized classes, each experiment was executed for 30 iterations, meaning for 30 different sets of data, of the same split percentage. This enabled us to validate our results and to be sure of the integrity of our data.

For a pseudocode implementation of the above, please see A.1.1.

4.1.2 K - Nearest Neighbors Classifier

K - Nearest Neighbors is a well-known classification algorithm. Although many implementations exist, we decided to implement our own, in order to have total control over the variables and its run-time execution. The steps of the execution are the following and the methodology is relatively close to the one of Nearest Centroid Classifier.

The first step was to choose the attribute for which we want to run the classifier, this can be either Product Cleaning Group or Product Group. Following, we split the data-set into training and test, for three different percentages, as in Nearest Centroid Classifier. For each iteration, 30 in total, we split the data-set into random sets, of specific percentage, in order to compare the integrity of the results. Having selected the distance method, we calculate the distance of each chunk object in the test set with all of the chunk objects in the training set. After that, we sorted the values for each test chunk in ascending order. The principal of K-NN algorithm is that each element of the test set is classified to the class that the majority of its k-Nearest Neighbors belongs to. Once we had created a sorted list of all the neighbors, it was easy to evaluate the classification for each value of k, within the desired limits.

At this point we have classified all the chunk objects of the test set for every k in our range, for a specific distance method. In Chapter 5 you can see the impact of k in the evaluation of k-NN. We repeated the process for all the distance methods that concerned us.

For a pseudocode implementation of the above, please see A.1.2.

4.2 Clustering

The purpose of our clustering implementation is to assign our product chunks data into clusters based on some of their attributes. In this direction, we implemented a custom k - Means algorithm, analyzed as following:

4.2.1 K - Means Implementation

- **Clustering parameter k:** The number of clusters that the data is expected to be assigned to is initialized as the number of values in each of the attributes we performed the clustering on. Thus, for Product Cleaning Group, we have k=5 and for Product Group we have k=12.

- **Initial Centroids:** The centroid is represented as a transition matrix of a chunk class object. The experiments were held with a variety of initial centroids, both random and specific, so as to test the performance and accuracy of the algorithm. In the first scenario, each centroid was

picked to be a random transition table from one of the products of the input data, whilst in the second one, we initialized the centroids choosing objects with the clustering attribute in specific ranges or even of a certain value. More specifically, centroids were chosen to be the matrices of products either all in the same Production Cleaning Group (or Product Group, according to the clustering attribute), or all in different Production Cleaning Group/ Product Group, or, finally, all in random Production Cleaning Group/ Product Group. Each of these 3 separate experiments was executed 100 times to eliminate variation in our selections.

- **Average:** In each repetition of the algorithm, a new centroid of each cluster was calculated, based on its currently assigned members. In our implementation, this is represented by the average transition matrix of all transition matrices of the product chunks in this cluster.

For a pseudocode implementation of the above, please see A.2.1.

4.3 Distances

The processes defined before, classification and clustering, and the algorithms implemented in both, require a distance calculation between the different elements. In our problem, considering the procedures are based on production messages, distance is calculated as the distance between the two dimensional transition matrices of two separate product chunks. Different distance algorithms were parameterized and applied in the experiments, so as to examine the performance and accuracy of each implementation. We used the following methods:

- Euclidean distance
- Cosine Distance
- Kullback - Leibler Divergence
- Kolmogorov - Smirnov Test
- Infinity Norm

Most of them, by definition, apply only on one-dimensional matrices (vectors), thus an important step was the correct transformation of the two dimensional matrices into vectors, in order to calculate the distances. Our approaches can be divided in three categories:

- **Average rows:** the distance algorithm is applied between the same rows of the matrices and the result was the average of all scores calculated.
- **Vector:** each row is appended to the first one, thus creating a $1xN$ vector, on which the distance method is applied.
- **Diagonal:** for $j > i$, the average of elements $[i, j]$ and $[j, i]$ is calculated and considered as one, thus creating an upper triangular matrix. Then, with the previous Flatten method, each row is appended to the first, for the distance method to be applied on.

The exact implementation of the previous methods can be found in the subsection 4.3.1 .
The total list of the implemented distance methods is the following:

- **Euclidean distance**
  - Euclidean distance between each corresponding cell
  - Average of the Euclidean distances between corresponding rows

  - Average of the Euclidean distances between corresponding columns

- **Cosine distance**
  - Average of the cosine distance between corresponding rows
  - Cosine distance of the created vectors using the Vector transformation
  - Cosine distance of the created vectors using the Diagonal transformation

- **Kullback–Leibler Divergence**
  - Average of the KL Divergence between corresponding rows
  - KL Divergence of the created vectors using the Vector transformation
  - KL Divergence of the created vectors using the Diagonal transformation

- **Kolmogorov–Smirnov test**
  - Average of the KS test score between corresponding rows
  - KS test score of the created vectors using the Vector transformation
  - KS test score of the created vectors using the Diagonal transformation

- **Infinity Norm**
  - Infinity Norm of the difference of the two matrices

For the implementation of the above mentioned distance algorithms, the Python SciKit-Learn Library was used. This library provides many useful distance calculation functions but most of them are implemented for vectors.

4.3.1 Implementation of Distance Methods

Two of the functions that convert a $2D$ matrix into a $Vector$, can be found in a pseudocode implementation at A.3.1.

The different implementations of the distance algorithms, as described in the beginning of this Section, can be found at A.3.2.

4.3.2 Evaluation of Distance Methods

In order to test all the different distance evaluation methods, we implemented the following script. We were interested in evaluating the performance of all distance methods under different variation of the variables in question, for both classification and clustering cases. The reason we chose to run different evaluation tests on classification and clustering is that we wanted to understand the impact of the distance method in each case.

- Distance Methods Evaluation for Classification

  We took the decision to only run the following experiments with the Nearest Centroid Classifier for the 80% train split. The classifier run for both Product Cleaning Group and Product Group for 30 iterations, as described in previous experiments, to ensure integrity of the results. For each iteration, we randomly split the data-set in 80%-20%, training-testing respectively, and calculated the centers of each instance, either 5 or 12 classes. After selecting a distance method, we assigned the chunk objects to their nearest center, based on the selected distance algorithm.

  A pseudocode implementation of the above algorithm can be found at A.3.3. The results of this implementation can be found in Section 5.1.2.

- Distance Methods Evaluation for Clustering

  In the clustering case, the only tested algorithm is K - Means, thus we evaluated all distance methods on K - Means algorithms. A pseudocode implementation of the above algorithm can be found at A.3.4.

4.4 Outcome Evaluation

All of the above can only produce a usable result if they can be evaluated. In early experiments we used the Confusion Matrix of the outcome, which enabled us to have a more direct overview of the results. As the experiments evolved, we had to evaluate each outcome with a single score. The need for evaluation scores became more clear and we had to decide which scores to choose between the many evaluation methods that exist, for both classification and clustering. We used two different evaluation methods for each case. The main difference on evaluating Classification and Clustering results is that in the first case one knows the correct answer, because all data are labeled from the beginning. In the case of Clustering, however, one does not know the correct value of the data and has to evaluate the outcome based on different factors.

4.4.1 Implementation of Classification Evaluations

For the evaluation of the Classification Results we used the following methods:

- **Accuracy**

  Accuracy is defined as the ration between the correctly classified items and the total number of classified items. In our case, the correctly classified chunk objects were the ones that the cluster's name was the same as the initial label of the respective attribute. The attributes of the classification were the Production Cleaning Group and the Production Group, as analyzed before.

$$\text{Accuracy} = \frac{\text{Number of Correctly Classified Elements}}{\text{Total Number of Classified Elements}}$$

- Kappa

  The Kappa algorithm is part of the SciKit-Learn Laboratory Library and specifically, of the Metrics Module. SciKit is a well known Library for Python and has proved to be very helpful in our thesis project.

  Source: SciKit-Learn Laboratory -> Metrics -> Kappa

4.4.2 Implementation of Clustering Evaluations

For the evaluation of the Clustering Results we used the following methods. Although in other clustering evaluation scenario one might not had the labels, in this case we had all our data labelled. Both algorithms require ground truth class labels, which are used as reference in order to evaluate the predicted.

- V Measure

  Source: SciKit-Learn -> Metrics -> V Measure Score

- Rand Index

  Source: SciKit-Learn -> Metrics -> Adjusted Rand Score

Chapter 5

Results

In this chapter, we present the results from the application of classification and clustering algorithms on our data set. All algorithms, as described in Chapters 4 and 5, were executed for multiple parameters, thus providing different results and conclusions for each settings' combination.

In Section 6.1, we present the classification results, divided into three categories. At first, we compare both Nearest Centroid classifier and k - Nearest Neighbors classifier, for different training and test data splits, keeping the distance parameter fixed. Next on, we focus exclusively on the Nearest Centroid algorithm, for a specific data split of 80% - 20% and evaluate the classifier's performance for all different chunk distances. Finally, we examine the application of k - Nearest Neighbors classifier for different values of k and assess the outcome.

In Section 6.2, the clustering results are presented and evaluated. We used k - Means algorithm for clustering the chunk objects product data into different classes, based again on two attributes, Product Cleaning Group and Product Group. The algorithm's input data are the chunk objects. The centroids and distances vary in each execution, so that we can evaluate the clustering result for different problem parameterizations, using two different metrics, V - Measure and Rand Index.

5.1 Classification

The classification was performed for two distinct attributes, Product Cleaning Group and Product Group.

The algorithms tested were:

- Nearest Centroid Classifier
- K - Nearest Neighbors Classifier

The distance algorithm was a variable in all cases.

For each run of every algorithm, a confusion matrix was produced. Its rows reflect the classifier 's results, meaning the number of products per label that were classified as being in the column's label instance after running the algorithm. Its columns represent the ground truth, the actual values of each instance to be classified.

The sum of each column is the number of products classified as instances of this label, while the sum of each row represents the true number of this label's products in our initial sample. The sum of the diagonal of the matrix is the number of the correctly classified products.

The final confusion matrix of each run is further used to calculate the performance for each of the parameters considered, as mentioned above.

All data used were split into a training and a testing sample, which varies, so as to better examine the accuracy of the classifiers predicting the correct class for the test set, regarding the initial data they were trained on.

5.1.1 Baseline

A baseline result was required, in order to compare the classification results to, and evaluate the outcome of each method.

We used the ZeroR algorithm, which selects the class that has the most observations, and uses that class as the result for all predictions.

The baseline score for the input chunks was calculated by finding the percentage of each class's items in total given number of products and selecting the class with the bigger one as our accuracy metric.

The ZeroR Accuracy for the two attributes can be found below:

- ZeroR Product Cleaning Group Accuracy: *0.520*
- ZeroR Product Group Accuracy: *0.377*

5.1.2 Distance Method Evaluation and Selection

Setup:

- *Algorithm:* Nearest Centroid Classifier
- *Attributes:*
  - Product Cleaning Group
  - Product Group
- *Split:* 80% training set, 20% test set
- *Distances:*
  - Euclidean Total
  - Euclidean Row
  - Euclidean Column
  - Cosine Average
  - Cosine Vector
  - Cosine Diagonal
  - KL - Divergence Average
  - KL - Divergence Vector
  - KL - Divergence Diagonal
  - KS - Test Average
  - KS - Test Vector
  - KS - Test Diagonal
  - Infinity Norm

In this experiment we assess the different Distance Methods, considering a fixed split of 80% train data and 20% test data, and using the Nearest Centroid classifier.

The results are visible in Figure 5.1 for Product Cleaning Group.

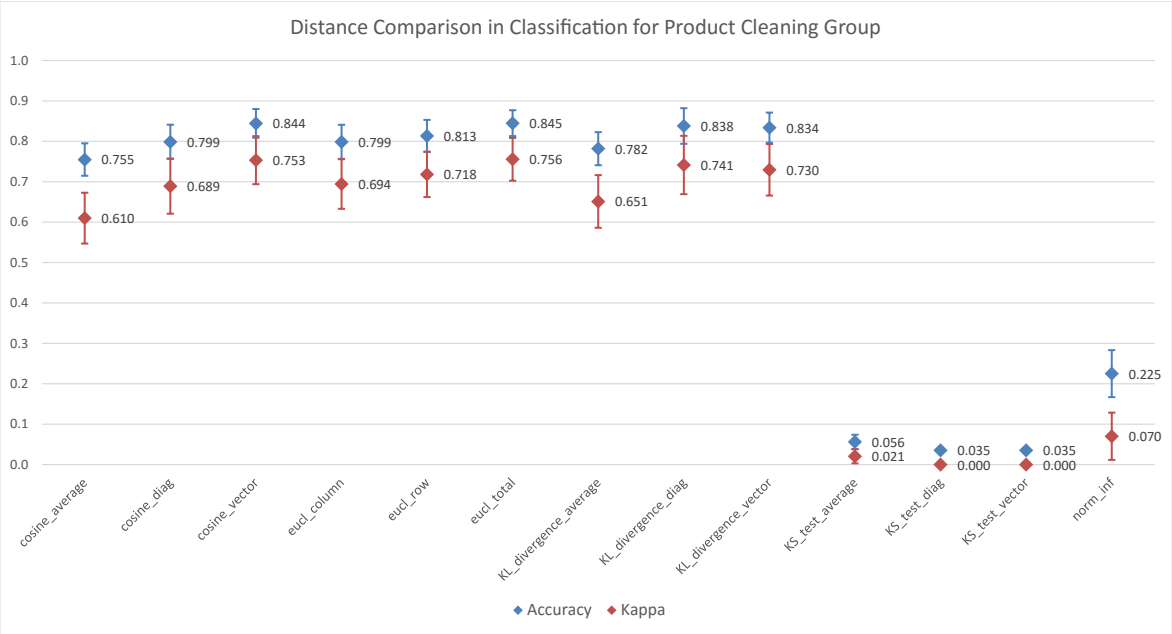

Figure 5.1: Distance Comparison for Product Cleaning Group Classification

The best Accuracy and Kappa Coefficient scores appear when Euclidean Total distance is applied, and are followed by Cosine Vector Distance and KL - Divergence- Diagonal Scores, which are also very close. All other Euclidean, Cosine and KL metrics are in quite proximity with the above maximum scores.

Scores of KS - Test metrics and Infinity Norm metric appear to be significantly low, implying that they are inappropriate for distance calculation in the current problem.

The standard Deviation in Accuracy has a maximum value of 0,058, while in Kappa Coefficient it maximizes at 0,068, thus is considered negligible for our conclusions in both metrics.

For Product Group, results are displayed in Figure 5.2, with the KL - Divergence Diagonal and KL - Divergence Vector distances to be having the best Accuracy and Kappa Coefficient scores. Next in line follow the KL - Divergence Average and the Cosine Distance Average, with very close highest scores each.

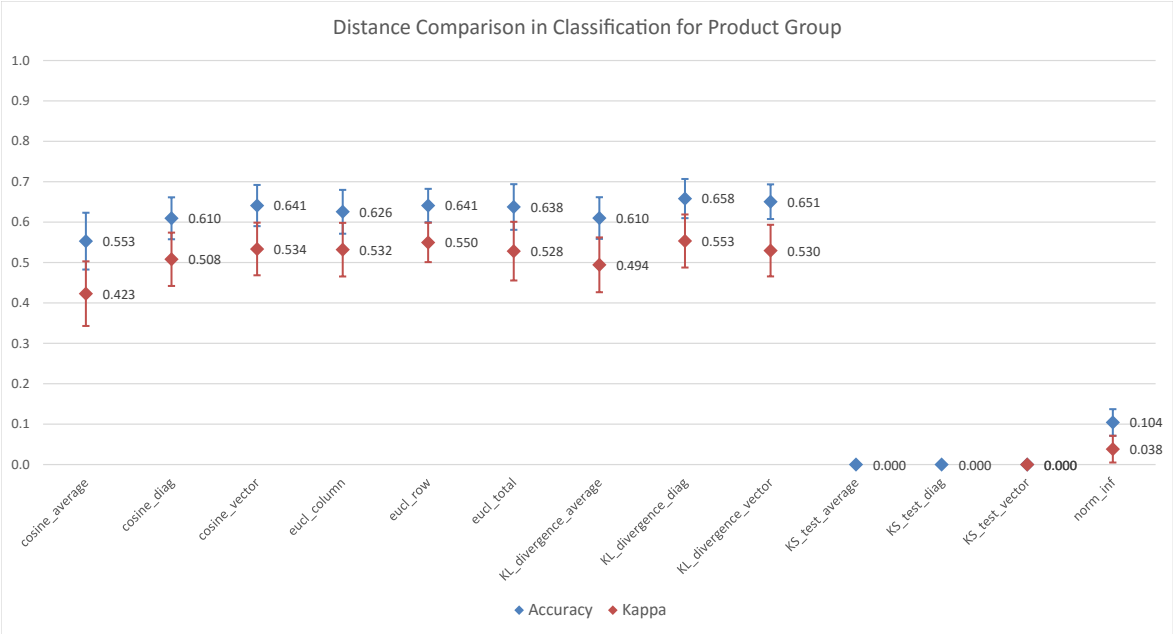

Figure 5.2: Distance Comparison for Product Group Classification

As in the previous experiment, the lowest scores appear in KS - tests and Uniform Norm results, which are concluded to be non- fitting for the current classification procedure.

Standard Deviation of both metrics has its maximum value at 0,056 and 0,799 respectively, so it is again insignificant to our results' accuracy.

5.1.3 Nearest Centroid and K - Nearest Neighbors Algorithms Comparison

Setup:

- *Algorithms:*
  - Nearest Centroid Classifier
  - k - Nearest Neighbors Classifier
- *Attributes:*
  - Product Cleaning Group
  - Product Group
- *Splits:*
  - 80% - 20%
  - 65% - 35%
  - 50% - 50%
- *Distance: Average of*
  - Euclidean Total
  - Cosine Vector
  - KL - Divergence Diagonal

In this section we compare the outcome of the application of the two algorithms, Nearest Centroid and k - Nearest Neighbors, on our data, focusing on the following parameters:

- Execution time
- Percentage of training- test data split
- Accuracy and Kappa Coefficient of classification result

The distance between the transition matrices of each chunk class object was kept fixed in this part of the simulation and was calculated as the average of the three different distances' results that appeared to have the maximum scores in the previous application : Euclidean Total, Cosine Distance-Vector and KL - Divergence Diagonal.

The results of the Average Algorithm's application are displayed in Figures 5.3 and 5.4, while the K -NN results in Figures 5.5 and 5.6.

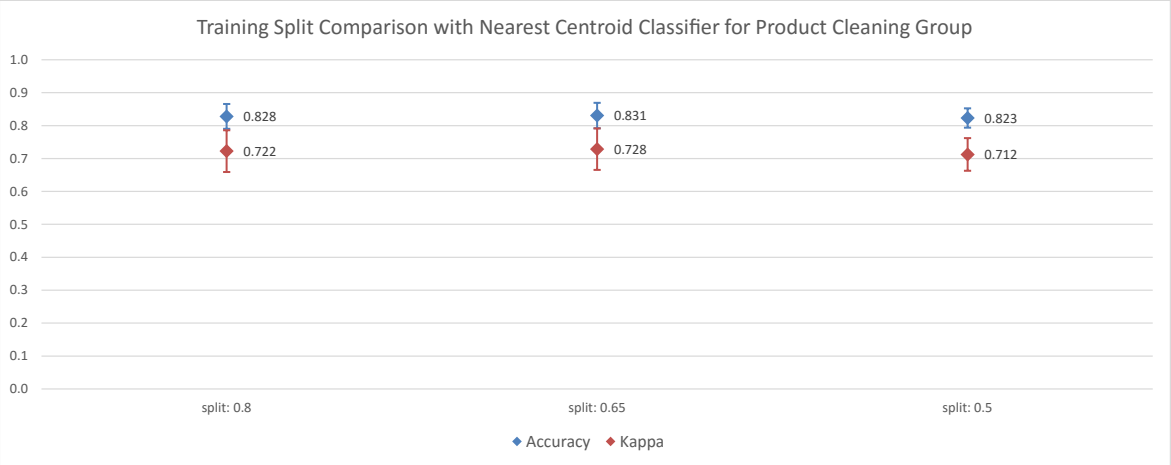

Figure 5.3: Nearest Centroid Classifier for Product Cleaning Group

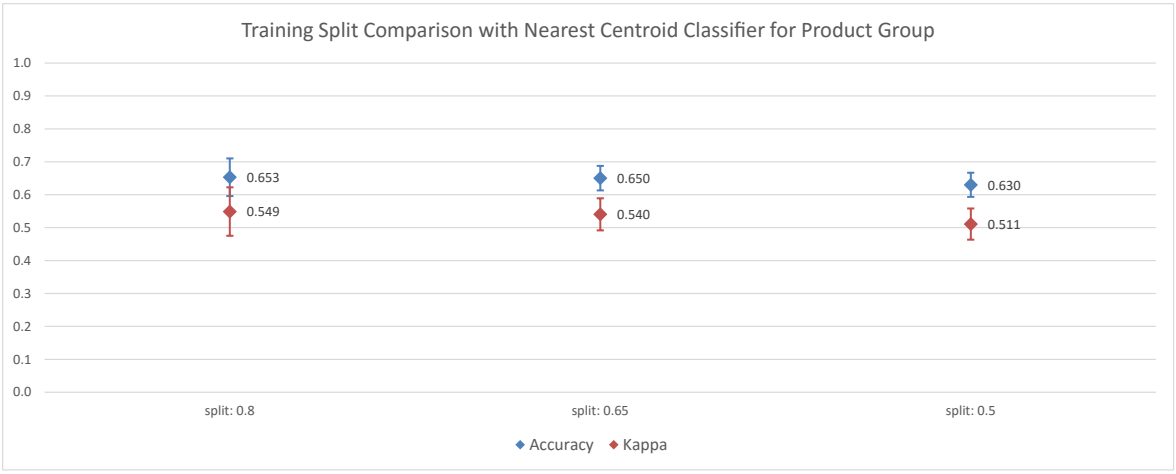

Figure 5.4: Nearest Centroid Classifier for Product Group

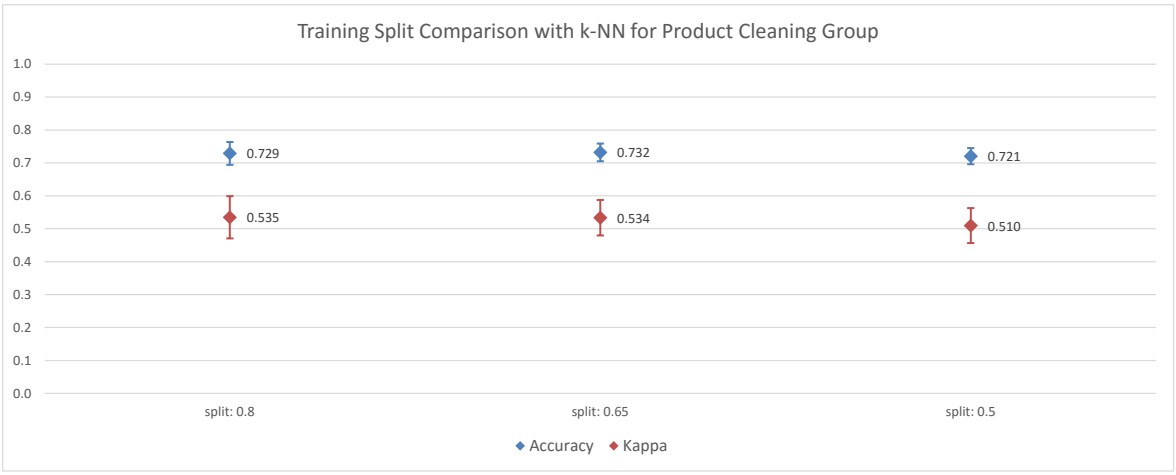

Figure 5.5: K Nearest Neighbors Classifier for Product Cleaning Group

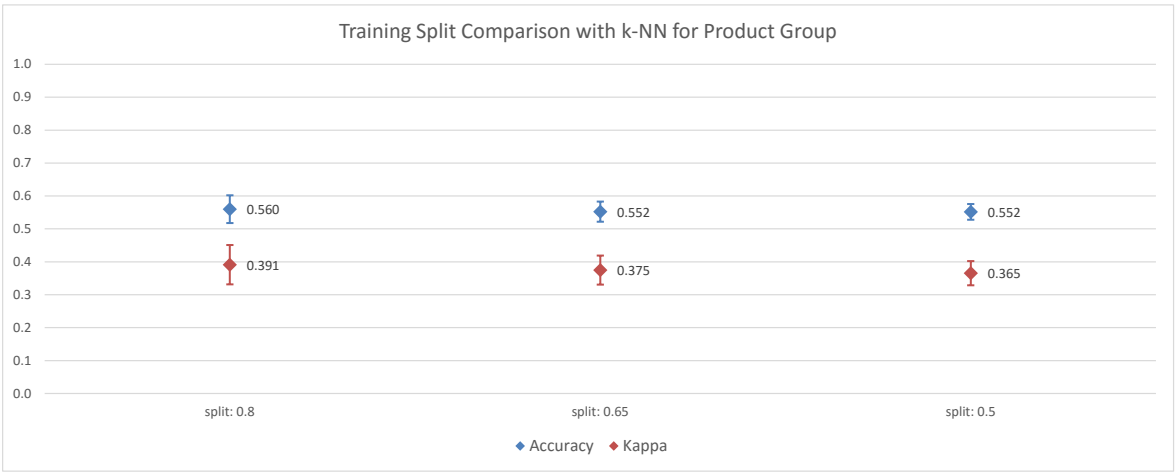

Figure 5.6: K Nearest Neighbors Classifier for Product Cleaning Group

It is obvious that, despite the three splits, the results of the classification are quite close, in both algorithms' charts. Regarding the Product Cleaning Group attribute, the Nearest Centroid algorithm gives maximum scores in the second split, with Accuracy in 83,1% and Kappa Coefficient in 72,8%,

while the same scores for the current attrribute in K - Nearest Neighbors are 73,2% and 53,4%.

The conclusions are similar when examining the Product Group attribute charts. Here, differences are also very subtle between the distinct splits, with the first one to achieve a slightly bigger score than the others. For the Nearest Centroid algorithm, that maximum is a 65,3% in Accuracy and a 54,9% in Kappa Coefficient, whilst for the k -NN the respective scores are 56% and 39,1%.

From these results, it is obvious that the Nearest Centroid algorithm is superior to the k -NN, for the current problem and regarding the above parameters.

A strong argument for this is also each algorithm's runtime, as the first one required 20,15 seconds, while the second one run in 161,46 seconds, time amount at least 8 times greater than the other classifier's, which implies that the algorithm is not suitable for the examined classification.

5.1.4 K - Nearest Neighbors Algorithm for Different k

Setup:

- *Algorithms:* k - Nearest Neighbors Classifier
- *Attributes:*
  - Product Cleaning Group
  - Product Group
- *Split:* 80% - 20%
- *Distance: Average of*
  - Euclidean Total
  - Cosine vector
  - KL - Divergence diagonal

The k -NN algorithm was applied for the above setup, so as to compare the effect of the selection of k in the classifier's general accuracy and performance.

The algorithm was implemented for the two usually examined attributes, Product Cleaning Group and Product Group, and for a fixed split of 80% - 20% of training and testing data.

The distance was kept fixed as well, and equal to the average of the distances which had the maximum scores in the first experiment, Euclidean Total, Cosine Distance Vector and KL - Divergence Diagonal.

Results are presented in Figure 5.7 for Product Cleaning Group and Figure 5.8 for Product Group.

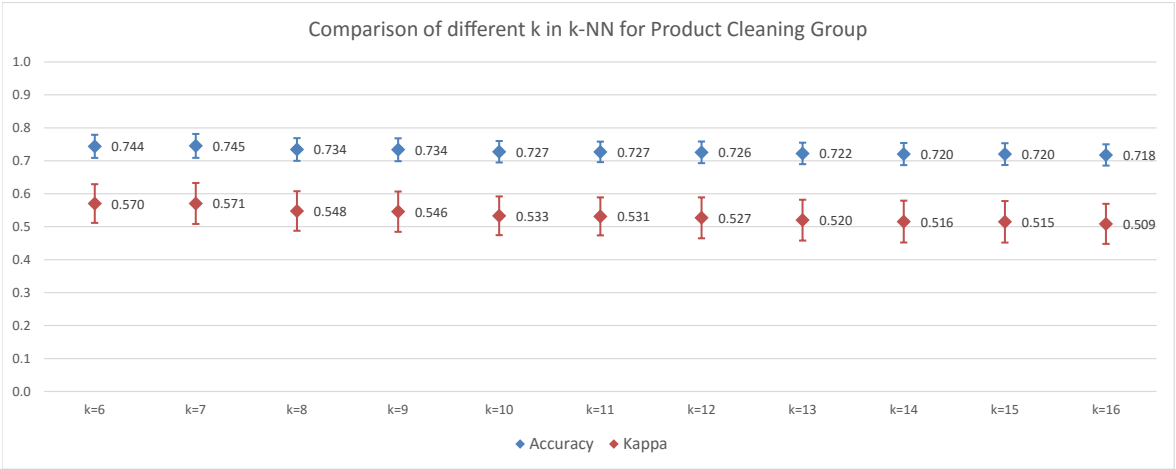

Figure 5.7: K - NN for Different k in Product Cleaning Group

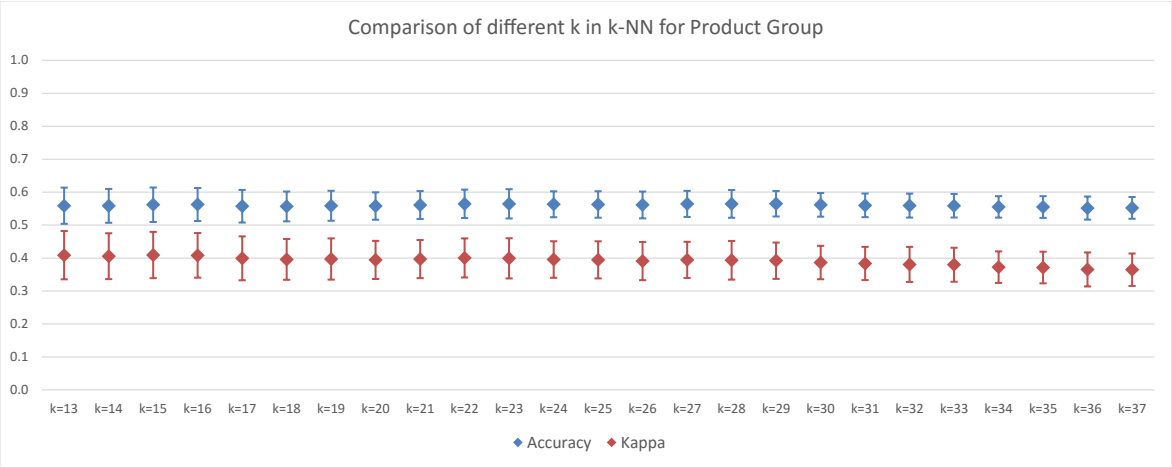

Figure 5.8: K - NN for Different k in Product Group

Observing the charts, it is obvious that the selection of k has little impact on the Accuracy and Kappa Coefficient scores of the classification process, even for very different and distant values of k.

In Product Cleaning Group classification, the maximum score is achieved with k=7 for both Accuracy (74,5%) and Kappa Coefficient (57,1%) metrics, whilst classifying on Product Group attribute gives a different best k for the two metrics. In this case maximum Accuracy is 57,7% for k= 23 and maximum Kappa score is 39,6% for k=13.

5.2 Clustering

5.2.1 Baseline

The baseline metric for the clustering procedure is calculated from the initial setup of the k - means for each simulation, which provides us with two initial scores. The output of each run is compared to these results, which are depicted on all the following diagrams, so as to assess the performance of the experiments performed.

5.2.2 Distance Algorithms

Setup:

- *Algorithms:*
  - k - Means
  - Baseline
- *Attributes:*
  - Product Cleaning Group
  - Product Group
- *Initial Centroid Sets Type:*
  - All centroids of each set belonged to different clusters (Alldiff)
    - Average of 20 sets
  - All centroids of each set belonged to the same cluster (Allsame)
    - Average of 20 sets
- *Distances:*
  - Euclidean Total

- Euclidean Row
- Euclidean Column
- Cosine Average
- Cosine Vector
- Cosine Diagonal
- KL - Divergence Average
- KL - Divergence Vector
- KL - Divergence Diagonal
- KS - Test Average
- KS- Test Vector
- KS- Test Diagonal
- Infinity Norm

- *Evaluation Methods:*

  - V - Measure
  - Rand - Index

In Figures 5.9 and 5.10 the k - means algorithm was executed for 20 different sets of initial centroids, which were selected so that all of them, in each set, belonged to either a different or the same cluster, which is one of the 5 different Product Cleaning Groups. It is observed that the Euclidean distance methods perform the best, although the value variation is relatively big. The maximum score is achieved for the Euclidean Row distance metric and is a 33,4% V - Measure and a 26,4% Rand - Index.

The low scoring percentages in all distances though, indicate that the clustering in general does not perform very well, yet its scores remain significantly above the baseline results.

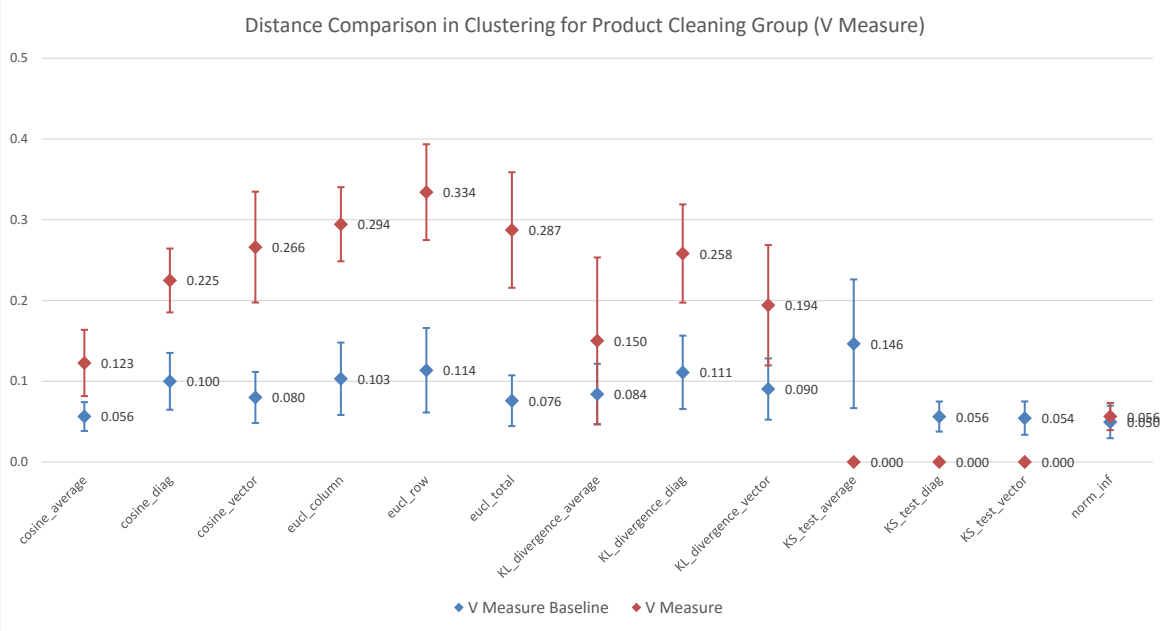

Figure 5.9: K means Clustering in Product Cleaning Group (V Measure)

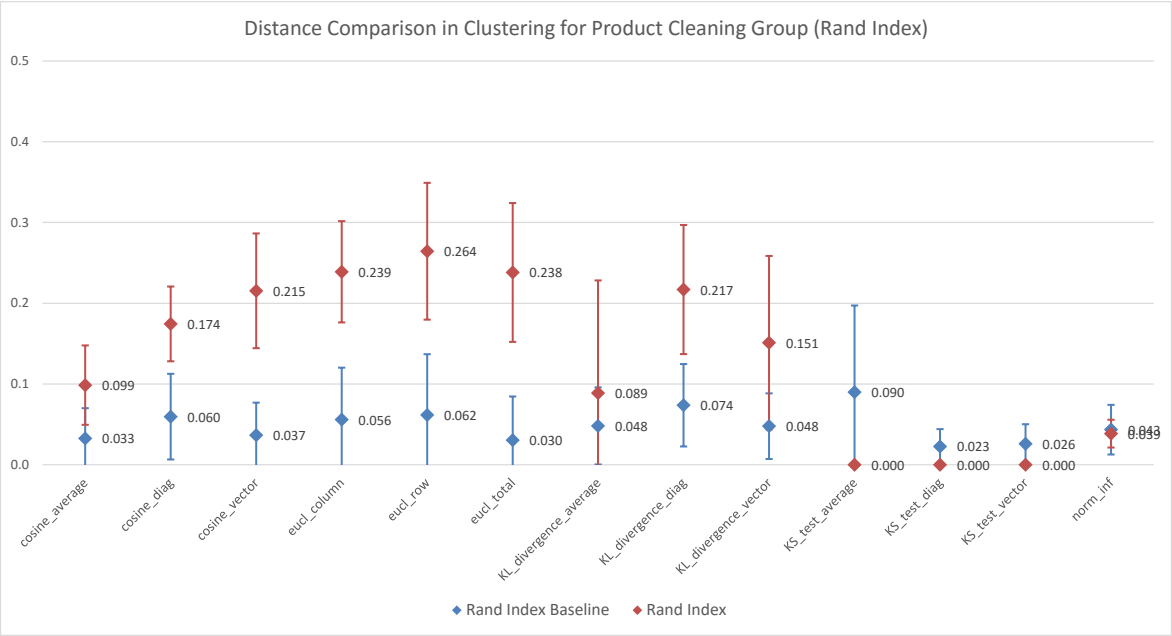

Figure 5.10: K means Clustering in Product Cleaning Group (Rand Index

Figures 5.11 and 5.12 is of the same logic as the previous ones, but here the clustering was performed in the Product Group level. This means that we have 12 potential clusters instead of 5 we had in the previous two experiments.

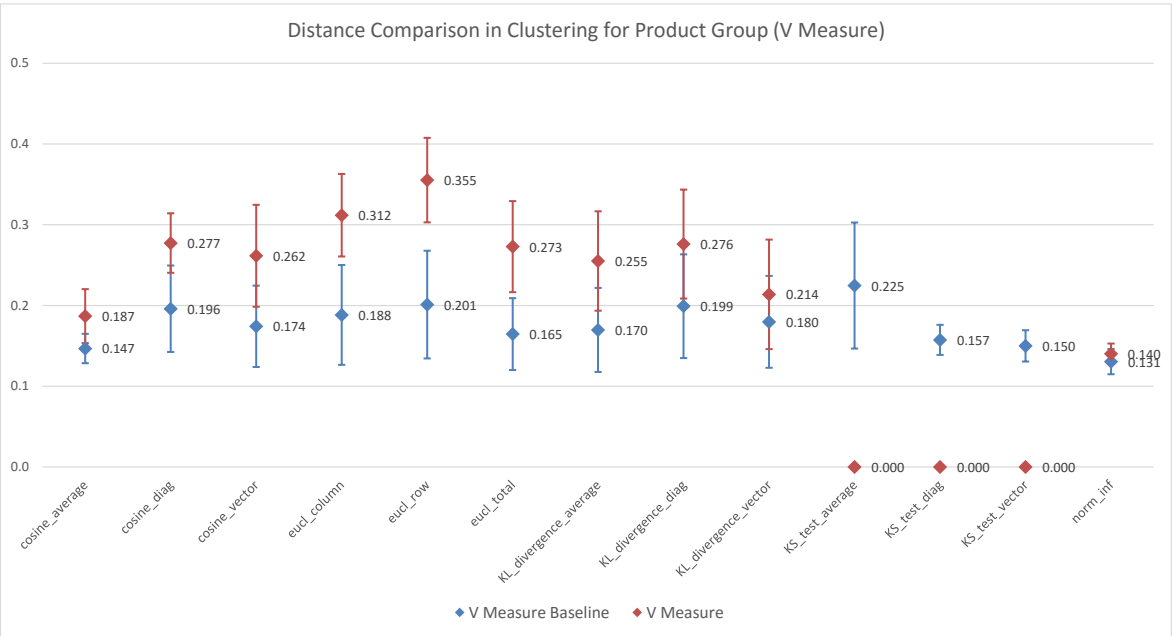

Figure 5.11: K means Clustering in Product Group (V Measure

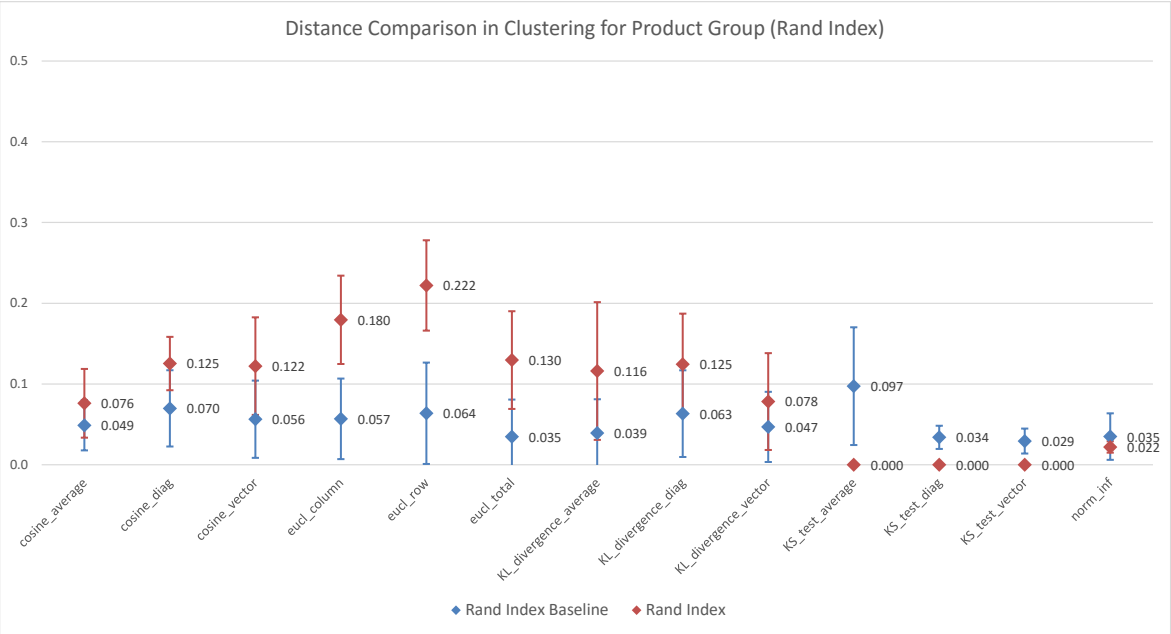

Figure 5.12: K means Clustering in Product Group (Rand Index

Euclidean Distance Metrics have the dominant scores here again, with maximum accuracy for Euclidean Row, which scores a V - Measure of 35,5% and a Rand - Index of 22,2%. KL - Divergence scores appear quite high as well.

In all cases, lowest scores come from the KS - Test and Uniform Norm metrics, and sometimes do not even exceed the baseline results.

5.2.3 Centroids

Setup:

- *Algorithms:*
  - K - Means
  - Baseline
- *Attributes:*
  - Product Cleaning Group
  - Product Group
- *Initial Centroid Sets Type:*
  - All centroids of each set belonged to different clusters (Alldiff)
    - Average of 100 sets
  - All centroids of each set belonged to the same cluster (Allsame)
    - Average of 100 sets
  - All centroids of each set belonged to a random cluster (Allrand)
    - Average of 100 sets
- *Distance: Average Of*
  - Euclidean Total
  - Euclidean Row
  - Euclidean Column
- *Evaluation Methods:*

– V - Measure
– Rand - Index

Figure 5.13 presents the results of the above setup for the scenario of clustering to 5 clusters, the Product Cleaning Groups.

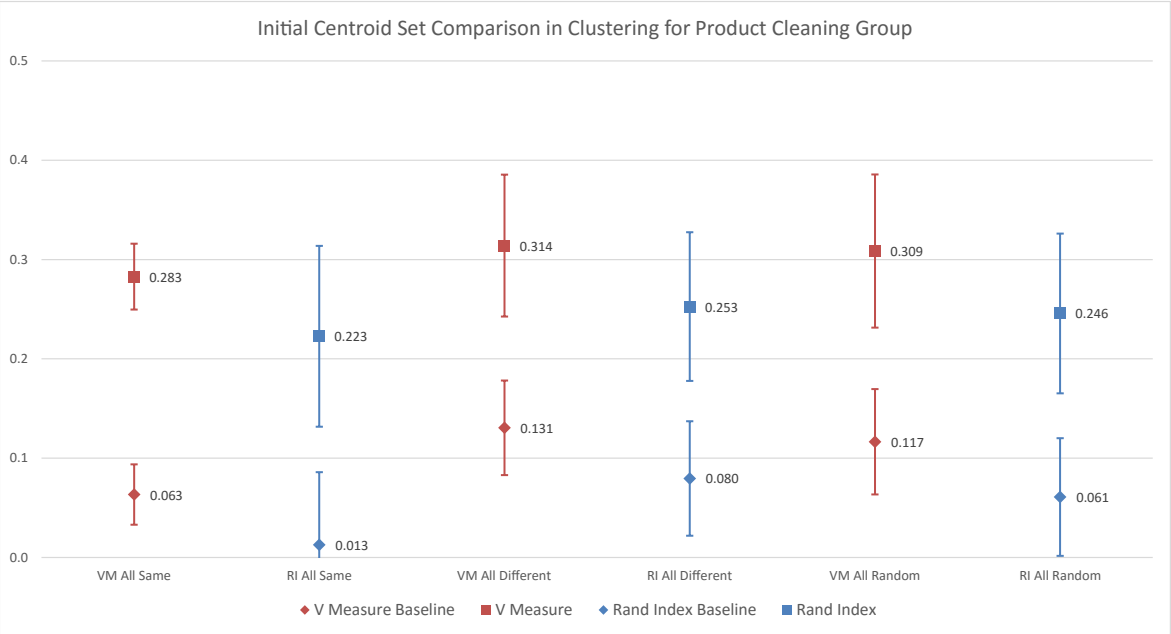

Figure 5.13: K means Clustering(Product Cleaning Group)

The best scores are visible when choosing initial centers that belong to different clusters, in which case there is a V - Measure of 31,4% and a Rand - Index of 25,3%. On the contrary, there is a lowest score when the initial centers all belong to the same Product Cleaning Group.

Figure 5.14 follows the above methodology with the difference that the process is here targeted to 12 clusters, as we are examining the Product Group attribute.

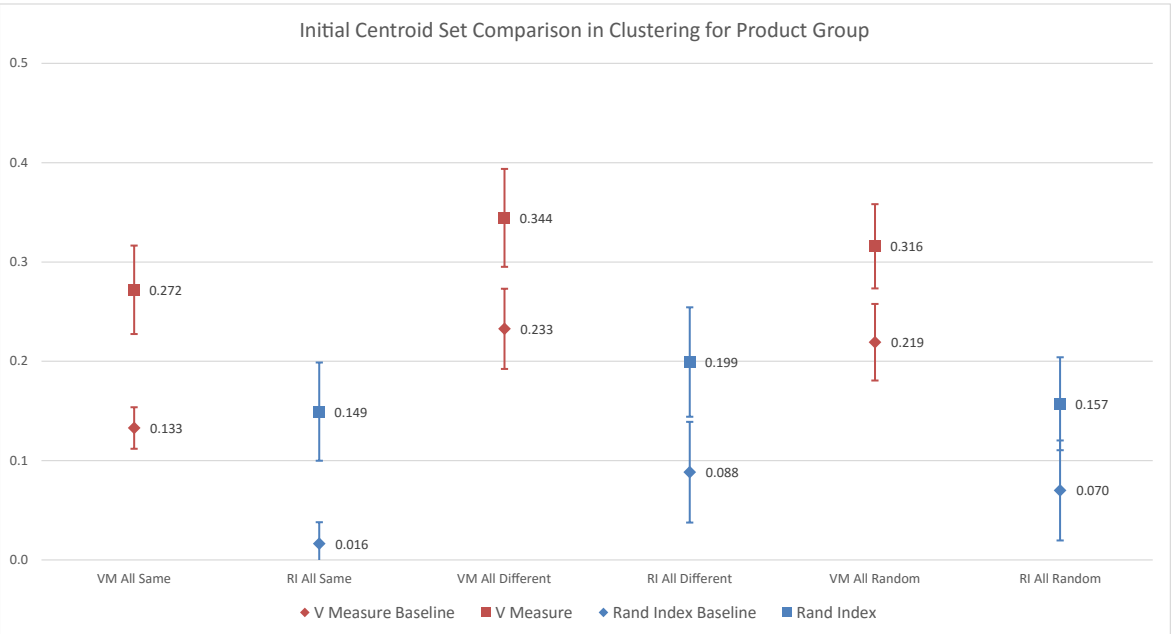

Figure 5.14: K means Clustering(Product Group)

As in the previous scenario, it can be observed that the result is best in the case of initial centroids belonging to different clusters and worst when they belong to the same.

It is quite obvious from the overall results that the clustering procedure has a poor general performance, as the scores do not exceed 35% for both attributes and all initial centers' combinations.

Chapter 6

Epilogue

In this Chapter, we present the conclusions from the executed experiments and also our proposals for future work on the problem.

In Section 6.1 we present the conclusions derived from all experiments of Chapter 5, and summarize the performance of Classification and Clustering, evaluating the contribution of each parameter.

In Section 6.2, we describe possible topics that we did not have the opportunity to examine during the current thesis, and are worth looking into in the future.

6.1 Conclusions

In this section we summarize the observations from our experiments and discuss the final conclusions.

6.1.1 Classification

Regarding the efficiency of the distance methods, the Euclidean ones always gave top scores, along with the KL- Divergence and the Cosine distance. It was quite difficult to identify whether the average, diagonal or vector methods performed better, as for every metric, the optimal score was achieved for a different one, and there was no general pattern. The KS- Test and the Infinity Norm scores proved to be unsuitable for our data, and their results rarely overcame the baseline ones. We assume that this was due to the nature of our distributions, which came in the form of the probability transition matrices.

Comparing the classifiers, Nearest Centroid classifier and K-Nearest Neighbours classifier, the first one has a significantly better performance, considering both Accuracy and Kappa scores, for all classification attributes. This is interesting, as the k-NN algorithm is the most commonly used classifier, and is well documented, while the Nearest Centroid algorithm was our adaptation of k-Means algorithm on the classification process, and was developed and defined intuitively based on our needs. We conclude that regarding the classes as ”centers” and assigning the data to the closest class outscores assigning the chunk to the closest neighbour, having examined the nearest ones. As mentioned in Chapter 5, the runtime for the k-NN was almost 8 times greater than that of the other classifier, which strongly validates the previous statement. This is due to the fact that k-NN has to calculate all distances from the k nearest class' centers, a task that can be very complex. The splitting in training and testing data had little effect on the resulting scores, and this is remarkable, if we take into account that the amount of data we used was few, and could show substantial differences for the different splitting percentages.

No significant variations were observed when running the k-NN for different k. This is a positive result, indicating that our data is insensitive to the selection of k, probably due to the format of our distributions.

6.1.2 Clustering

In all simulations, regardless of the parameterization of k-Means algorithm, the accuracy scores were quite low. So in general, the performance of k-Means on clustering the production chunks is considered to be poor.

As regard to the distances, the top three metrics were again Euclidean, Cosine and KL-Divergence, but with the Euclidean distances to rise significantly above the other two. Again, the KS-Test and Infinity Norm scores are slightly over the baseline results, proving that they fail to correlate the data in an efficient way.

It was observed in the initialization of the algorithm with different types of centroids, that initial centroids belonging all to different clusters gave the maximum score, followed closely by centroids all belonging to random clusters. This is expected, as k-Means by definition has to assign all data to different clusters, so, starting off with those clusters as centroids, gave a much quicker convergence and a more accurate result. On the contrary, when initial centroids were all in the same cluster, it would take many iterations to decentralize from the currently selected cluster, gradually calculate the others and assign the data correctly to them.

6.2 Future Work

Here are briefly presented the issues we did not have the time to investigate during the current thesis, and also issues that came up after our experiments and research, which could be further examined.

- **Distance Metrics**

  More distance methods could be used to calculate the distances between the Product chunks. Moreover, the way we calculate distances between the two dimensional matrices could vary again, and, similar to transforming our two dimensional matrix into a horizontal vector, we could also compress it into a vertical one and apply the different distant measures on that.

- **Clustering Algorithms** The poor performance of k - Means in Clustering could be a good motive to experiment with other clustering algorithms, which could prove to be more efficient in our distributions.

- **Classification using frequency vector and 3D transition matrix**

  The Classifier algorithms can be modified to receive different parameters. Instead of classifying the objects based on their messages' transition matrix, we could compute a simple frequency vector for each object, which depicts the percentage of appearance of each of our 45 different messages in this specific product. This vector is now the classifier's input, and all distances will be calculated between vectors, with the same methods as previously. In this way we evaluate each algorithm's performance, for all distances, when applied in the one dimensional frequency table, which obviously provides less information than the transitions one. The same thing could be applied for a three- dimensional transition matrix, which examines two successive messages and finds the transition probabilities in their sequence. In this way, complexity of the information given to the classifier is increased, and accuracy of results and performance of the algorithm could be a lot different than the one we observed after our simulations.

- **Different Variable Selection**

  In this work, we based our implementations on the Message Number sequence of each Production Chunk, as explained in the data Pre- Processing Chapter. In a different scenario, all variables characterizing our process (Temperature, Pressure, Agitation etc) could be taken into account in the algorithms' execution. This would require a different splitting of the initial data, and also a way to encorporate the extra parameters as attributes of the objects created. The input of each algorithm would need to be redefined, as the added variables are numerical, and so would the approach to calculate distances and centers, of either a class or a cluster in each execution. The experiments would have to be run on a differently parameterized setup, to examine the contribution of the new parameters separately, keeping all others fixed or making suitable combinations to identify correlations.

- **Scoring of production process**

  Production chunks refer to separate products being created. During the labelling process, we observed that production of the same product was not a solid fixed procedure, in terms of time duration, and order and kind of actions followed. This means that we had products all with the same attributes (Product Cleaning Group, Product Code, Product Group etc), but with different transition matrices, as the messages and their sequences in each one did not match.

  In order to overcome this problem, a recording of the optimal procedure is needed, for all products, so as to compare the actual one to this and eliminate the variations. For each product, one could track the series of actions that are considered to be ideal, and create the chunk objects for them. Then, the comparison of their transition matrices with the ones from the true products would give a measure of performance for each process, and allow us to score the Productions and determine if they were close to the ideal or not. This is a crucial first step for production optimization, and could boost the factory's overall performance, regarding quality of products, as well as minimization of production time.

- **Evaluation Metrics**

  In evaluating the results, we used a few indicative metrics, commonly applied in classification and clustering assesment. There are though, many more which could be applied, providing different correlations and conclusions. Two examples on this are the Silhouette Coefficient, which measures classes' or clusters' consistency, meaning how close each point in one cluster is to points in the neighbouring clusters, and the Student's T- Test, which determines the probability that two sets of data are samples of the same distribution. In this way, evaluation of results could be advanced in different levels and provide us some interesting results.

Appendix A

Scripts

A.1 Classification

A.1.1 Nearest Centroid Classifier

```
# Nearest Centroid Performance


attribute_list    = [”pr_cl_group”,”pr_group”]
split_percentages = [0.8,0.65,0.5]
num_iters         = 30
dist_methods      = [”eucl_total”,\
                     ”cosine_vector”,\
                     ”KL_divergence_diag”]

for attribute in attribute_list:
  for split in split_percentages:
    for iteration in range(num_iters):

      (train_set,test_set) = split_random(chunk_list,attribute,split)
      centers = calculate_centers(train_set,attribute)

      for dist_method in dist_methods:
        for chunk in test_set:
          assign_to_nearest_center(chunk,centers,dist_method)

        (accuracy,kappa) = evaluation(test_set,attribute)
```

Listing A.1: Nearest Centroid Classifier

A.1.2 K - Nearest Neighbors Classifier

```
# Classification Algorith K- Nearest Neighbors


attribute_list    = [”pr_cl_group”,”pr_group”]
split_percentages = [0.8,0.65,0.5]
num_iters         = 30
dist_methods      = [”eucl_total”,\
                     ”cosine_vector”,\
                     ”KL_divergence_diag”]
unique_pr_cl_gr   = 5
```

```
unique_pr_gr      = 12


for attribute in attribute_list:

  if attribute==”pr_cl_group”:
    min_k = unique_pr_cl_gr+1
    max_k = unique_pr_cl_gr*3+1
  elif attribute==”pr_group”:
    min_k = unique_pr_gr+1
    max_k = unique_pr_gr*3+1

  for split in split_percentages:
    for iteration in range(num_iters):

      (train_set,test_set) = split_random(chunk_list,attribute,split)

      for dist_method in dist_methods:

        for chunk in test_set:
          distances = calculate_distances(chunk,train_set,dist_method)
          sorted_distances = sort(distances)

          for k in range(min_k,max_k+1):
            assign_to_most_frequent(chunk,k,sorted_distances[:k])

        for k in range(min_k,max_k+1):
          (accuracy,kappa) = evaluation(test_set,k,attribute)
```

Listing A.2: K - Nearest Neighbors Classifier

A.2 Clustering

A.2.1 K - Means Implementation

```
# Clustering Algorith, Implementation of k-means


attribute_list     = [”pr_cl_group”,”pr_group”]
centers_sets_types = [centers_sets_all_random,\
                      centers_sets_all_diff,\
                      centers_sets_all_same]
num_sets           = 100
dist_methods       = [”eucl_total”,\
                      ”eucl_row”,\
                      ”eucl_column”]

for attribute in attribute_list:
  import_centers_sets(num_sets,attribute)
  for centers_set_type in centers_sets_types:

    for centers in centers_set_type:

      for dist_method in dist_methods:
```

```
20
21        assign_to_nearest_center(chunk_list,centers,dist_method)
22        evaluate_clustering_baseline(chunk_list,attribute)
23        centers_have_changed = True
24
25        while centers_have_changed:
26          old_centers = centers
27          centers = calculate_clusters_centers(chunk_list)
28          assign_to_nearest_center(chunk_list,centers,dist_method)
29
30          if (centers == old_centers) :
31            centers_have_changed = False
32
33        evaluate_clustering(chunk_list,attribute)
```

Listing A.3: Clustering Implementation K-means

A.3 Distances

A.3.1 2D to Vector Convertion

```
1 #Convert 2D matrix to Vector using the Vector algorithm
2
3 def m2v_vector(matrix):
4     ret_vect = []
5     [ret_vect.extend(row) for row in matrix]
6     return ret_vect
```

Listing A.4: Vector algorithm

```
1  #Convert 2D matrix to Vector using the diagonal algorithm
2
3  def m2v_diag(matrix):
4      ret_vect = []
5      for i in range(len(matrix)):
6          for j in range(len(matrix[0])):
7              if (i<j):
8                  ret_vect.append((matrix[i][j]+matrix[j][i])/2.0)
9              elif (i==j):
10                 ret_vect.append(matrix[i][j])
11     return ret_vect
```

Listing A.5: Diagonal algorithm

A.3.2 Distance Algorithms Implementation

```
1 #Euclidean Total distance algorithm
2 from scipy.spatial.distance import euclidean
```

```

def dist_eucl_total(chunk1, chunk2):
    return euclidean(m2v_vector(chunk1.TM),m2v_vector(chunk2.TM))
```

Listing A.6: Distance Euclidean Total

```
#Euclidean Total distance algorithm
from scipy.spatial.distance import euclidean

def dist_eucl_row(chunk1, chunk2):
    tt_sum=0
    for i in range(len(chunk1.TM)):
        tt_sum+=euclidean(chunk1.TM[i],chunk2.TM[i])
    return tt_sum
```

Listing A.7: Distance Euclidean Row

```
#Euclidean Total distance algorithm
from scipy.spatial.distance import euclidean
import numpy as np

def dist_eucl_column(chunk1, chunk2):
    tt_sum=0
    for i in range(len(chunk1.TM)):
        t_sum = 0
        for j in range(len(chunk1.TM[0])):
            t_sum+= np.square(chunk1.TM[j][i]-chunk2.TM[j][i])
        tt_sum+=np.sqrt(t_sum)
    return tt_sum
```

Listing A.8: Distance Euclidean Column

```
from scipy.spatial.distance import cosine
import numpy as np

def dist_cosine_average(chunk1,chunk2):
    results = []
    for i in range(len(chunk1.TM)):
        cosrow = cosine(chunk1.TM[i],chunk2.TM[i])
        if not np.isnan(cosrow):
            results.append(cosrow)
    return np.average(results)
```

Listing A.9: Distance Cosine Average

```
from scipy.spatial.distance import cosine

def dist_cosine_vector(chunk1, chunk2):
```

```
    return cosine(m2v_vector(chunk1.TM),m2v_vector(chunk2.TM))
```

Listing A.10: Distance Cosine Vector

```
from scipy.spatial.distance import cosine

def dist_cosine_diag(chunk1, chunk2):
    return  cosine(m2v_diag(chunk1.TM),m2v_diag(chunk2.TM))
```

Listing A.11: Distance Cosine Diagonal

```
from scipy.stats import entropy
import numpy as np

def dist_KL_divergence_average(chunk1,chunk2):
    results = []
    for i in range(len(chunk1.TM)):
        tmp_ch_1 = chunk1.TM[i]
        tmp_ch_2 = chunk2.TM[i]
        for j in range(len(tmp_ch_1)):
            if tmp_ch_1[j] == 0 :
                tmp_ch_1[j] = 0.001
            if tmp_ch_2[j] == 0 :
                tmp_ch_2[j] = 0.001
        results.append(entropy(tmp_ch_1,tmp_ch_2))
    return np.average(results)
```

Listing A.12: Distance Kullback–Leibler Divergence Average

```
from scipy.stats import entropy

def dist_KL_divergence_vector(chunk1,chunk2):
    tmp_ch_1 = m2v_vector(chunk1.TM)
    tmp_ch_2 = m2v_vector(chunk2.TM)

    for i in range(len(tmp_ch_1)):
        if tmp_ch_1[i] == 0 :
            tmp_ch_1[i] = 0.001
        if tmp_ch_2[i] == 0 :
            tmp_ch_2[i] = 0.001

    return  entropy(tmp_ch_1,tmp_ch_2)
```

Listing A.13: Distance Kullback–Leibler Divergence Vector

```
from scipy.stats import entropy

def dist_KL_divergence_diag(chunk1, chunk2):
```

```
    tmp_ch_1 = m2v_diag(chunk1.TM)
    tmp_ch_2 = m2v_diag(chunk2.TM)

    for i in range(len(tmp_ch_1)):
        if tmp_ch_1[i] == 0 :
            tmp_ch_1[i] = 0.001
        if tmp_ch_2[i] == 0 :
            tmp_ch_2[i] = 0.001

    return  entropy(tmp_ch_1,tmp_ch_2)
```

Listing A.14: Distance Kullback–Leibler Divergence Diagonal

```
from scipy.stats import ks_2samp
import numpy as np

def dist_KS_test_average(chunk1,chunk2):
    results = []
    for i in range(len(chunk1.TM)):
        Y = ks_2samp(chunk1.TM[i],chunk2.TM[i])
        results.append(Y[0]/Y[1])
    return np.average(results)
```

Listing A.15: Distance Kolmogorov–Smirnov test Average

```
from scipy.stats import ks_2samp

def dist_KS_test_vector(chunk1,chunk2):
    Y = ks_2samp(m2v_vector(chunk1.TM),m2v_vector(chunk2.TM))
    return Y[0]/Y[1]
```

Listing A.16: Distance Kolmogorov–Smirnov test Vector

```
from scipy.stats import ks_2samp

def dist_KS_test_diag(chunk1,chunk2):
    Y = ks_2samp(m2v_diag(chunk1.TM),m2v_diag(chunk2.TM))
    return Y[0]/Y[1]
```

Listing A.17: Distance Kolmogorov–Smirnov test Diagonal

```
import numpy as np

def dist_norm_inf(chunk1, chunk2):
    return np.linalg.norm(np.array(chunk1.TM) - np.array(chunk2.TM), np.inf)
```

Listing A.18: Infinity Norm

A.3.3 Evaluation of Distance Methods in Classification

```
# Classification Algorith to compare differnt distance methods


attribute_list    = [”pr_cl_group”,”pr_group”]
split_percentages = [0.8]
num_iters         = 30
dist_methods      = [”eucl_total”,\
                     ”eucl_row”,\
                     ”eucl_column”,\
                     ”cosine_vector”,\
                     ”cosine_diag”,\
                     ”cosine_average”,\
                     ”KL_divergence_vector”,\
                     ”KL_divergence_diag”,\
                     ”KL_divergence_average”,\
                     ”KS_test_vector”,\
                     ”KS_test_diag”,\
                     ”KS_test_average”,\
                     ”norm_inf”]

for attribute in attribute_list:
  for split in split_percentages:
    for iteration in range(num_iters):

      (train_set,test_set) = split_random(chunk_list,attribute,split)
      centers = calculate_centers(train_set,attribute)

      for dist_method in dist_methods:
        for chunk in test_set:
          assign_to_nearest_center(chunk,centers,dist_method)

        (accuracy,kappa) = evaluation(test_set,attribute)
```

Listing A.19: Classification Distance Methods Evaluation

A.3.4 Evaluation of Distance Methods in Clustering

```
# Clustering Algorith to compare differnt distance methods


attribute_list     = [”pr_cl_group”,”pr_group”]
centers_sets_types = [centers_sets_all_diff,\
                      centers_sets_all_same]
num_sets           = 20
dist_methods       = [”eucl_total”,\
                      ”eucl_row”,\
                      ”eucl_column”,\
                      ”cosine_vector”,\
                      ”cosine_diag”,\
                      ”cosine_average”,\
                      ”KL_divergence_vector”,\
                      ”KL_divergence_diag”,\
```

```
                    "KL_divergence_average",\
                    "KS_test_vector",\
                    "KS_test_diag",\
                    "KS_test_average",\
                    "norm_inf"]

for attribute in attribute_list:
  import_centers_sets(num_sets,attribute)
  for centers_set_type in centers_sets_types:

    for centers in centers_set_type:

      for dist_method in dist_methods:

        assign_to_nearest_center(chunk_list,centers,dist_method)
        evaluate_clustering_baseline(chunk_list,attribute)
        centers_have_changed = True

        while centers_have_changed:
          old_centers = centers
          centers = calculate_clusters_centers(chunk_list)
          assign_to_nearest_center(chunk_list,centers,dist_method)

          if (centers == old_centers) :
            centers_have_changed = False

        evaluate_clustering(chunk_list,attribute)
```

Listing A.20: Clustering Distance Methods Evaluation

Bibliography

[1] C.C Taylor D. Michie D.J. Spiegelhalter.
Machine Learning, Neural and Statistical Classification. Ellis Horwood, Feb. 1994, pp. 9–11, 14–16, 20.
URL: http://www1.maths.leeds.ac.uk/~charles/statlog/whole.pdf.

[2] Stuart J. Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice-Hall, Inc, 1995, pp. 26–28.

[3] Rokach Lior Maimon Oded. Data Mining and Knowledge Discovery Handbook. Second. Springer US, 2005.

[4] Julia Hirschberg Andrew Rosenberg. "Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning".
In: V-Measure: A conditional entropy-based external cluster evaluation measure.
Lecture Notes in Artificial Intelligence.
Prague: Association for Computational Linguistics, June 2007.

[5] S. B. Kotsiantis.
Supervised Machine Learning: A Review of Classification Techniques. Informatica 31:249–268. 2007.

[6] Elena Deza Michel Marie Deza. Encyclopedia of Distances. Springer, 2009.

[7] Petra Perner. "Industrial Conference on Data Mining (ICDM 2009)".
In: Advances in Data Mining Applications and Theoretical Aspects.
Lecture Notes in Artificial Intelligence. Leipzig, Germany: Springer-Verlag, July 2009.

[8] Ethem Alpaydın. Introduction to Machine Learning. Second.
The Adaptive Computation and Machine Learning. The MIT Press, 2010.

[9] Prof. Kimito Funatsu. Knowledge-Oriented Applications in Data Mining. InTech, Jan. 2011.
URL: http://www.intechopen.com/books/knowledge-oriented-applications-in-data-mining/data-mining-industrialapplications.

[10] Mark A. Hall Ian H. Witten Eibe Frank.
Data Mining: Pactical Machine Learning Toold and Techniques. Third.
Morgan Kauffman, Elsevier, 2011, pp. 3–9.

[11] URL: http://www.pmean.com/definitions/kappa.htm.

List of Figures