POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Matematica

Tesi di laurea Magistrale

Machine Learning methods for root cause analysis in Industry 4.0

Design and implementation of analytic models in the Food Digital Monitoring project

Relatore

prof. Francesco Vaccarino

Corelatore

dott. Fabrizio Pio Baldini

Simone Scarsi

matricola 244156

Anno accademico 2018-2019

Tesi: Machine Learning methods for root cause analysis in Industry 4.0

Candidato: Simone Scarsi Matricola: s244156

Relatore: Francesco Vaccarino

Corelatore: Fabrizio Pio Baldini

28 Marzo 2019

Contents

Acknowledgment 3

Summary 4

1 Introduction 5

1.1 Root cause analysis 5

1.2 Industry 4.0 - FDM 5

1.3 Bosch data collection 9

2 Exploratory analysis 11

2.1 Pre-processing analysis 11

2.2 Data processing 15

2.3 Post-processing analysis 16

2.3.1 Line 0 16

2.3.2 Line 1 38

2.3.3 Line 2 54

2.3.4 Line 3 71

3 Objectives 92

4 Methods 93

4.1 Group technology 93

4.1.1 Rank Order Clustering 96

4.1.2 Similarity Coefficient Algorithm 99

4.1.3 Customized Similarity Coefficient Algorithm 101

4.2 Random forest 104

4.2.1 Standard approach 105

4.2.2 Downsampling approach 110

5 Results 116

5.1 Application 117

5.2 Conclusion 120

Acknowledgment

I would like to thank my supervisor Prof. Francesco Vaccarino for his expertise, great knowledge and the continuous support offered during the writing of this Thesis.

My gratitude also goes to aizoOn Consulting for the hospitality and the stimulating environment, that made internship and opportunity of personal and professional growth. A special thank goes to Fabrizio Pio Baldini and Simona Dutto, that accompanied me in this experience.

A grateful thank also goes to my family, for the encouragement and support offered me through all my studies, that allowed me to reach this goal.

I would like to thank my classmates and all my friends that helped me to complete my studies.

Finally I want to sincerely thank Giulia for her great patience and all the support she gave me in these years.

Summary

The rising of data science, internet of things and artificial intelligence have profoundly impacted manufacturing. The amount of data produced by manufacturing is growing and adopting data-driven strategies can make factories more efficient and competitive. The final goal of this Thesis is to improve performances of production chains in Industry 4.0, detecting and correcting failures through a root cause analysis approach. Machine learning approaches have been combined to Group Technology algorithms, bringing to new analysis patterns. Results have been applied to food manufacturing sector, giving the possibility of many future development of FDM project.

Chapter 1

Introduction

1.1 Root cause analysis

Root Cause Analysis (RCA) is a technique used for identifying the reason of a problem, instead of simply reporting and correcting the problem itself, as described in [7]. The problem can be intended as a non-compliant situation or an error in a production chain. Focusing on root causes correction has the benefit of improving long term efficiency of industrial processes, systematically reducing the occurrence of errors.

Analysis is done after the event occurrence, thus RCA process is different to incident management. Causes of a problem can be multiple and both correlated or independent of each other. After a deep comprehension of causes, RCA can be used to predict probable failures even before their occurrence.

Application of RCA to industrial process can lead to the development of a smarter production chain, with great benefits in production costs and efficiency.

1.2 Industry 4.0 - FDM

Industry 4.0, as described in [8], is the name of a project of the German government started 2011, focused on the modernization of the productive

system. That terminology and initiatives, which refers to the fourth industrial revolution, have been adopted and extended first by European Union then by the rest of the world. Figures of this section have been found in [9], in Figure 1.1 is shown industry evolution.

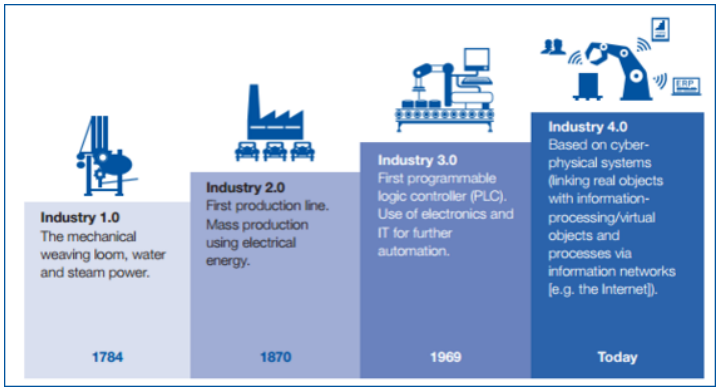


Figure 1.1: Industry evolution

The main categories of technological development supported by the industry 4.0 plan indicated by the Economic Development Ministry are shown in Figure 1.2.

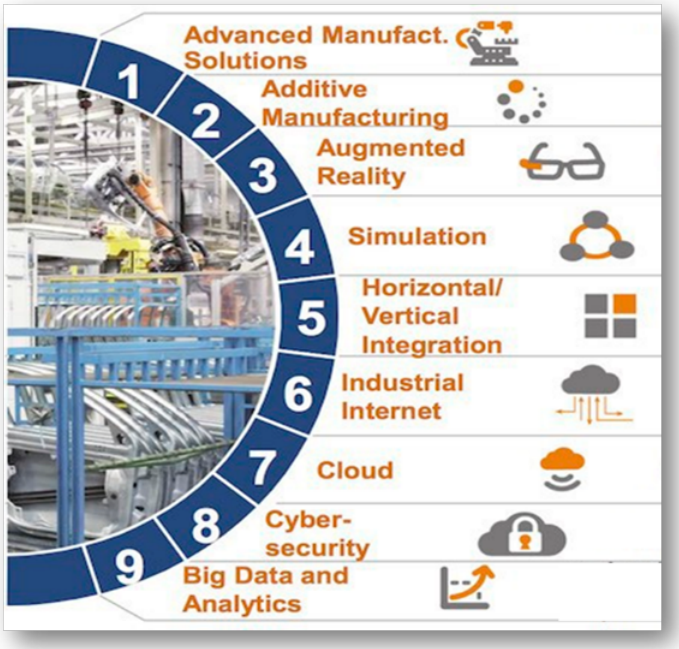


Figure 1.2: Industry 4.0 categories

The project in exam belongs to the category of Big data and analytics,

it refers to the macro area of IoT (Internet of Things), namely industrial internet integration with innovative sensors and data analytics. A definition of IoT is 'network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment'. In particular, Industrial Internet of Things (IIoT) refers to the application of IoT to the specific industrial fields. The characterizing elements of IIoT are:

- Interconnection of sensors, processes, machines, environment and people
- Huge data generation, with possibility of exchanging and processing
- Possibility of adopting, complementary to the existing industrial automation infrastructures, of innovative sensors to better characterize real-time processes, materials, consumption
- Usage of data generated from the production machines, often considered not useful and discarded
- Exploitation of the Advanced Analytics and Machine Learning techniques for real time processing of the vast amount of data generated (Data Lake), with the purpose to extract valuable information
- Possibility of generating automatic feedback to production processes, displaying the results of data analysis

Industrial IoT allows a better characterization of the production process, giving rational support for management decisions and leading to reduction of human interventions and wastes. Moreover, with the identification of predictive patterns on the evolution of the state of machines and processes, is possible to avoid or reduce production block events, optimizing the planning of maintenance operations. Finally, the possibility of a real-time production feedback allows to a rapid identification of production errors that brings to increase and stabilize the quality of the outgoing product despite the variability of the raw material.

Food Digital Monitoring

Food Digital Monitoring (FDM) is one of Finpiemonte's a research & development projects that aims to create a model of industrial food processing control, based on IoT, open data and big data. Data related to chemical, physical and microbiological parameters are collected throughout the food production chain, then they are elaborated, allowing a detailed monitoring over the productive process. The monitoring system obtained allows to detect and correct anomalies very quickly, improving efficiency and reducing waste and rework. The ultimate goal of FDM project is to create a 'Smart Industry' able to monitor in Near Real Time every critical phase of the production process, improving security and economic and environmental sustainability.

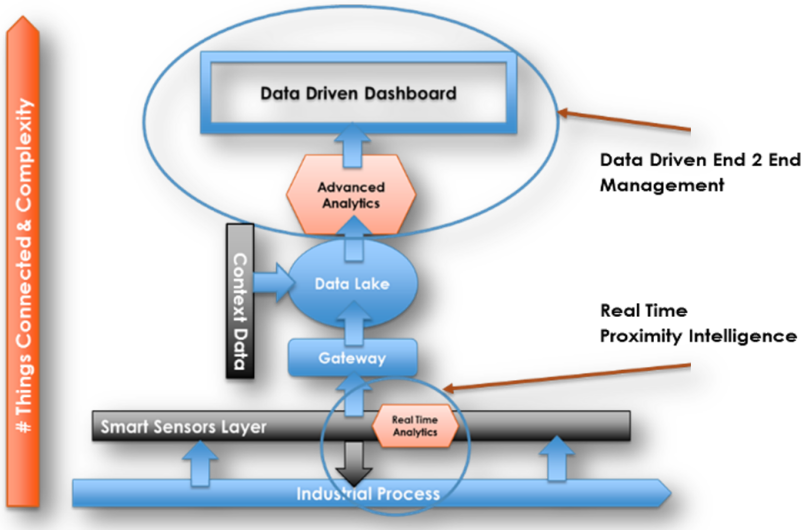


Figure 1.3: Data flow

FDM project is coordinated by aizoOn Consulting, in collaboration with Piedmont's food industries related to products of excellence, such as chocolate, beer, oil, probiotics and food supplements. Data of production chain are not available at the moment because project is under development, thus most of the analysis will be performed over open data obtained by Bosch production chain. Bosch datasets have been published on Kaggle web site,

in occasion of a machine learning competition.

1.3 Bosch data collection

Bosch data collection, available at the link [5], contains a great number of measurements generated by parts while they move through Bosch's production lines. Each part is characterized by a quality control Response and a unique Id. If the part passes the quality control, it will be characterized by Response=0, while if it fails, it will be characterized by Response=1. The goal is to predict which part will fail quality control and to understand the cause of the failure.

The data collection is split in three datasets: Categorical, Numeric and Date.

- **Numeric**: contains the values of all the numeric variables in the dataset, included the value of Response
- **Categorical**: contains the values of all the categorical variables in the dataset
- **Date**: contains a sort of time stamp that specifies when the data of categorical and numeric have been collected

Each file contains a single row collecting data generated by a certain part, indexed with an unique *Id*, and a large number of columns, characterized by a unique code. The code denotes a particular feature and depends on the production Line, the Station on the Line and the feature number. Categorical and Numeric describe feature values and are coded using the same convention: for example, a feature characterized by the code 'L0_S1_F31' is the feature number 31, measured on Line 0, Section 1.

Date specifies when values have been collected. Each column name ends in a number that corresponds to the previous feature number: for example, in the column labeled 'L0_S1_D32' in Date, we can find the timestamp of the feature named 'L0_S1_F31', contained in Numeric or Categorical. It

must be observed that there isn't a two-way correspondence between the columns in Date and the columns in Numeric and Categorical, in fact, some variable hasn't a correspondent timestamp.

Numeric, Categorical and Date share only one common column, named *Id*, containing numeric values sorted in growing order. Each value in *Id* is a unique number that refers to a particular part. If I want to collect all data related to a part, I have to join the three tables using *Id* as key.

Chapter 2

Exploratory analysis

2.1 Pre-processing analysis

Before applying any kind of cleaning process, datasets need an introductory analysis. The structure of each file will be studied separately.

Pre-processing Numeric

The Numeric dataset, before any kind of cleaning process, will be called PreProcNumeric. PreProcNumeric dataset contains all the values of the numeric features and the column *Response*, which describes the error that will be the target of the prediction. Figure 2.1 is a sample of a few columns of the dataset.

Id	L0_S0_F0	L0_S0_F2	L0_S0_F4	L0_S0_F6	L0_S0_F8	L0_S0_F10	L3_S51_F4258	L3_S51_F4260	L3_S51_F4262	Response
4	0.030	-0.034	-0.197	-0.179	0.118	0.116	NaN	NaN	NaN	0
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
7	0.088	0.086	0.003	-0.052	0.161	0.025	NaN	NaN	NaN	0
9	-0.036	-0.064	0.294	0.330	0.074	0.161	NaN	NaN	NaN	0
11	-0.055	-0.086	0.294	0.330	0.118	0.025	NaN	NaN	NaN	0

Figure 2.1: Header of PreProcNumeric

In Figure 2.2 is shown the result obtained calculating the percentage of not NaN values and *Response=1* rows.

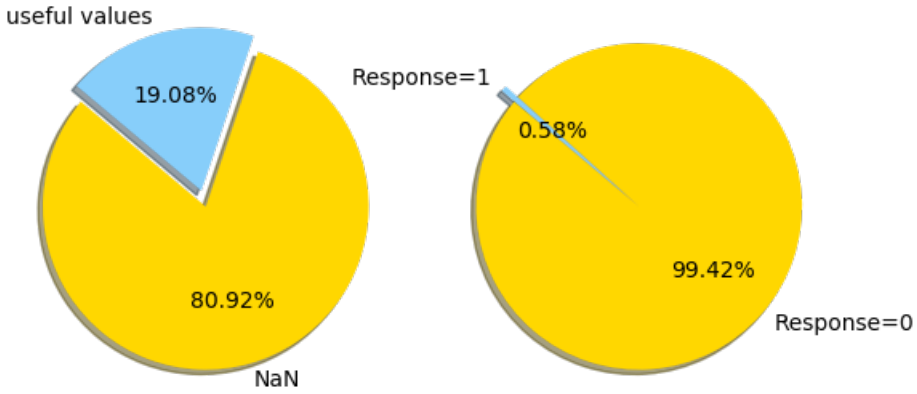


Figure 2.2: Percentage of not NaN values and Response=1 rows in PreProc-Numeric

In Figure 2.2, there are less than 20% of useful values, while the remaining part is composed by NaN, i.e. 'Not a Number'. Because each Line describes the passage of a particular part through the production lines, we can imagine that a NaN in a particular columns means that the part didn't passed through the area of the production line related to the feature described in that column.

The errors, described by $Response = 1$, are only about 0.5% of the total. Since positive responses ($Response = 0$) and negative responses ($Response = 1$) are disproportionately distributed, to make a good prediction of the errors, we will have to focus to get a very low false positive rate.

Pre-processing Categorical

The Categorical dataset, before any kind of cleaning process, will be called PreProcCategorical. PreProcCategorical dataset contains all the values of the categorical features. There are only a few not NaN values, in fact, if we check a few columns of the header as done in PreProcNumeric, we only find NaN values.

The row shown in Figure 2.3 is the first one also containing not NaN values and it has a very big Id number.

Id	L0_S1_F25	L0_S1_F27	L0_S1_F29	L0_S1_F31	L0_S2_F33	L3_S49_F4237	L3_S49_F4239	L3_S49_F4240
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...								
15497	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15498	T1	T9	T1	T9	NaN	NaN	NaN	NaN
15500	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15502	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15505	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 2.3: First not null row in PreProcCategorical

Calculating the percentage of not NaN values, is possible to notice that there are less than 3% of useful values, while the remaining part is composed by NaN. Dropping useless rows and columns during the cleaning process will help to increase this percentage.

Pre-processing Date

The Date dataset, before any kind of cleaning process, will be called PreProcDate. PreProcDate dataset contains all the timestamps, that describes when data of Numeric and Categorical have been collected. In Figure 2.4 we can see the header.

Id	L0_S0_D1	L0_S0_D3	L0_S0_D5	L0_S0_D7	L0_S0_D9	L0_S0_D11	L0_S0_D13	L0_S0_D15	L0_S0_D17	L0_S0_D19	...
4	82.24	82.24	82.24	82.24	82.24	82.24	82.24	82.24	82.24	82.24	...
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
7	1618.70	1618.70	1618.70	1618.70	1618.70	1618.70	1618.70	1618.70	1618.70	1618.70	...
9	1149.20	1149.20	1149.20	1149.20	1149.20	1149.20	1149.20	1149.20	1149.20	1149.20	...
11	602.64	602.64	602.64	602.64	602.64	602.64	602.64	602.64	602.64	602.64	...

Figure 2.4: Header of PreProcDate

As well as the other datasets, there are a lot of NaN and only less than 18% of useful values, as shown in Figure 2.5.

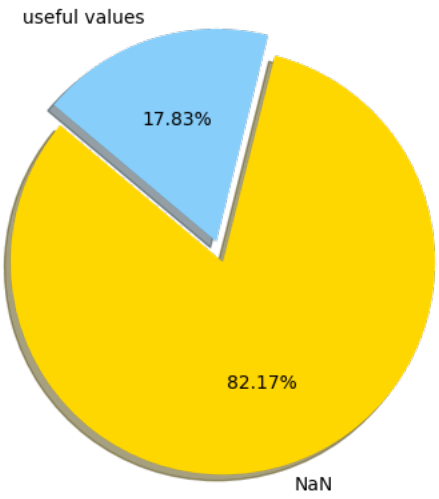


Figure 2.5: Percentage of not NaN values in PreProcDate

Timestamps are saved as numeric values, not in a date format as the name could suggest. We can imagine that each value could represent the time, in seconds, passed from the passage of the previous part through the line.

2.2 Data processing

After a brief analysis, datasets can be cleaned, in order to get them smaller and more significant. The following is a resume of each step of the cleaning process.

Data splitting

Data sources are semi-structured files in *CSV* format: each file weights more than $2Gb$. To make them easier to be read and elaborated, each file has been split by *production line*. There are four lines, called '$L0$', '$L1$', '$L2$', '$L3$'. From now, each Line will be considered separately.

Data normalization

A variable is considered useful for the analysis if it assumes different values in different rows, while a constant variable doesn't bring any information and can be eliminated. PreProcNumeric and PreProcDate didn't contain any useless variable, while 81 columns have been eliminated from PreProcCategorical, which contained only NaN values and didn't have a correspondent column in Date.

A row, characterized by a particular Id, is considered useless for a certain *Line* if it contains only NaN values in all three tables PreProcNumeric, PreProcCategorical and PreProcDate, except in the column *Response*, that can assume only 0/1 values.

After a separate analysis for each *Line*, each useless row has been deleted. In the end of the process, some thousand of row from each Line have been eliminated. Only two rows with Response=1 have been deleted from all four lines: comparing them with the thousands of rows with Response=0 deleted, the loss of information can be considered insignificant.

2.3 Post-processing analysis

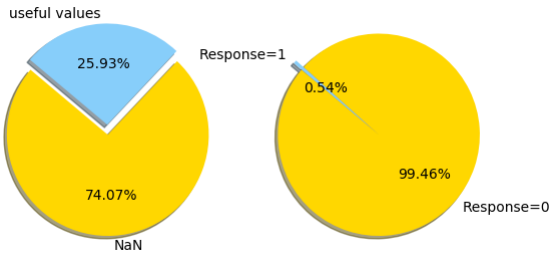
After processing, datasets can be examined more in depth. Reading the documentation, we can understand that each Line represent a different production line inside the Bosch factory. Each Line will be considered separately, studying how the structure of each dataset changes from Line to Line and then focusing on the relation among the three data sources. The goal is to find for each Line the most useful features for the error prediction.

2.3.1 Line 0

After the cleaning process, each dataset contains 916029 rows related to *Line*0, indexed by the common primary key *Id*. Each file will be initially studied separately, then will be studied how the three dataset are each other related.

Line 0 Numeric

The dataset obtained processing PreProcNumeric and isolating information related to Line 0, will be called L0Numeric. L0Numeric contains 169 columns. In Figure 2.6 is shown the result obtained calculating the percentage of not NaN values and Response=1 rows in this particular Line.



useful values	NaN	Response=1	Response=0
25.93%	74.07%	0.54%	99.46%

Figure 2.6: Percentage of not NaN values and Response=1 rows in L0Numeric

The number of not NaN values has been slightly increased after the cleaning process, but they are still very frequent. To visualize the distribution of NaN values over the dataset, has been used an heatmap: a different

color is assigned for each numeric value, while the NaN values are left without color. In Figure 2.7 is shown the result obtained plotting the first 100 rows.

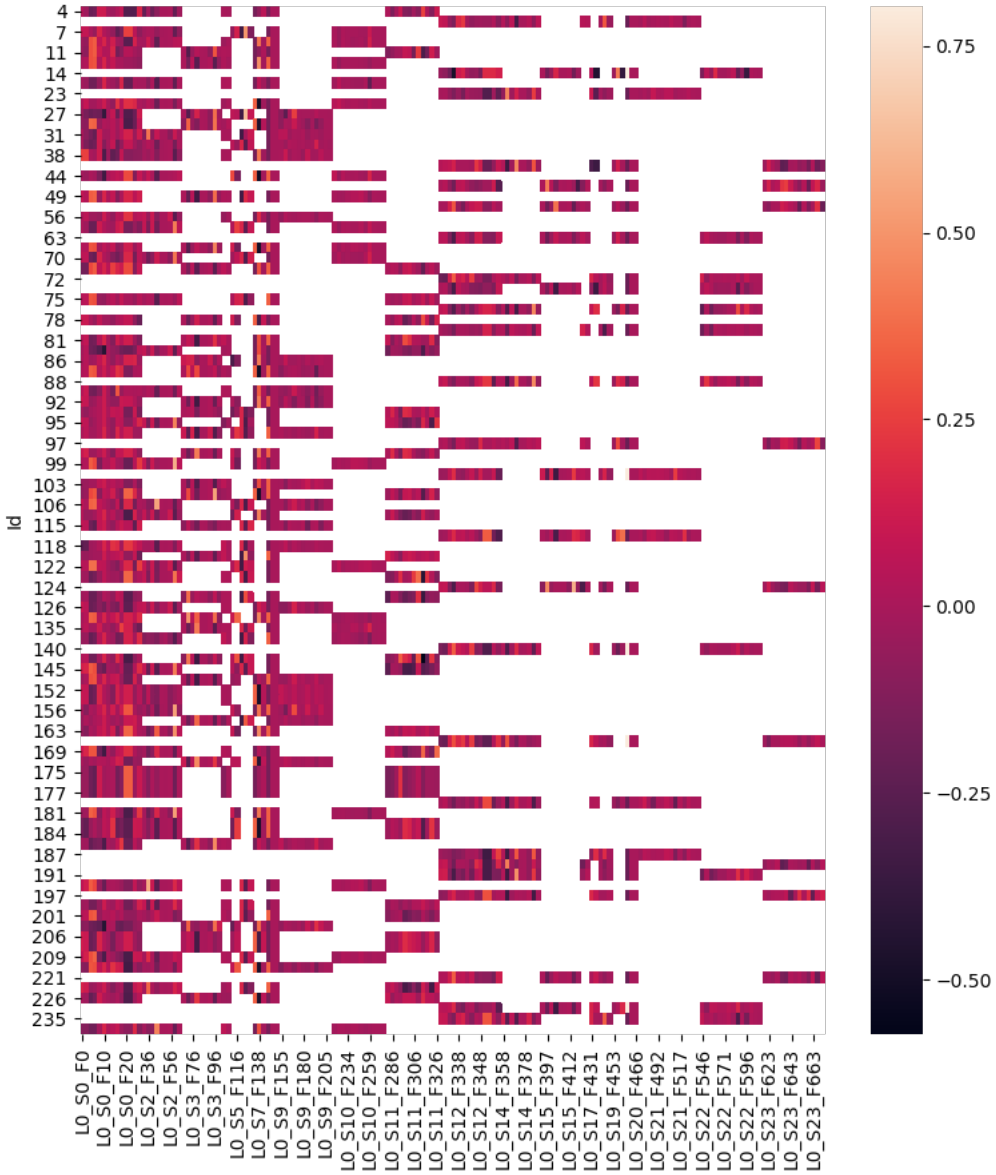


Figure 2.7: Heatmap of the first 100 rows of L0Numeric

NaN values are not random distributed. In fact, features look to be gathered in clusters within which, they have a similar behavior: or all they are NaN, or all they assume a numerical value. Looking at the identification

codes, we can understand that features with the same behavior belong to the same Section. We can suppose a Section to be a particular machinery, containing various sensors: each sensor is represented by a specific feature in the dataset. This theory explains why features in the same section have a common behavior: if a certain part passes through the machinery represented by a certain Section number, all the features in that Section will assume numeric values, while if the part doesn't pass through that machinery, all the features will assume NaN value. With this mind, we can study the relation between Sections and Response.

A Section affects the value of Response only if the part passes through the machinery it represents, i.e. only if its features assume not NaN values. In order to find a correlation between the error and the activation of a particular section, features have been grouped by Section (for example: features 'L0_S0_F0' and 'L0_S0_F2' belong to the group 'S0'), then can be defined an index, that will be called *error rate* of the Section S_i:

$$err(S_i) = \frac{N_1(S_i)}{N(S_i)} \tag{2.1}$$

Where $N_1(S_i)$ indicates the number of rows where the Section S_i has not NaN values and Response=1, $N(S_i)$ indicates all the rows where the Section S_i has not NaN values.

The average error rate $\overline{err}$ of the dataset coincides with the percentage of Response=1 described above. If $err(S_i)$ is bigger than the average, the probability of Response=1 is higher when the Section S_i is active (i.e. assumes not NaN values), vice versa the probability is lower if the $err(S_i)$ is lower than average. In Figure 2.8 has been calculated the $err(S_i)$ for each Section in Line 0 and it has been compared with the average $\overline{err} = 0.00537$ (red line).

There aren't Section with a particular importance, but we will have more interesting results in the following Lines.

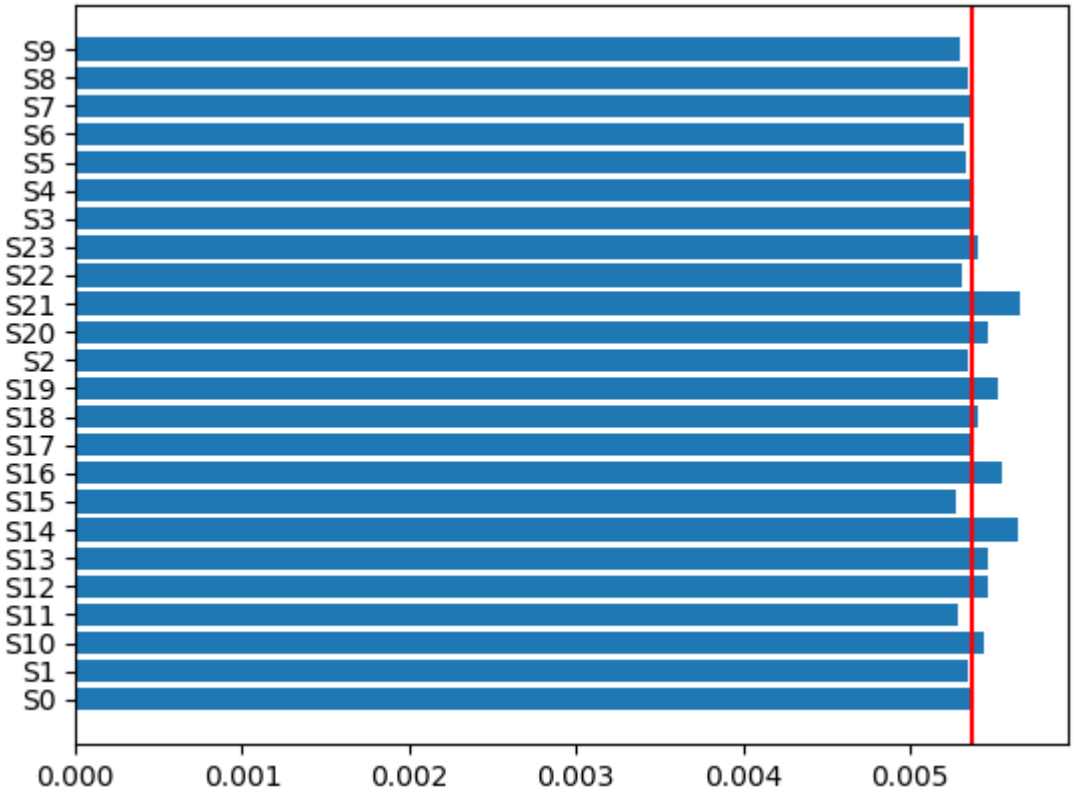
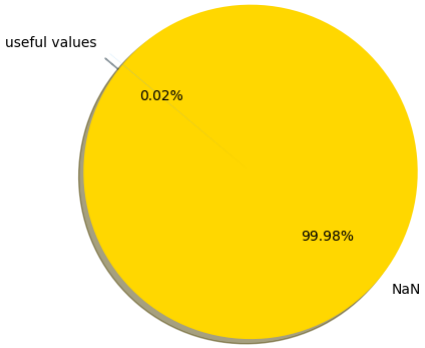


Figure 2.8: Error rate for each Section

Line 0 Categorical

The dataset obtained processing PreProcCategorical and isolating information related to Line 0, will be called L0Categorical. L0Categorical dataset contains 294 columns. In Figure 2.9 is shown the percentage of not NaN values in this particular Line.



useful values	NaN
0.02%	99.98%

Figure 2.9: Percentage of not NaN values in L0Categorical

There are really a few useful values. Since this dataset contains categorical values, it is necessary to study how the categorical labels are distributed. Different features can assume the same label value. In Figure 2.10 is shown the frequency of appearance for each label value, excluding NaN.

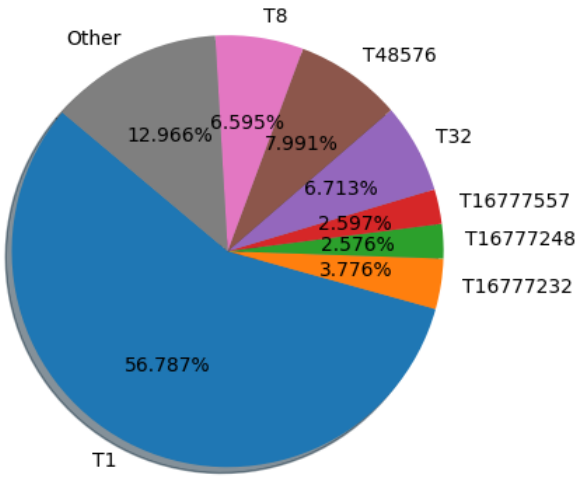


Figure 2.10: Labels with relative frequency

A few labels appear in the most of the cases, while the others appear very rarely. All the labels with an appearance frequency lower than 2.5%

have been grouped in 'Other'. To calculate the *error rate* for each label, has been used the same formula used for features in Numeric dataset:

$$err(L_i) = \frac{N_1(L_i)}{N(L_i)} \tag{2.2}$$

Where $N_1(L_i)$ indicates the number of rows where the label L_i appears and Response=1, while $N(L_i)$ indicates all the rows where label L_i appears. The result shown in Figure 2.11 is very different from the one obtained studying L0Numeric.

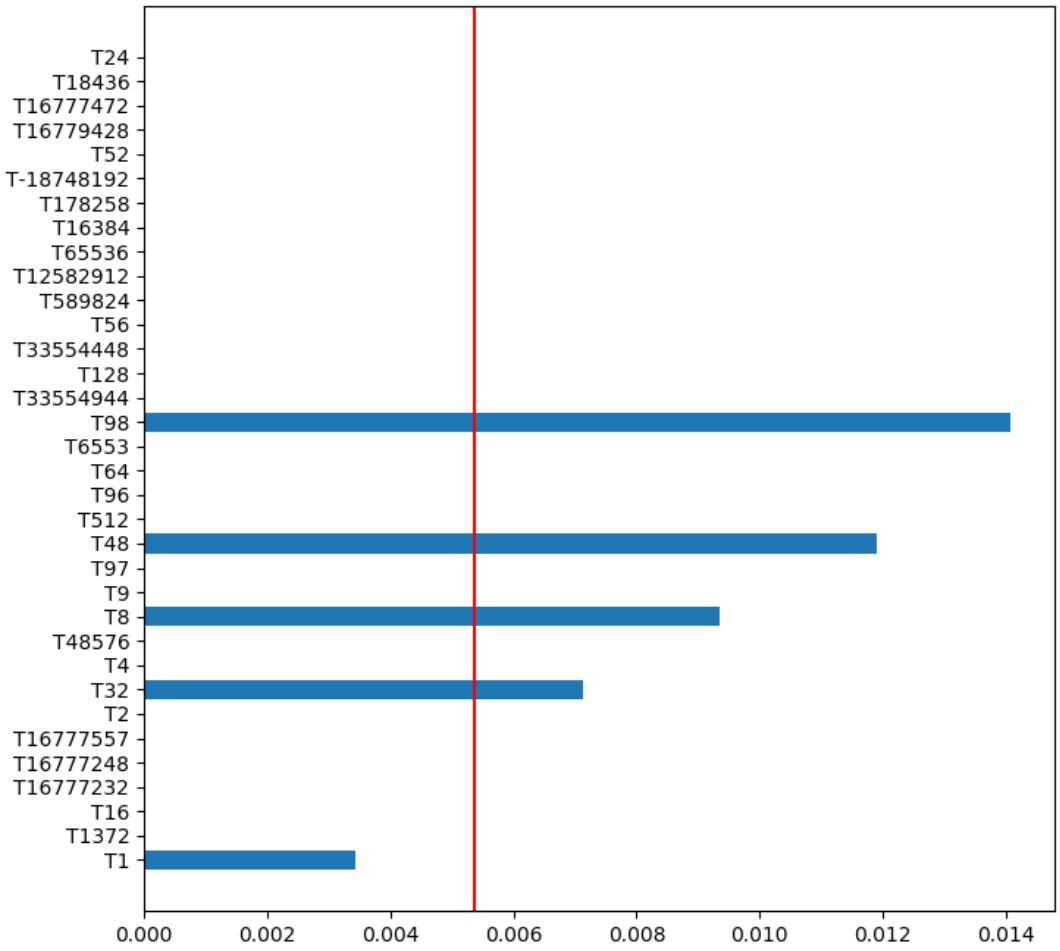


Figure 2.11: Error rate for each label

To explain the difference, we need to consider that the dataset contains

only 0.02% of not NaN values. Since also $Response = 1$ appears very rarely, we can suppose that the $error\_rate = 0$ of the majority of the labels is due to the fact that the simultaneous presence of two rare events is very improbable. With this presupposition, we can see that some label have interesting error rates, compared to the average of 0.53%:

- $T98$: $err(T98) = 1.4\%$, almost three times more than the average, but $N(T98) = 142$ (i.e. it compares only 142 times in all the dataset). It's not very useful.
- $T48$: $err(T48) = 1.2\%$, more than the double of the average and $N(T48) = 1054$. It may be useful.
- $T8$: $err(T8) = 0.9\%$ and $N(T8) = 2796$. It has an error rate a bit lower than $T48$, but it appears more frequently.
- $T32$: $err(T32) = 0.7\%$ and $N(T32) = 2846$. It appears with the same frequency of $T8$, but it has a lower error rate.
- $T1$: $err(T1) = 0.3\%$, about an half of the average and $N(T1) = 240762$. It is the label which appears with more frequency, it may be an useful index.

In Figure 2.12 we can see the error rate of each Section, calculated using (2.10).

Since each Section has a few rows with not NaN values, we need to compare the results with the number of useful rows in the dataset, indicated in Figure 2.13 by the column 'Samples'.

The most interesting Sections are:

- $S9$: $err(S9) = 0.4\%$,a bit lower than the average and $N(S9) = 496$
- $S4$: $err(S4) = 0.8\%$, higher that the average,but $N(S9) = 123$
- $S2$: $err(S2) = 1.1\%$, more than the double of the average, but $N(S2) = 263$

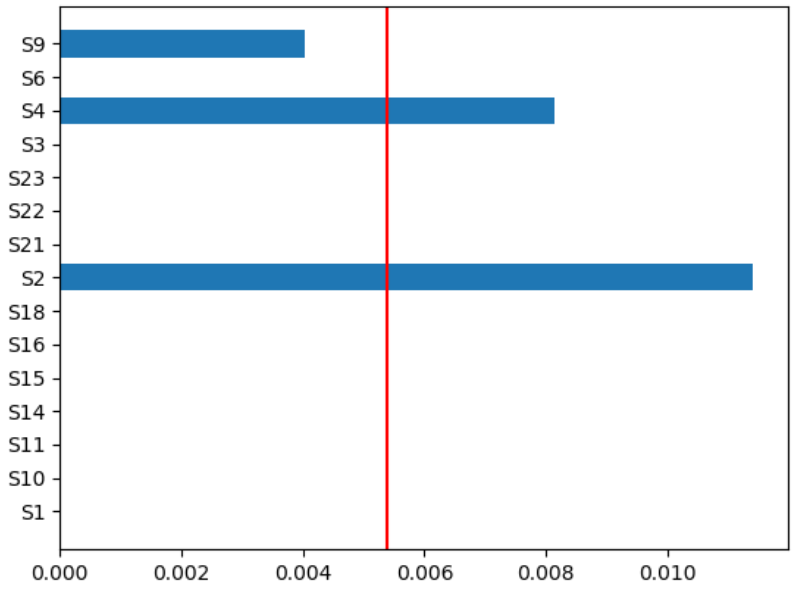


Figure 2.12: Error rate for each Section

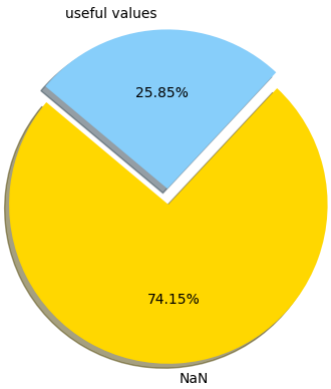
	Features	Samples	Error_Rate
S1	4	27	0.000000
S10	39	503	0.000000
S11	26	6	0.000000
S14	18	68	0.000000
S15	9	1	0.000000
S16	6	77	0.000000
S18	5	0	0.000000
S2	18	263	0.011407
S21	45	107	0.000000
S22	45	80	0.000000
S23	15	0	0.000000
S3	9	0	0.000000
S4	6	123	0.008130
S6	10	7	0.000000
S9	39	496	0.004032

Figure 2.13: Error rate and appearance frequency for each Section

- $S10$: $N(S10) = 503$, is the section that appears more often, but $err(S10) = 0\%$, it looks a very strange coincidence

Line 0 Date

The dataset obtained processing PreProcDate and isolating information related to Line 0, will be called L0Date. L0Date dataset contains 184 columns. In Figure 2.14 we can see the percentage of NaN values in this particular Line.



useful values	NaN
25.85%	74.15%

Figure 2.14: Percentage of NaN values in L0Date

The percentage is coherent with the other two datasets. An heatmap has been used to visualize data distribution. In Figure 2.15 is shown the plot of the first 100 rows.

As already noticed in L0Numeric, there is a clusterization by Section. We can also notice that each features in a Section has exactly the same value and it means that they have been recorded exactly in the same time. This fact supports the hypothesis that each Section contains the values of the sensors of a certain machinery. We can also see that values of different Sections in the same row defer a little, and this can mean that the part passes quite quickly from a machinery to the following. As example, in Figure 2.16 are plotted all the not NaN values of a row.

They only differ by some decimal point: while values in the same row are very similar, values in different rows have a great variability. In Figure 2.17 is shown the trend of the average value calculated row by row,over the first

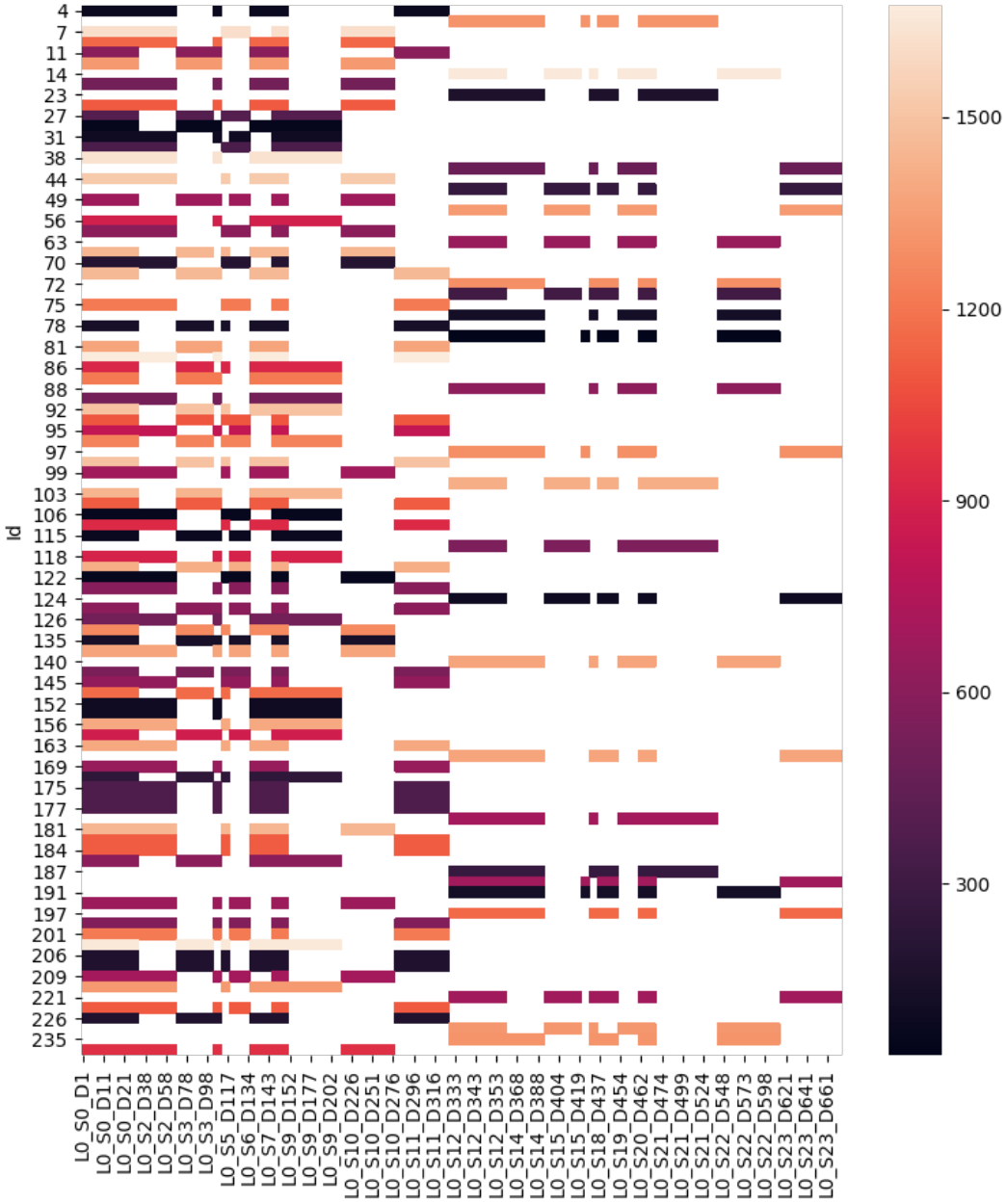


Figure 2.15: Heatmap of the first 100 rows of L0Date

100 Ids.

There is a great variance, that can be due to the different elaboration time for different products.

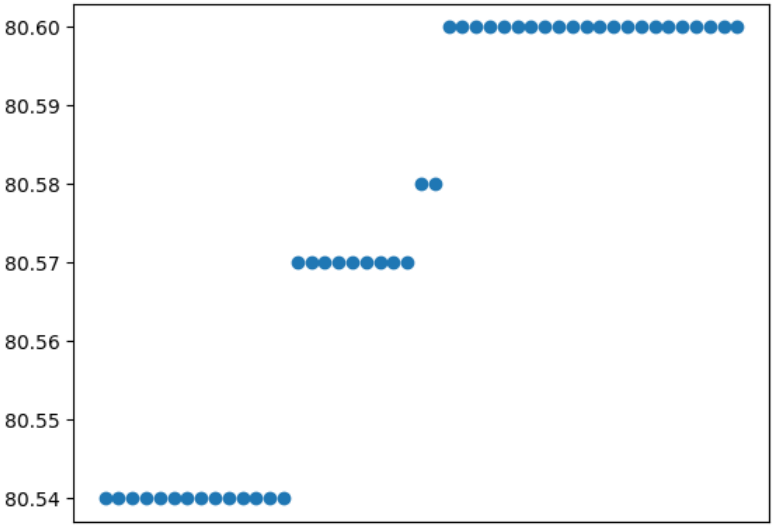


Figure 2.16: Plot of not NaN values of a row in L0Date

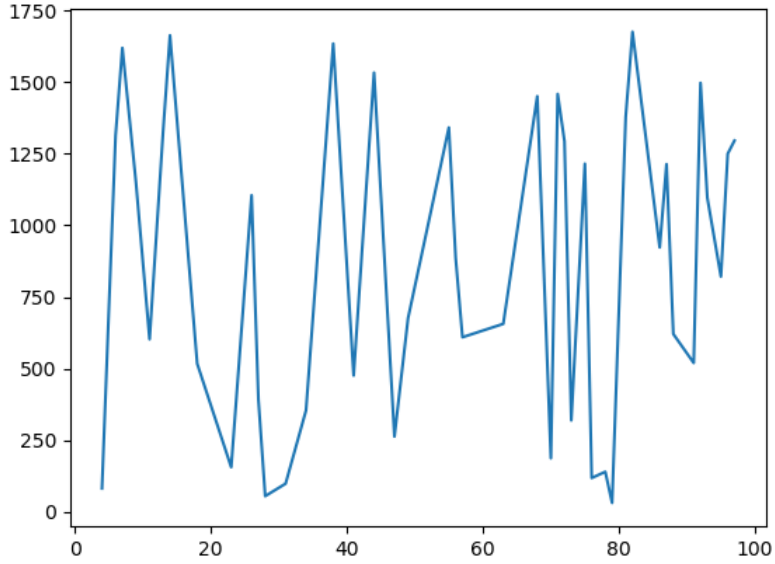


Figure 2.17: Plot of the average values of the first 100 rows of L0Date

Heatmaps

L0Date dataset should give a numeric time value for each value recorded in L0Numeric and L0Categorical, but dimensions suggest that a lot of data are missing. L0Numeric contains 169 columns and L0Categorical contains 294 columns, hence L0Date should contain $169 + 294 = 463$ columns, while it only contains 184. In this section the three datasets will be compared qualitatively, using heatmaps and studying the presence/absence of data.

L0Numeric vs L0Date

The purpose of this heatmap is to to check if each value in L0Numeric has a correspondent timestamp in L0Date. Comparing L0Numeric and L0Date datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L0Numeric and L0Date cells contain not NaN values (Green)
- L0Date=NaN, L0Numeric filled: L0Numeric cell contains a numeric value, while the corresponding cell in L0Date is NaN (Blue)
- L0Date not exists, L0Numeric filled: L0Numeric cell contains a numeric value, while the corresponding cell in L0Date doesn't exist (Yellow)
- L0Date filled, L0Numeric=NaN: L0Numeric cell contains a NaN value, while the corresponding cell in L0Date contains a not NaN value (Red)
- L0Numeric=Nan, L0Date=NaN/not exists: Both cells contains NaN or L0Numeric contains NaN, while L0Date cell doesn't exists (No color)

In Figure 2.18 is shown the result obtained plotting the first 1000 rows.

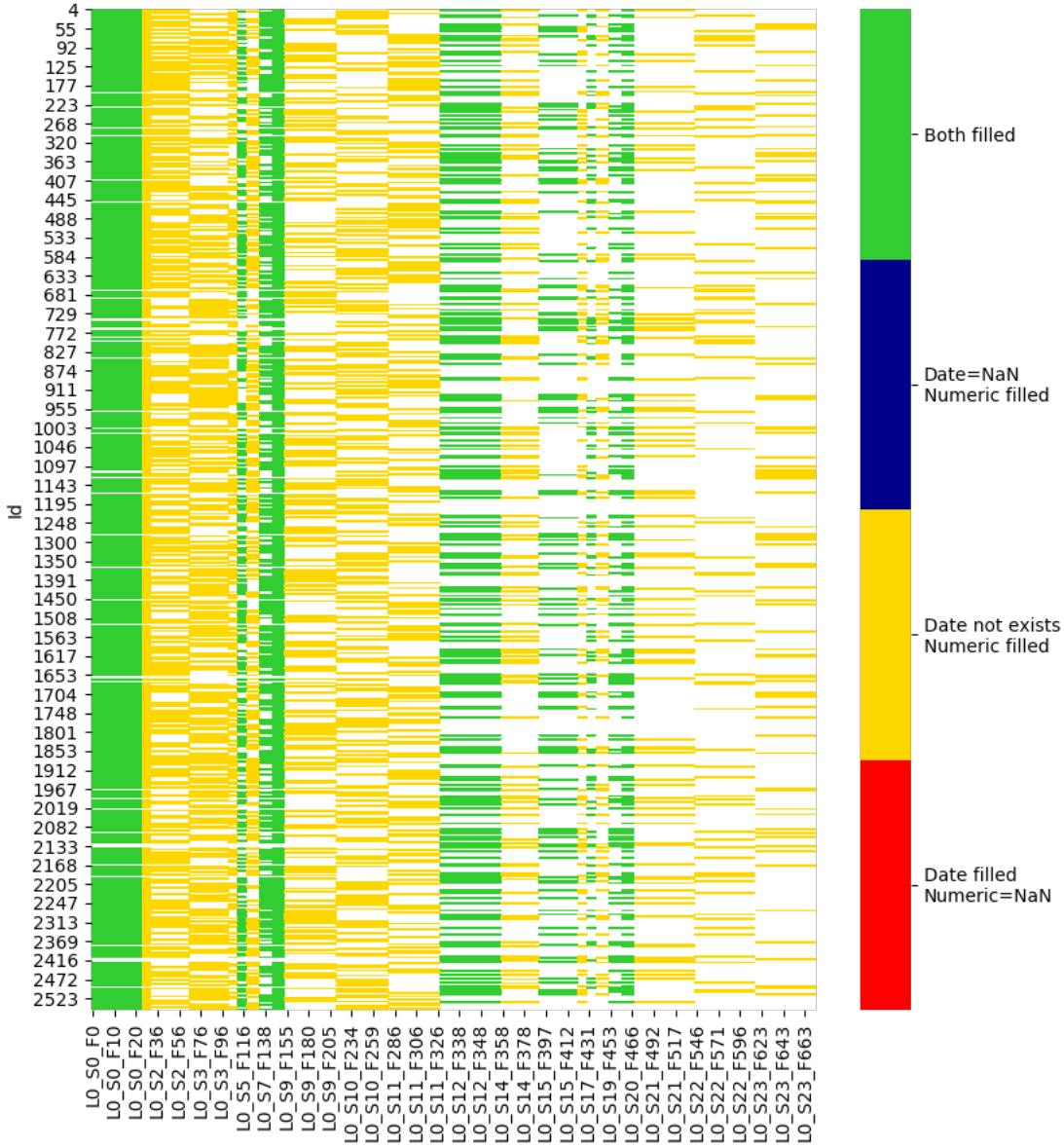


Figure 2.18: L0Numeric vs L0Date heatmap

Beside the both NaN case, there are only two situations: "both filled" and "Numeric filled and Date doesn't exist". In a situation of perfect information, we should only find the "both filled" case, but as we can see, not every feature in L0Numeric has a correspondent timestamp. This may be due to the fact that recording a timestamp for each feature is wasteful, since a lot of features are recorded in the same time or simply because a

part of the instrumentation is old and it hasn't the possibility to record a time value.

L0Categorical vs L0Date

The purpose of this heatmap is to check if each value in L0Categorical has a correspondent timestamp in L0Date. As seen before, comparing L0Categorical to L0Date dataset, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L0Categorical and L0Date cells contain not NaN values (Green)
- L0Date=NaN, L0Categorical filled: L0Categorical cell contains a not NaN value, while the corresponding cell in L0Date is NaN (Blue)
- L0Date not exists, L0Categorical filled: L0Categorical cell contains a not NaN value, while the corresponding cell in L0Date doesn't exist (Yellow)
- L0Date filled, L0Categorical=NaN: L0Categorical cell contains a NaN value, while the corresponding cell in L0Date contains a not NaN value (Red)
- L0Categorical=Nan, L0Date=NaN/not exists: Both cells contains NaN or L0Categorical contains NaN, while L0Date cell doesn't exists (No color)

In Figure 2.19 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, the situations "both filled" and "Categorical filled and Date doesn't exist" appear very rarely, while the the case "Date filled and Categorical=NaN" appears in the most of the cases. This situation can be explained only supposing that categorical sensor can give NaN as output. This means that in Categorical dataset we can find two different "kind of NaN":

- Case 1: NaN means "no data have been collected". In this case, the corresponding value in Date is also "NaN

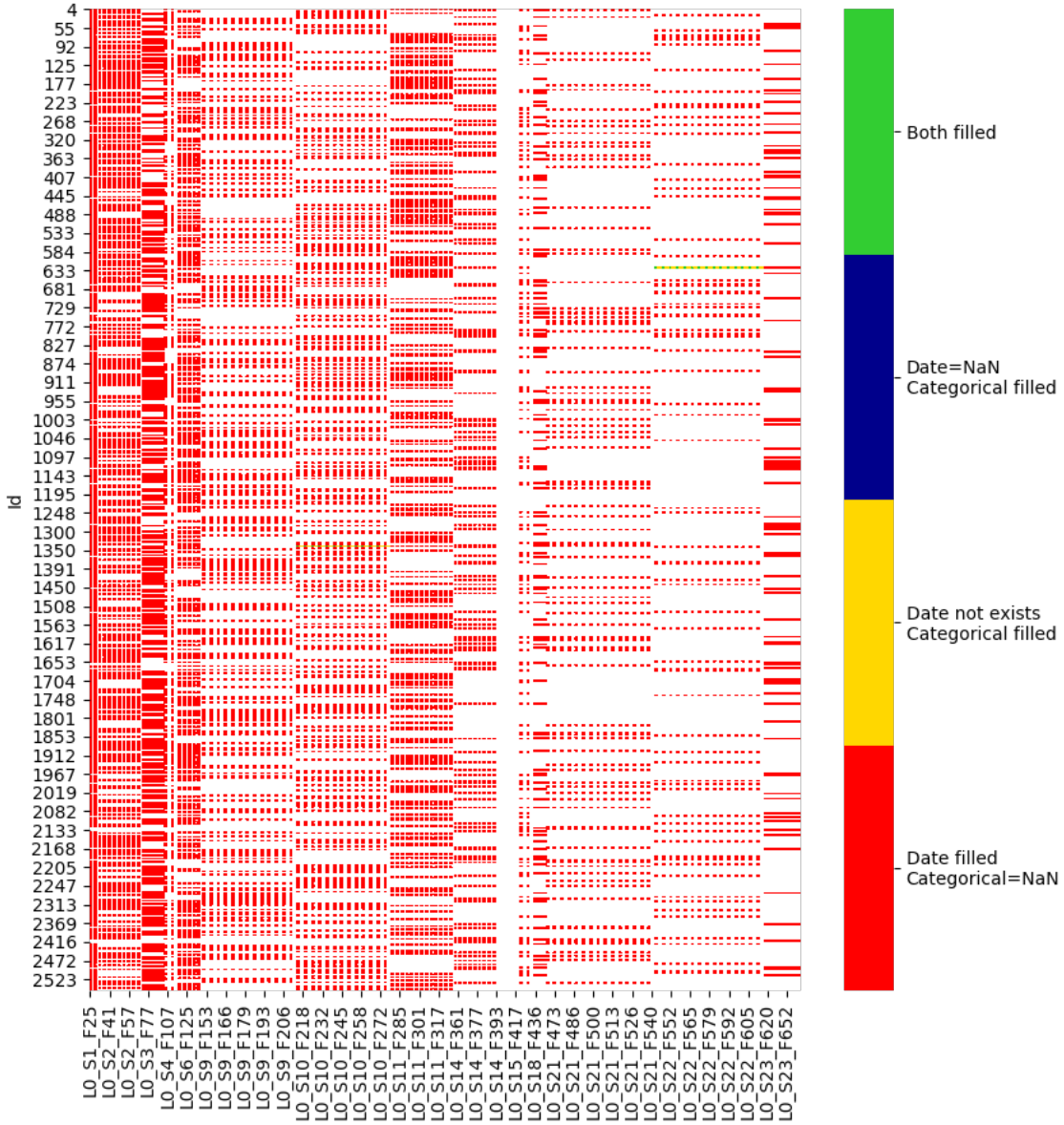


Figure 2.19: L0Categorical vs L0Date heatmap

- Case 2: NaN means "data with NaN value has been collected". In this case, the corresponding value in Date is not NaN

In a situation of perfect information, we could easily distinguish the two different kind of NaN by checking whether the corresponding value in Date is NaN or not, but in this case, a part of the feature haven't a corresponding column in Date, making a perfect discrimination impossible.

L0Date vs L0Numeric & L0Categorical

The purpose of this heatmap is to check if each value in L0Date is the timestamp of a correspondent value in L0Numeric or L0Categorical. We need remember that the three datasets only have *Id* as common column, so each column of L0Date can refer only to one column in L0Numeric or L0Categorical, but not to both of them. As seen before, comparing L0Date to L0Numeric and L0Categorical datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: L0Date and one of L0Numeric or L0Categorical cell contain not NaN values (while the other doesn't exist) (Green)
- L0Numeric or L0Categorical=NaN, L0Date filled: L0Date cell contains a not NaN value, while one corresponding cell in L0Numeric or L0Categorical is NaN (while the other doesn't exist) (Blue)
- Both L0Numeric and L0Categorical do not exist, L0Date filled: L0Date cell contains a not NaN value, while the corresponding cell in L0Numerical and L0Categorical doesn't exist (Yellow)
- L0Numeric or L0Categorical filled, L0Date=NaN: L0Date cell contains a NaN value, while the corresponding cell in L0Numeric or L0Categorical contain a not NaN value (Red)
- L0Date=Nan, L0Numeric and L0Categorical=NaN/not exist: L0Date cell contains NaN, while L0Numeric and L0Categorical contain NaN or don't exist (No color)

In Figure 2.20 is shown the result obtained plotting the first 1000 rows.

The result obtained is almost what we could expect looking at the heatmaps before. The green columns are due to the result obtained in "L0Numeric vs L0Date", while the blue ones are explained in "L0Categorical vs L0Date". We can see that there are not red cells: it means that every not NaN value in L0Numeric and L0Categorical has a correspondent not

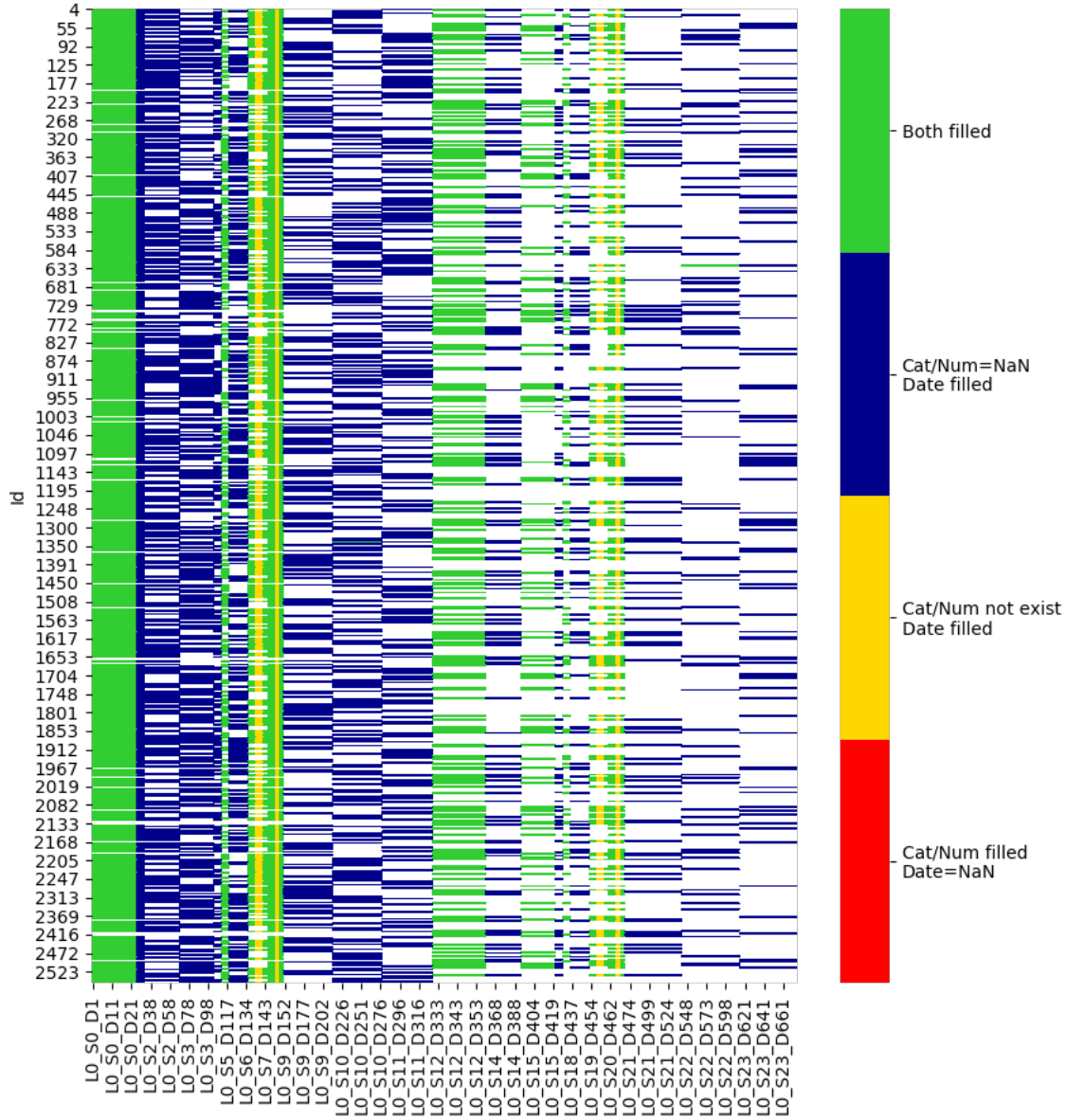


Figure 2.20: L0Date vs L0Numeric & L0Categorical heatmap

NaN timestamp in L0Date, if the related column exists. The only exception we can see are some yellow cells that represent columns in L0Date without any correspondence in L0Numeric or L0Categorical. At the moment can't be find any good explanation for this phenomenon except that data inconsistency.

Correlation test

The purpose of this analysis is to find any kind of correlation between different Sections. Since almost each Section has features in both L0Numeric and L0Categorical, each dataset will be studied separately, then results will be compared.

Given a and b two Sections, the index called $confidence$ is defined as follows:

$$conf(a,b) = P(a|b) = \frac{\#\{a,b\}}{\#\{a\}} \tag{2.3}$$

It defines the ratio between the times Sections a and b are active in the same row and the times Section a is active. In particular, $conf(a,b)$ measures how much $b \Rightarrow a$ namely, how much the presence of a implies the presence of b. This index has values in [0,1] and is used for finding association rules between a and b: if $conf(a,b) \sim 1$, then the presence of a means that b has high probability to appear, while $conf(a,b) \sim 0$ means that the presence of a means that b has low probability to appear. We need to observe that the index isn't symmetric, in fact $conf(a,b) \neq conf(b,a)$.

The characteristic of confidence index is that it is influenced by relative frequencies of the events a and b. For example, if $\#\{a\} = 1000$, $\#\{b\} = 10$ and $\#\{a,b\} = 10$, we can see a strong relation between a and b, because b happens if and only if happens a, then $conf(b,a) = 1$, while $conf(a,b) = 0.01$, so we can understand that $b \Rightarrow a$, but $a \not\Rightarrow b$.

L0Numeric

In this preliminary analysis will be studied when each Section assumes not NaN values, not studying the particular values assumed, but only considering the cases active/not active. The confidence index (2.12) between each Section is plotted using an heatmap in Figure 2.21.

There are couples of Sections with negative association (i.e. if a is active, then b isn't active and vice versa), such as $S14$ and $S15$, $S16$ and $S17$ and

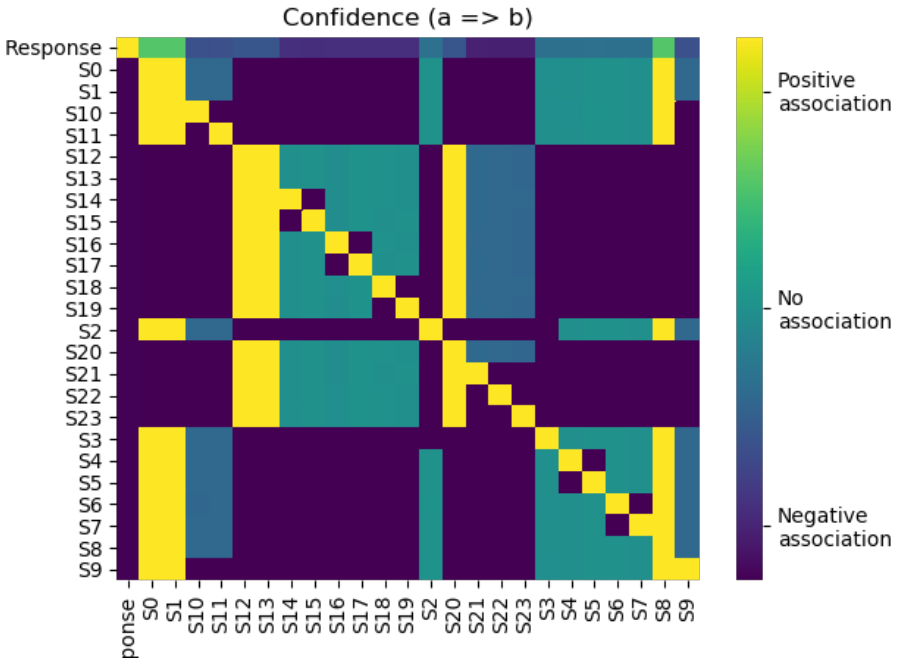


Figure 2.21: Confidence between Sections in L0Numeric

so on. We can suppose that these Sections could represent machinery that work in parallel or machinery with mutually exclusive utility (i.e. paintings of different colors). Response has also been plotted, but the associations we see aren't strong enough to be significant.

L0Categorical

In Figure 2.22, relation between Sections in L0Categorical is studied using (2.12).

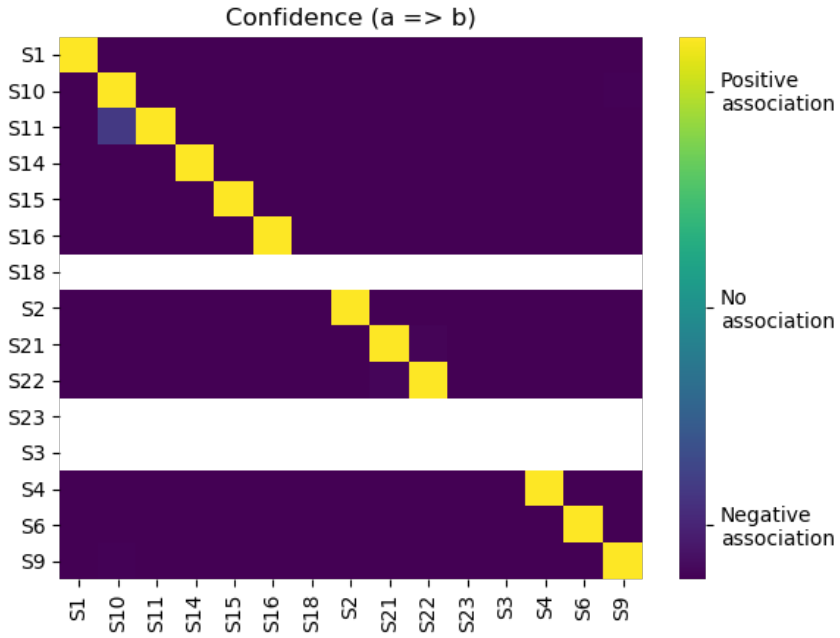


Figure 2.22: Confidence between Sections in L0Categorical

The result is less significant than the one obtained with L0Numeric, since there is a greater number of NaN values. In particular, we can see some Sections without values: this is due to the fact that some of the NaN values have a correspondent not NaN timestamp, so the Sections can't be eliminated, because the NaN we see are an output of the machinery, not an 'absence of information'.

L0Date

The relation between Sections in L0Date is studied in Figure 2.23 using (2.12).

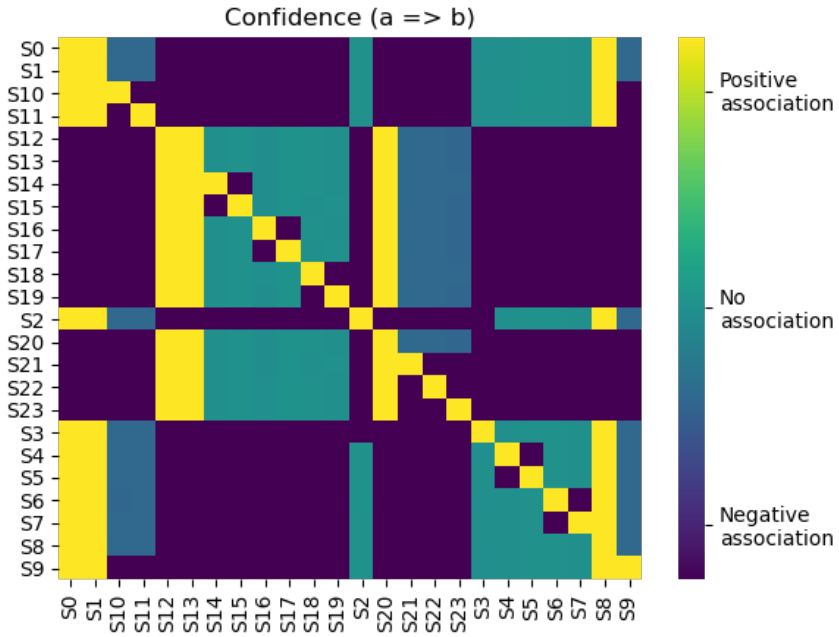


Figure 2.23: Confidence between Sections in L0Date

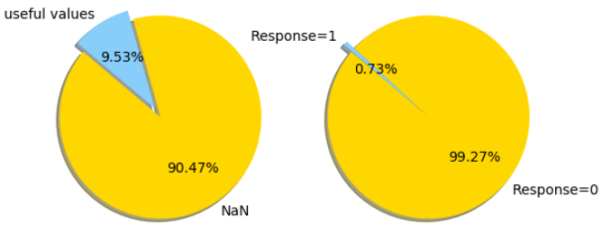
As expected, the result obtained confirms what already noticed studying L0Numeric.

2.3.2 Line 1

After the cleaning process, each dataset contains 267273 rows related to *Line*1, indexed by the common primary key *Id*. Each file will be initially studied separately, then will be studied how the three dataset are each other related.

Line 1 Numeric

The dataset obtained processing PreProcNumeric and isolating information related to Line 1, will be called L1Numeric. L1Numeric contains 514 columns. In Figure 2.24 is shown the result obtained calculating the percentage of not NaN values and Response=1 rows in this particular Line.



useful values	NaN	Response=1	Response=0
9.53%	90.47%	0.73%	99.27%

Figure 2.24: Percentage of not NaN values and Response=1 rows in L1Numeric

The number of not NaN values is lower than L0Numeric, but Response=1 are slightly more frequent. To visualize the distribution of NaN values over the dataset, has been used an heatmap: a different color is assigned for each numeric value, while the NaN values are left without color. In Figure 2.25 is shown the result obtained plotting the first 100 rows.

NaN values are not random distributed. As already noticed, features look to be gathered in clusters within which, they have a similar behavior: or all they are NaN, or all they assume a numerical value. Looking at the identification codes, we can understand that features with the same behavior belong to the same Section but, differently from Line 0, in the same Section is possible to find more groups of features. Since there are only two big Sections in this Line, we can suppose a Section to be a very big machinery, containing various working stations: since the machinery is

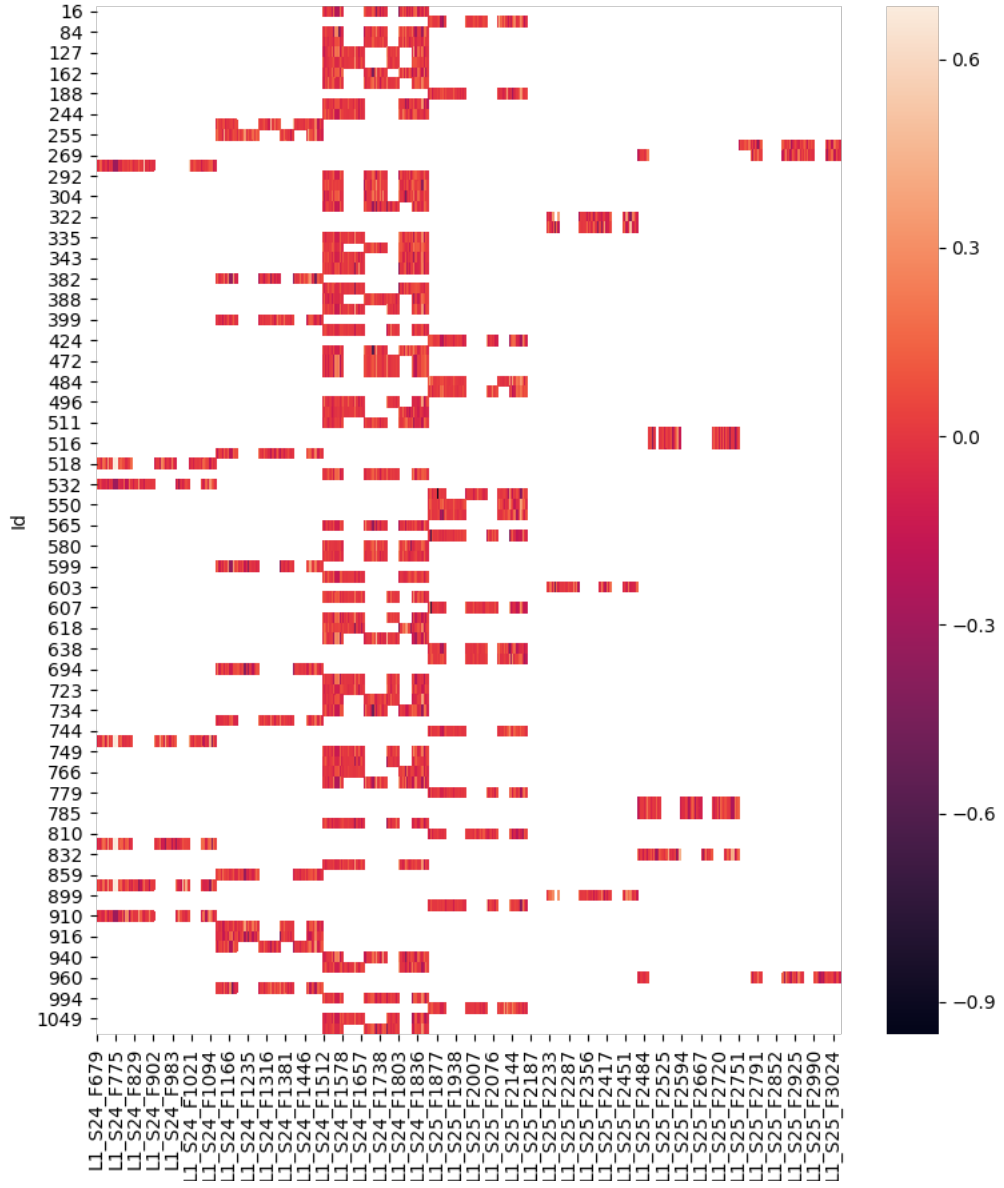


Figure 2.25: Heatmap of the first 100 rows of L1Numeric

very big, parts doesn't pass through every working station, this means that there are various groups of sensors with different recording time. With this mind, we can study the relation between Sections and Response.

A Section affects the value of Response only if the part passes through the machinery it represents, i.e. only if its features assume not NaN values.

In order to find a correlation between the error and the activation of a particular section, features have been grouped by Section, then can be defined an index, that will be called *error rate* of the Section S_i:

$$err(S_i) = \frac{N_1(S_i)}{N(S_i)} \tag{2.4}$$

Where $N_1(S_i)$ indicates the number of rows where the Section S_i has not NaN values and Response=1, $N(S_i)$ indicates all the rows where the Section S_i has not NaN values.

The average error rate $\overline{err}$ of the dataset coincides with the percentage of Response=1 described above. If $err(S_i)$ is bigger than the average, the probability of Response=1 is higher when the Section S_i is active (i.e. assumes not NaN values), vice versa the probability is lower if the $err(S_i)$ is lower than average. In Figure 2.26 has been calculated the $err(S_i)$ for each Section in Line 1 and it has been compared with the average $\overline{err} = 0.00725$ (red line).

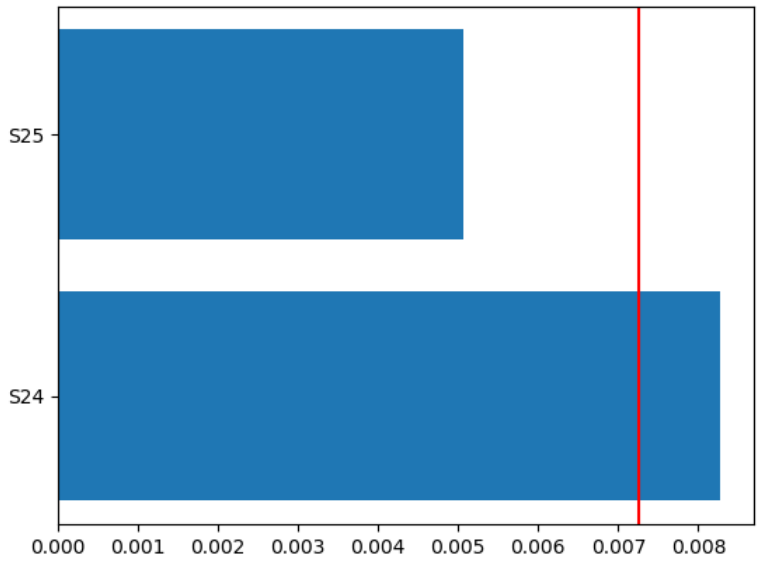
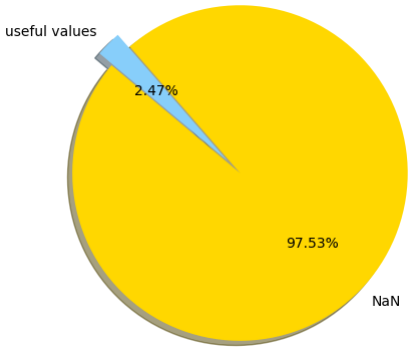


Figure 2.26: Error rate for each Section

There are only two big Sections, it is possible to notice that Section $S25$ has approximately half of the error of $S24$.

Line 1 Categorical

The dataset obtained processing PreProcCategorical and isolating information related to Line 1, will be called L1Categorical. L1Categorical dataset contains 1146 columns. In Figure 2.27 is shown the percentage of not NaN values in this particular Line.



useful values	NaN
2.47%	97.53%

Figure 2.27: Percentage of not NaN values in L1Categorical

There more useful values in this Line than in Line 0, however the percentage is very low. Since this dataset contains categorical values, it is necessary to study how the categorical labels are distributed. Different features can assume the same label value. In Figure 2.28 is shown the frequency of appearance for each label value, excluding NaN.

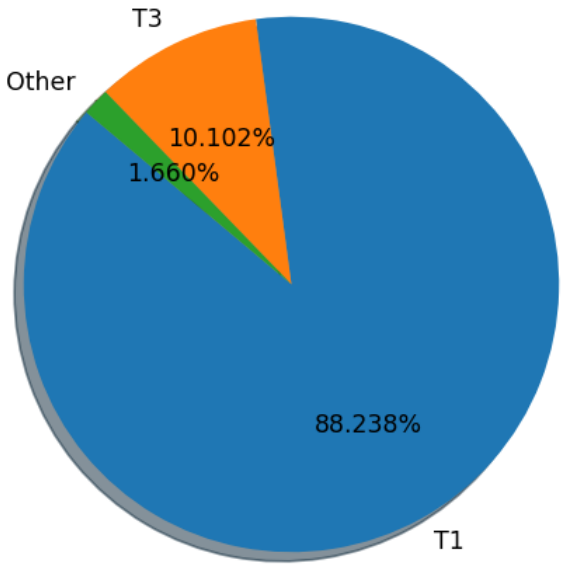


Figure 2.28: Labels with relative frequency

Only 2 labels appear in the most of the cases, while the others appear very rarely. All the labels with an appearance frequency lower than 1% have been grouped in 'Other'. To calculate the *error rate* for each label, has been used the same formula used for features in Numeric dataset:

$$err(L_i) = \frac{N_1(L_i)}{N(L_i)} \tag{2.5}$$

Where $N_1(L_i)$ indicates the number of rows where the label L_i appears and Response=1, while $N(L_i)$ indicates all the rows where label L_i appears. The result shown in Figure 2.29 is very different from the one obtained studying L1Numeric.

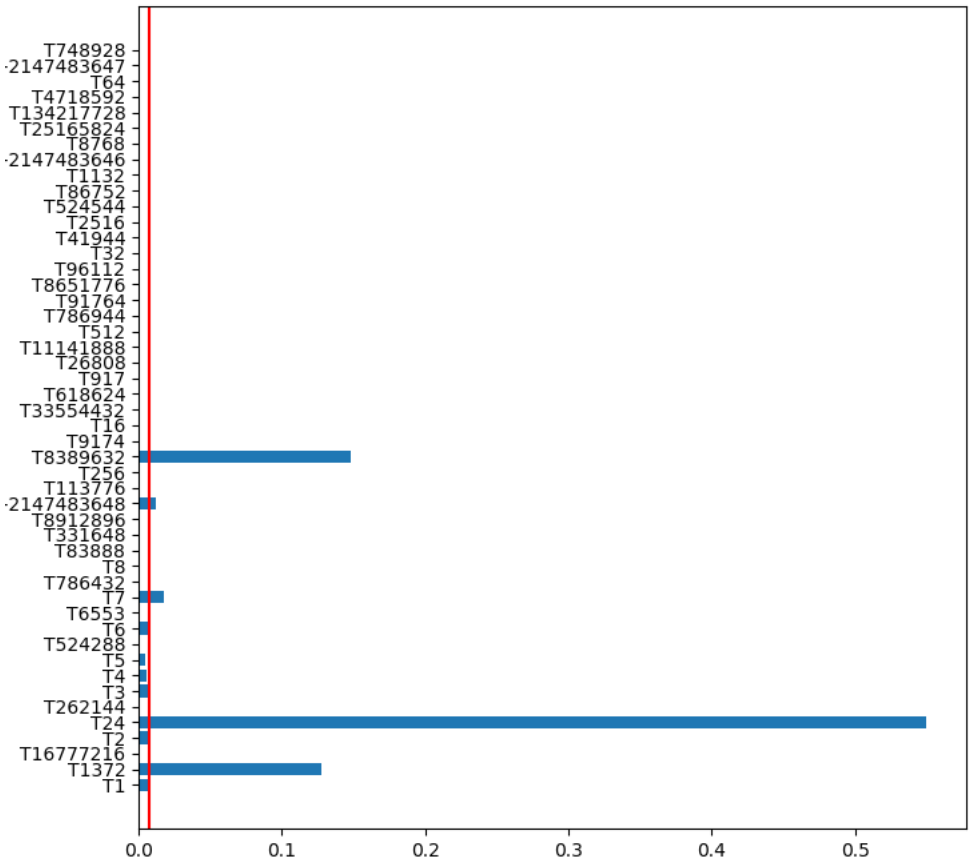


Figure 2.29: Error rate for each label

To explain the difference, we need to consider that the dataset contains

only 2.47% of not NaN values. Since also $Response = 1$ appears very rarely, we can suppose that the $error\_rate = 0$ of the majority of the labels is due to the fact that the simultaneous presence of two rare events is very improbable. With this presupposition, we can see that some label have interesting error rates, compared to the average of 0.72%:

- $T24$: $err(T98) = 55\%$, very high, compared to the average, but $N(T24) = 380$. It's particularly useful.
- $T1372$: $err(T48) = 12.7\%$, more than 17 times the average and $N(T1372) = 1805$. It is useful.
- $T8389632$: $err(T8) = 14.8\%$ and $N(8389632) = 513$. It has an error rate a bit higher than $T1372$, but it appears less frequently.

In Figure 2.30 we can see the error rate of each Section, calculated using (2.10).

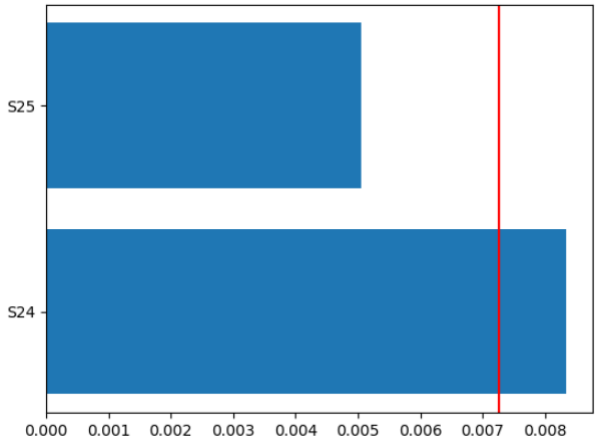


Figure 2.30: Error rate for each Section

Since there are only 2 Sections in this Line, the result is very similar to the one obtained studying L1Numeric.

Line 1 Date

The dataset obtained processing PreProcDate and isolating information related to Line 1, will be called L1Date. L1Date dataset contains 621 columns. In Figure 2.31 we can see the percentage of NaN values in this particular Line.

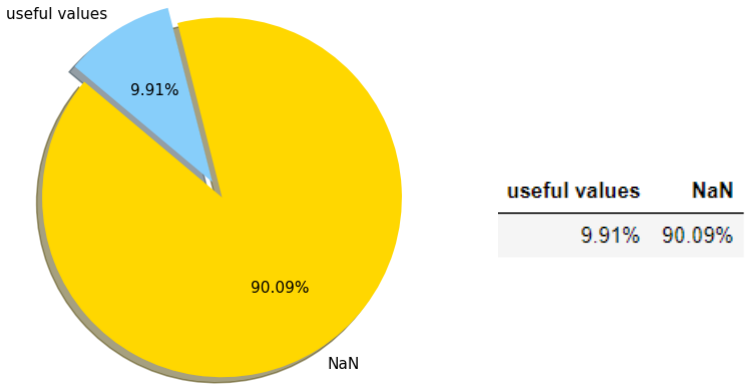


Figure 2.31: Percentage of NaN values in L1Date

The percentage is coherent with the other two datasets. An heatmap has been used to visualize data distribution. In Figure 2.32 is shown the plot of the first 100 rows.

As already noticed in L1Numeric, there is a clusterization by Section. We can also notice that each features in a Section has exactly the same value and it means that they have been recorded exactly in the same time. This fact supports the hypothesis that each Section contains the values of the sensors of a certain machinery. We can also see that values of different Sections in the same row defer a little, and this can mean that the part passes quite quickly from a machinery to the following. In Figure 2.33 is shown the trend of the average value calculated row by row,over the first 100 Ids.

There is a great variance, that can be due to the different elaboration time for different products.

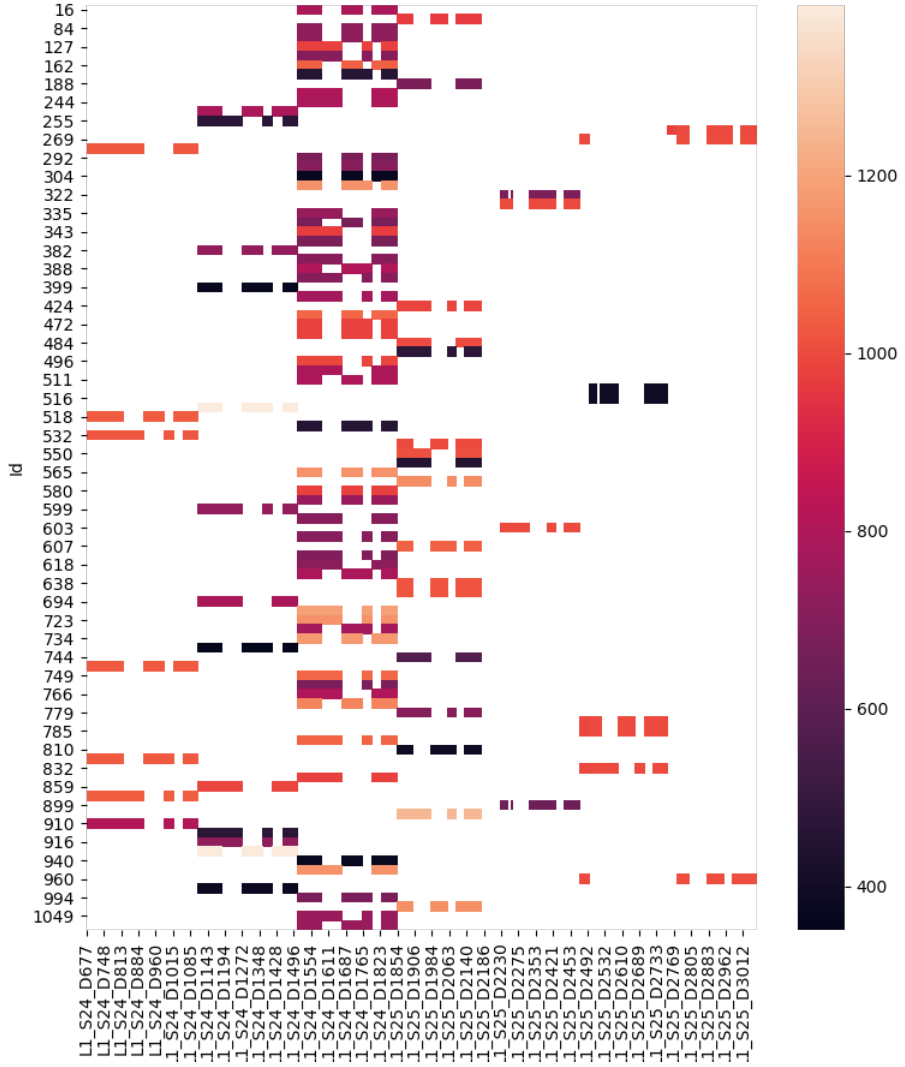


Figure 2.32: Heatmap of the first 100 rows of L1Date

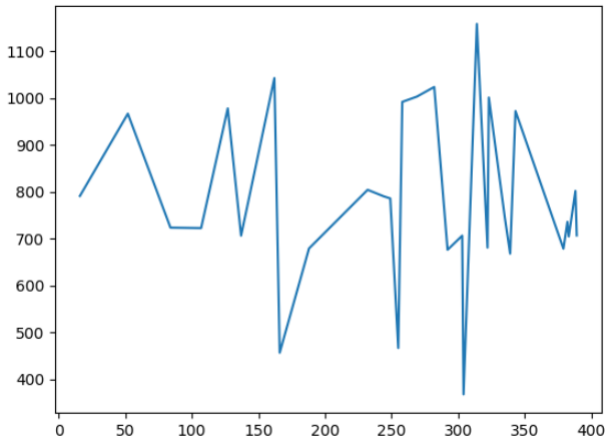


Figure 2.33: Plot of the average values of the first 100 rows of L1Date

Heatmaps

L1Date dataset should give a numeric time value for each value recorded in L1Numeric and L1Categorical, but dimensions suggest that a lot of data are missing. L1Numeric contains 514 columns and L1Categorical contains 621 columns, hence L1Date should contain $514 + 1146 = 1660$ columns, while it only contains 631. In this section the three datasets will be compared qualitatively, using heatmaps and studying the presence/absence of data.

L1Numeric vs L1Date

The purpose of this heatmap is to to check if each value in L1Numeric has a correspondent timestamp in L1Date. Comparing L1Numeric and L1Date datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L1Numeric and L1Date cells contain not NaN values (Green)
- L1Date=NaN, L1Numeric filled: L1Numeric cell contains a numeric value, while the corresponding cell in L1Date is NaN (Blue)
- L1Date not exists, L1Numeric filled: L1Numeric cell contains a numeric value, while the corresponding cell in L1Date doesn't exist (Yellow)
- L1Date filled, L1Numeric=NaN: L1Numeric cell contains a NaN value, while the corresponding cell in L1Date contains a not NaN value (Red)
- L1Numeric=Nan, L1Date=NaN/not exists: Both cells contains NaN or L1Numeric contains NaN, while L1Date cell doesn't exists (No color)

In Figure 2.34 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, that is more frequent in this Line than in Line 0, there are only two situations: "both filled" and "Numeric filled and Date

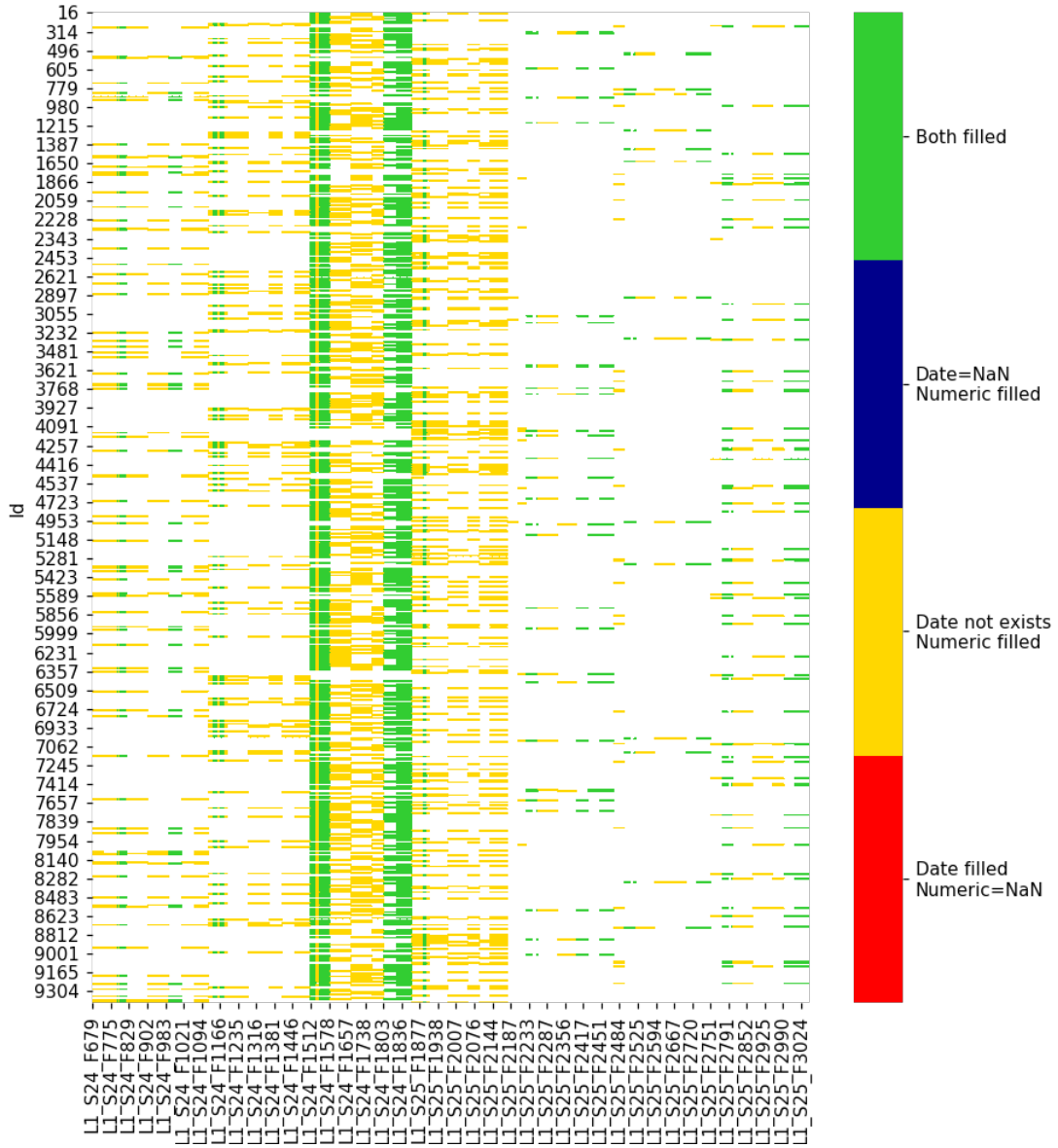


Figure 2.34: L1Numeric vs L1Date heatmap

doesn't exist". In a situation of perfect information, we should only find the "both filled" case, but as we can see, not every feature in L1Numeric has a correspondent timestamp. This may be due to the fact that recording a timestamp for each feature is wasteful, since a lot of features are recorded in the same time or simply because a part of the instrumentation is old and it hasn't the possibility to record a time value.

L1Categorical vs L1Date

The purpose of this heatmap is to check if each value in L1Categorical has a correspondent timestamp in L1Date. As seen before, comparing L1Categorical to L1Date dataset, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L1Categorical and L1Date cells contain not NaN values (Green)
- L1Date=NaN, L1Categorical filled: L1Categorical cell contains a not NaN value, while the corresponding cell in L1Date is NaN (Blue)
- L1Date not exists, L1Categorical filled: L1Categorical cell contains a not NaN value, while the corresponding cell in L1Date doesn't exist (Yellow)
- L1Date filled, L1Categorical=NaN: L1Categorical cell contains a NaN value, while the corresponding cell in L1Date contains a not NaN value (Red)
- L1Categorical=Nan, L1Date=NaN/not exists: Both cells contains NaN or L1Categorical contains NaN, while L1Date cell doesn't exists (No color)

In Figure 2.35 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, the situations "both filled" and "Categorical filled and Date doesn't exist" appear very rarely, while the the case "Date filled and Categorical=NaN" appears in the most of the cases. This situation can be explained only supposing that categorical sensor can give NaN as output. This means that in Categorical dataset we can find two different "kind of NaN":

- Case 1: NaN means "no data have been collected". In this case, the corresponding value in Date is also "NaN

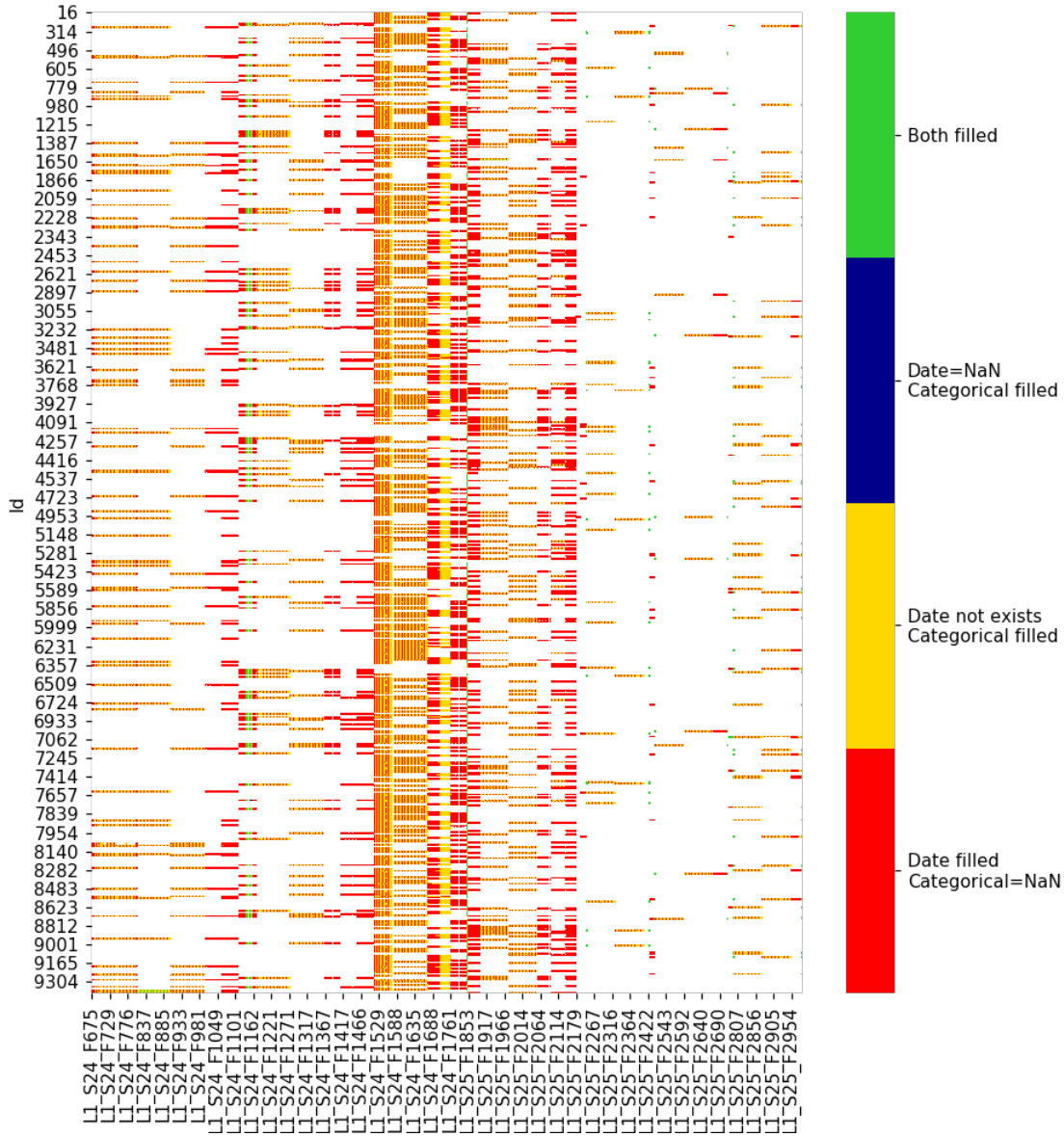


Figure 2.35: L1Categorical vs L1Date heatmap

- Case 2: NaN means "data with NaN value has been collected". In this case, the corresponding value in Date is not NaN

In a situation of perfect information, we could easily distinguish the two different kind of NaN by checking whether the corresponding value in Date is NaN or not, but in this case, a part of the feature haven't a corresponding column in Date, making a perfect discrimination impossible.

L1Date vs L1Numeric & L1Categorical

The purpose of this heatmap is to check if each value in L1Date is the timestamp of a correspondent value in L1Numeric or L1Categorical. We need remember that the three datasets only have *Id* as common column, so each column of L1Date can refer only to one column in L1Numeric or L1Categorical, but not to both of them. As seen before, comparing L1Date to L1Numeric and L1Categorical datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: L1Date and one of L1Numeric or L1Categorical cell contain not NaN values (while the other doesn't exist) (Green)
- L1Numeric or L1Categorical=NaN, L1Date filled: L1Date cell contains a not NaN value, while one corresponding cell in L1Numeric or L1Categorical is NaN (while the other doesn't exist) (Blue)
- Both L1Numeric and L1Categorical do not exist, L1Date filled: L1Date cell contains a not NaN value, while the corresponding cell in L1Numerical and L1Categorical doesn't exist (Yellow)
- L1Numeric or L1Categorical filled, L1Date=NaN: L1Date cell contains a NaN value, while the corresponding cell in L1Numeric or L1Categorical contain a not NaN value (Red)
- L1Date=Nan, L1Numeric and L1Categorical=NaN/not exist: L1Date cell contains NaN, while L1Numeric and L1Categorical contain NaN or don't exist (No color)

In Figure 2.36 is shown the result obtained plotting the first 1000 rows.

The result obtained is almost what we could expect looking at the heatmaps before. The green columns are mainly due to the result obtained in "L1Numeric vs L1Date", while the blue ones are explained in "L1Categorical vs L1Date". We can see that there are not red cells: it means that every not NaN value in L1Numeric and L1Categorical has a

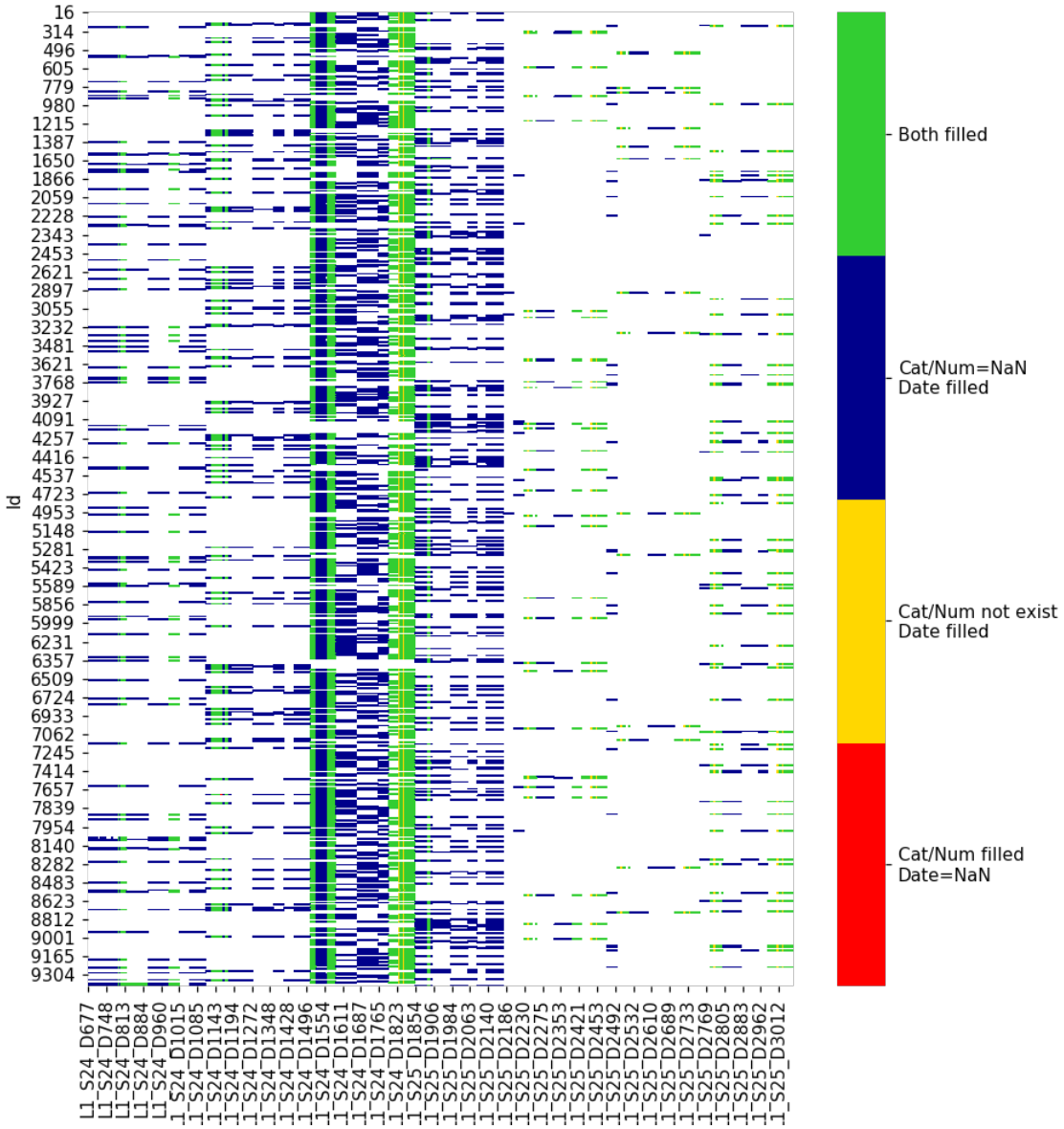


Figure 2.36: L1Date vs L1Numeric & L1Categorical heatmap

correspondent not NaN timestamp in L1Date, if the related column exists. The only exception we can see are some yellow cells that represent columns in L1Date without any correspondence in L1Numeric or L1Categorical. At the moment can't be find any good explanation for this phenomenon except that data inconsistency.

Correlation test

The purpose of this analysis is to find any kind of correlation between different Sections. Since almost each Section has features in both L1Numeric and L1Categorical, each dataset will be studied separately, then results will be compared.

Given a and b two Sections, the index called $confidence$ is defined as follows:

$$conf(a,b) = P(a|b) = \frac{\#\{a,b\}}{\#\{a\}} \tag{2.6}$$

It defines the ratio between the times Sections a and b are active in the same row and the times Section a is active. In particular, $conf(a,b)$ measures how much $b \Rightarrow a$ namely, how much the presence of a implies the presence of b. This index has values in [0,1] and is used for finding association rules between a and b: if $conf(a,b) \sim 1$, then the presence of a means that b has high probability to appear, while $conf(a,b) \sim 0$ means that the presence of a means that b has low probability to appear. We need to observe that the index isn't symmetric, in fact $conf(a,b) \neq conf(b,a)$.

The characteristic of confidence index is that it is influenced by relative frequencies of the events a and b. For example, if $\#\{a\} = 1000$, $\#\{b\} = 10$ and $\#\{a,b\} = 10$, we can see a strong relation between a and b, because b happens if and only if happens a, then $conf(b,a) = 1$, while $conf(a,b) = 0.01$, so we can understand that $b \Rightarrow a$, but $a \not\Rightarrow b$.

Since Line 1 contains only two Sections, studying correlation for each dataset is unnecessary, because the result obtained is always the same.

In this preliminary analysis will be studied when each Section assumes not NaN values, not studying the particular values assumed, but only considering the cases active/not active. The confidence index (2.12) between each Section is plotted using an heatmap in Figure 2.37.

The two Sections have a strong negative correlation, thus if a part passes through one of the Sections, the probability of passing also through the other is very low.

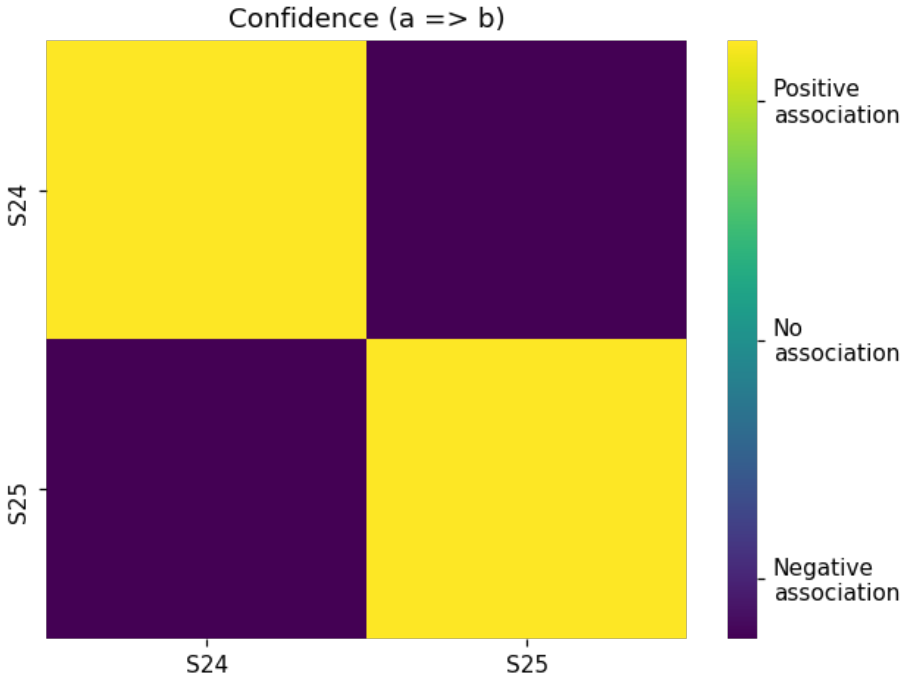


Figure 2.37: Confidence between Sections

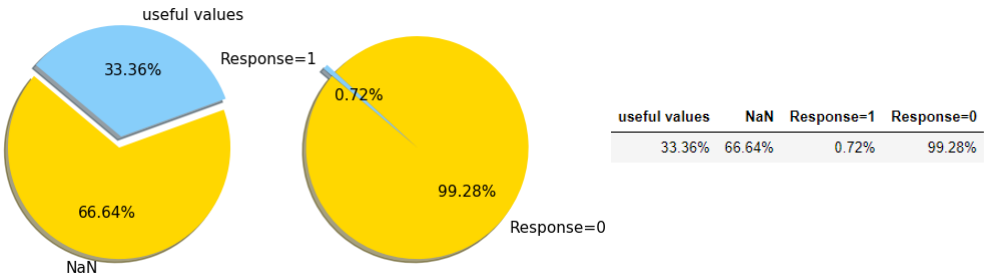
There are only two sections, then studying correlation of L1Categorical and L1Numeric dataset would give exactly the same results.

2.3.3 Line 2

After the cleaning process, each dataset contains 357019 rows related to *Line*2, indexed by the common primary key *Id*. Each file will be initially studied separately, then will be studied how the three dataset are each other related.

Line 2 Numeric

The dataset obtained processing PreProcNumeric and isolating information related to Line 2, will be called L2Numeric. L2Numeric contains 43 columns. In Figure 2.38 is shown the result obtained calculating the percentage of not NaN values and Response=1 rows in this particular Line.



useful values	NaN	Response=1	Response=0
33.36%	66.64%	0.72%	99.28%

Figure 2.38: Percentage of not NaN values and Response=1 rows in L2Numeric

The number of not NaN values is slightly lower than than in the other lines, but they are still very frequent. To visualize the distribution of NaN values over the dataset, has been used an heatmap: a different color is assigned for each numeric value, while the NaN values are left without color. In Figure 2.39 is shown the result obtained plotting the first 100 rows.

NaN values are not random distributed. In fact, features look to be gathered in clusters within which, they have a similar behavior: or all they are NaN, or all they assume a numerical value. Looking at the identification codes, we can understand that features with the same behavior belong to the same Section. We can suppose a Section to be a particular machinery, containing various sensors: each sensor is represented by a specific feature in the dataset. This theory explains why features in the same section have

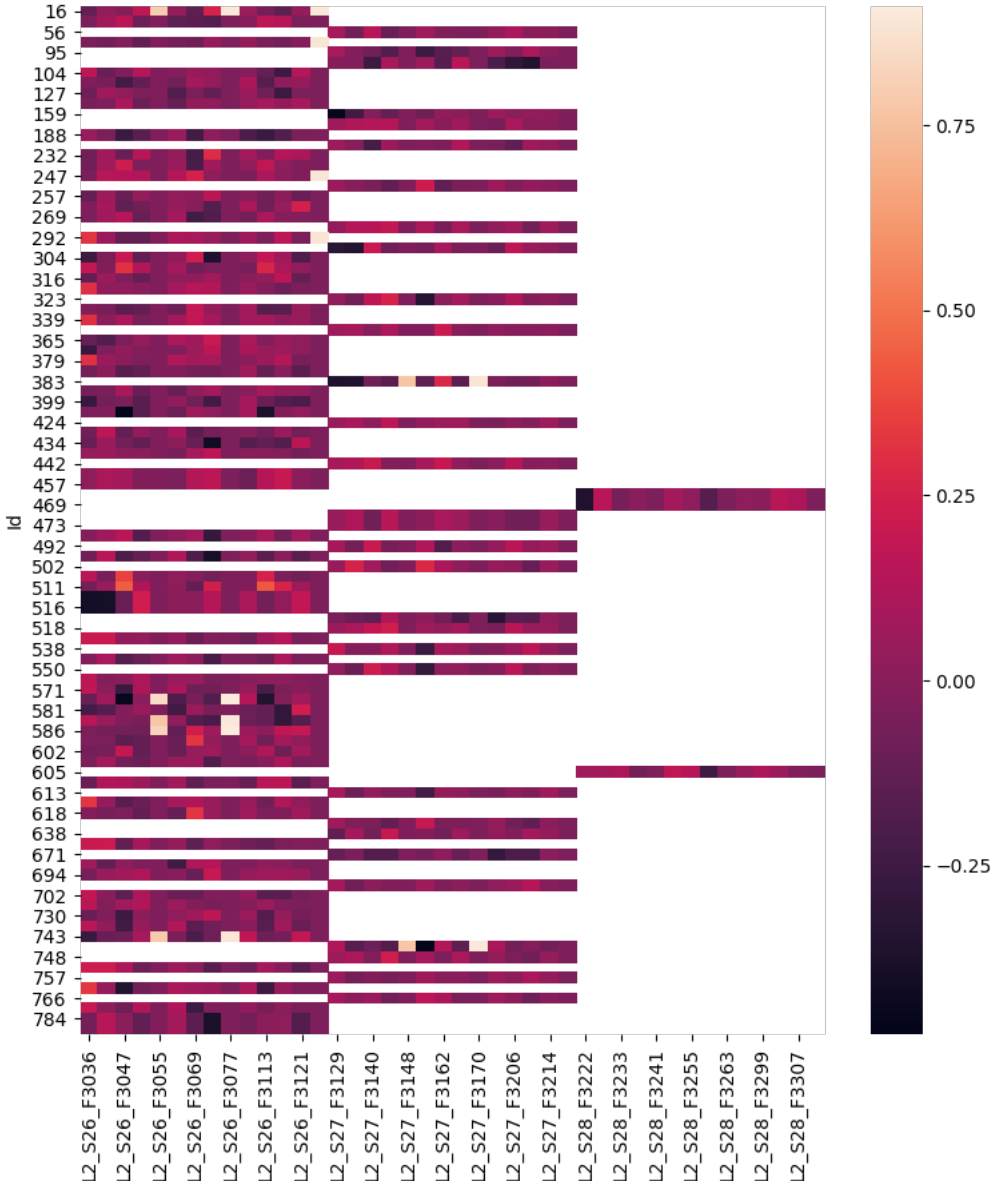


Figure 2.39: Heatmap of the first 100 rows of L2Numeric

a common behavior: if a certain part passes through the machinery represented by a certain Section number, all the features in that Section will assume numeric values, while if the part doesn't pass through that machinery, all the features will assume NaN value. With this mind, we can study the relation between Sections and Response.

This dataset contains three different sections, where the first one contains only a little percentage of NaN values, while the last one is almost completely filled of NaN values.

A Section affects the value of Response only if the part passes through the machinery it represents, i.e. only if its features assume not NaN values. In order to find a correlation between the error and the activation of a particular section, features have been grouped by Section, then can be defined an index, that will be called *error rate* of the Section S_i:

$$err(S_i) = \frac{N_1(S_i)}{N(S_i)} \tag{2.7}$$

Where $N_1(S_i)$ indicates the number of rows where the Section S_i has not NaN values and Response=1, $N(S_i)$ indicates all the rows where the Section S_i has not NaN values.

The average error rate $\overline{err}$ of the dataset coincides with the percentage of Response=1 described above. If $err(S_i)$ is bigger than the average, the probability of Response=1 is higher when the Section S_i is active (i.e. assumes not NaN values), vice versa the probability is lower if the $err(S_i)$ is lower than average. In Figure 2.40 has been calculated the $err(S_i)$ for each Section in Line 2 and it has been compared with the average $\overline{err} = 0.00721$ (red line).

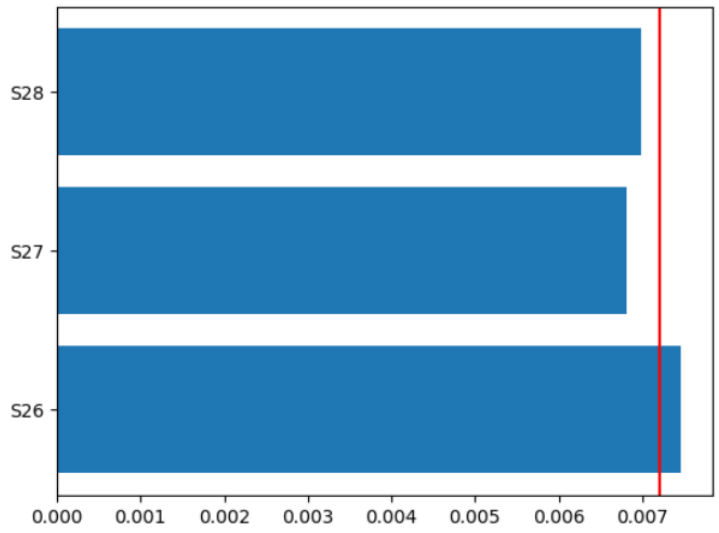


Figure 2.40: Error rate for each Section

The Section S26 contains only a few not null values, as seen in Figure 2.39, but it has the higher error rate. This fact can mean that Section S26 represent an old machinery, that processes part in a slow and inefficient way.

Line 2 Categorical

The dataset obtained processing PreProcCategorical and isolating information related to Line 2, will be called L2Categorical. L2Categorical dataset contains 159 columns. In Figure 2.41 is shown the percentage of not NaN values in this particular Line.

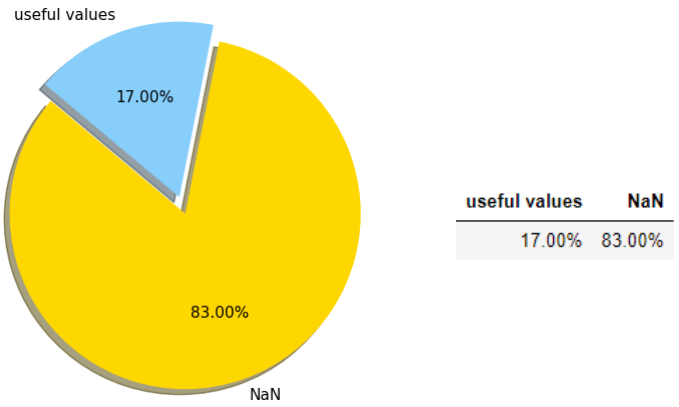


Figure 2.41: Percentage of not NaN values in L2Categorical

There percentage of useful values is particularly high respect to the others categorical datasets. Since this dataset contains categorical values, it is necessary to study how the categorical labels are distributed. Different features can assume the same label value. In Figure 2.56 is shown the frequency of appearance for each label value, excluding NaN.

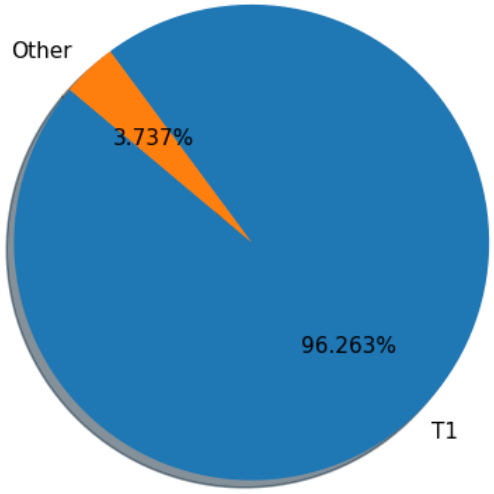


Figure 2.42: Labels with relative frequency

There are only three labels and only one appears in more than 96% of

the cases. All the labels with a low appearance frequency have been grouped in 'Other'. To calculate the *error rate* for each label, has been used the same formula used for features in Numeric dataset:

$$err(L_i) = \frac{N_1(L_i)}{N(L_i)} \tag{2.8}$$

Where $N_1(L_i)$ indicates the number of rows where the label L_i appears and Response=1, while $N(L_i)$ indicates all the rows where label L_i appears. The result shown in Figure 2.43 is very different from the one obtained studying L2Numeric.

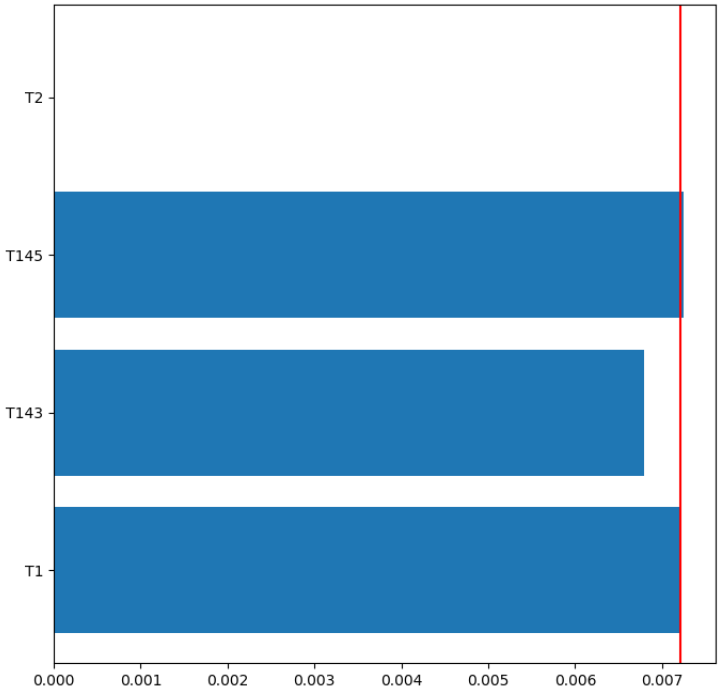


Figure 2.43: Error rate for each label

To explain the difference, we need to consider that the dataset contains only a little percentage of not NaN values related to label $T2$. Since also $Response = 1$ appears very rarely, we can suppose that the $error\_rate = 0$ of $T2$ is due to the fact that the simultaneous presence of two rare events is very improbable. With this presupposition, we can see that labels haven't

particular error rates.

In Figure 2.44 we can see the error rate of each Section, calculated using (2.10).

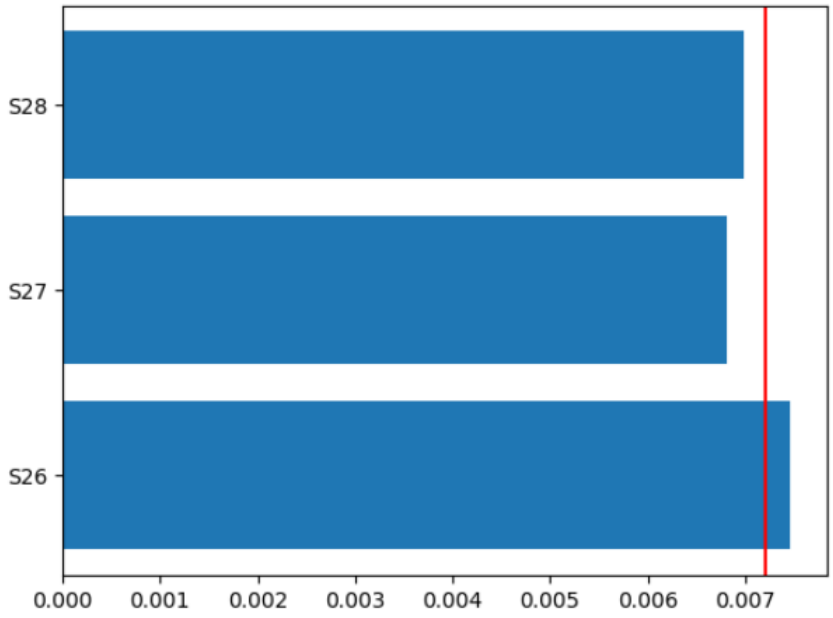


Figure 2.44: Error rate for each Section

These Sections have a very regular behavior, there are not particular results to analyze.

Line 2 Date

The dataset obtained processing PreProcDate and isolating information related to Line 2, will be called L2Date. L2Date dataset contains 78 columns. In Figure 2.45 we can see the percentage of NaN values in this particular Line.

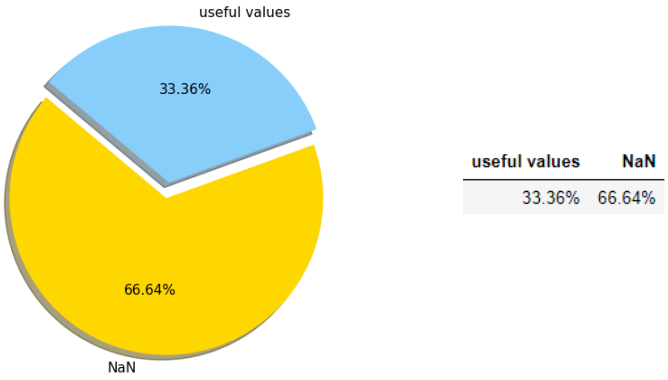


Figure 2.45: Percentage of NaN values in L0Date

The percentage is coherent with the other two datasets. An heatmap has been used to visualize data distribution. In Figure 2.46 is shown the plot of the first 100 rows.

As already noticed in L2Numeric, there is a clusterization by Section. We can also notice that each features in a Section has exactly the same value and it means that they have been recorded exactly in the same time. This fact supports the hypothesis that each Section contains the values of the sensors of a certain machinery. We can also see that values of different Sections in the same row defer a little, and this can mean that the part passes quite quickly from a machinery to the following. In Figure 2.47 is shown the trend of the average value calculated row by row,over the first 100 Ids.

There is a great variance, that can be due to the different elaboration time for different products.

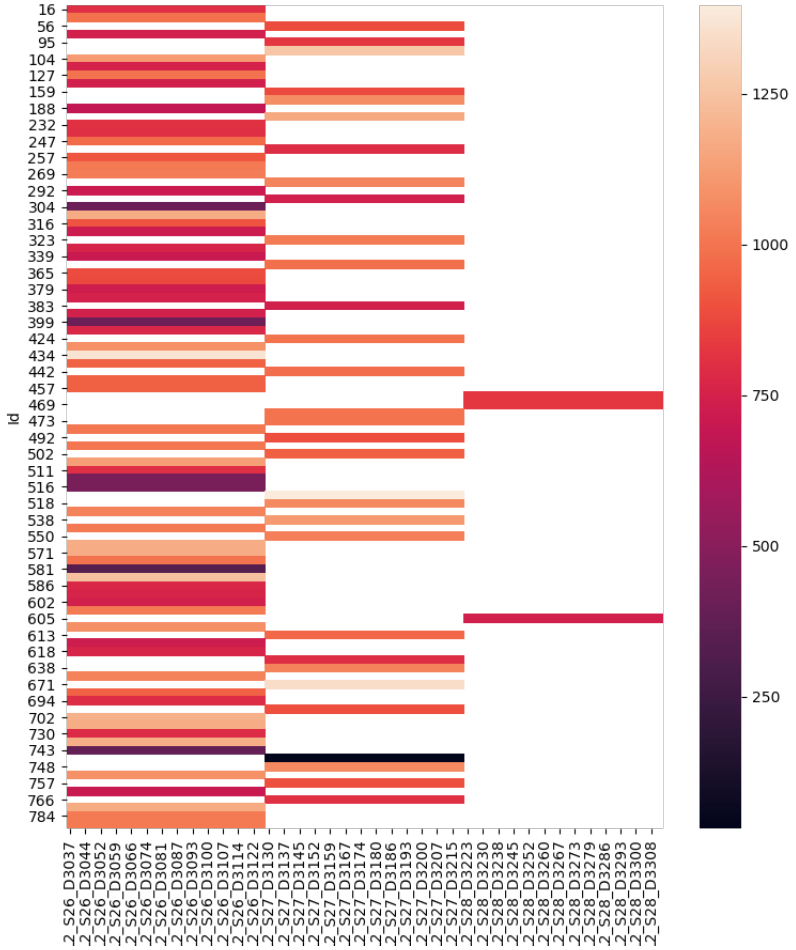


Figure 2.46: Heatmap of the first 100 rows of L2Date

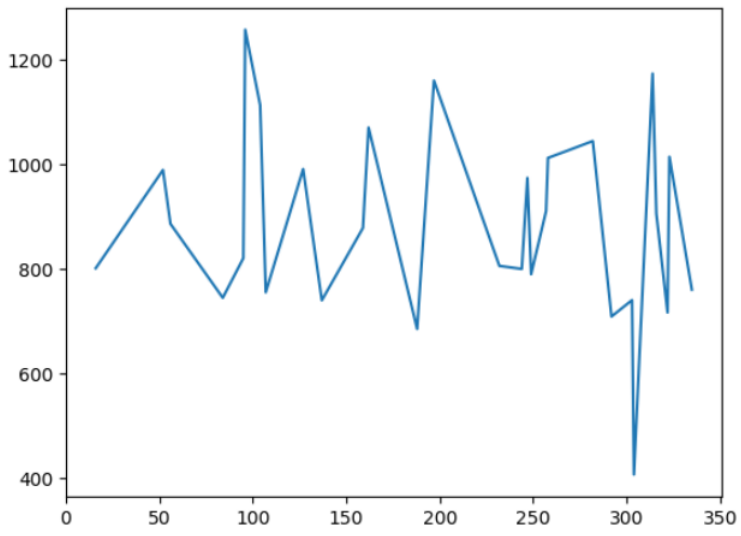


Figure 2.47: Plot of the average values of the first 100 rows of L2Date

Heatmaps

L2Date dataset should give a numeric time value for each value recorded in L2Numeric and L2Categorical, but dimensions suggest that a lot of data are missing. L2Numeric contains 43 columns and L2Categorical contains 159 columns, hence L2Date should contain $43 + 159 = 202$ columns, while it only contains 78. In this section the three datasets will be compared qualitatively, using heatmaps and studying the presence/absence of data.

L2Numeric vs L2Date

The purpose of this heatmap is to to check if each value in L2Numeric has a correspondent timestamp in L2Date. Comparing L2Numeric and L2Date datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L2Numeric and L2Date cells contain not NaN values (Green)
- L2Date=NaN, L2Numeric filled: L2Numeric cell contains a numeric value, while the corresponding cell in L2Date is NaN (Blue)
- L2Date not exists, L2Numeric filled: L2Numeric cell contains a numeric value, while the corresponding cell in L2Date doesn't exist (Yellow)
- L2Date filled, L2Numeric=NaN: L2Numeric cell contains a NaN value, while the corresponding cell in L2Date contains a not NaN value (Red)
- L2Numeric=Nan, L2Date=NaN/not exists: Both cells contains NaN or L2Numeric contains NaN, while L2Date cell doesn't exists (No color)

In Figure 2.48 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, there is only the situation "both filled". This is a situation of perfect information, were every feature in L2Numeric has

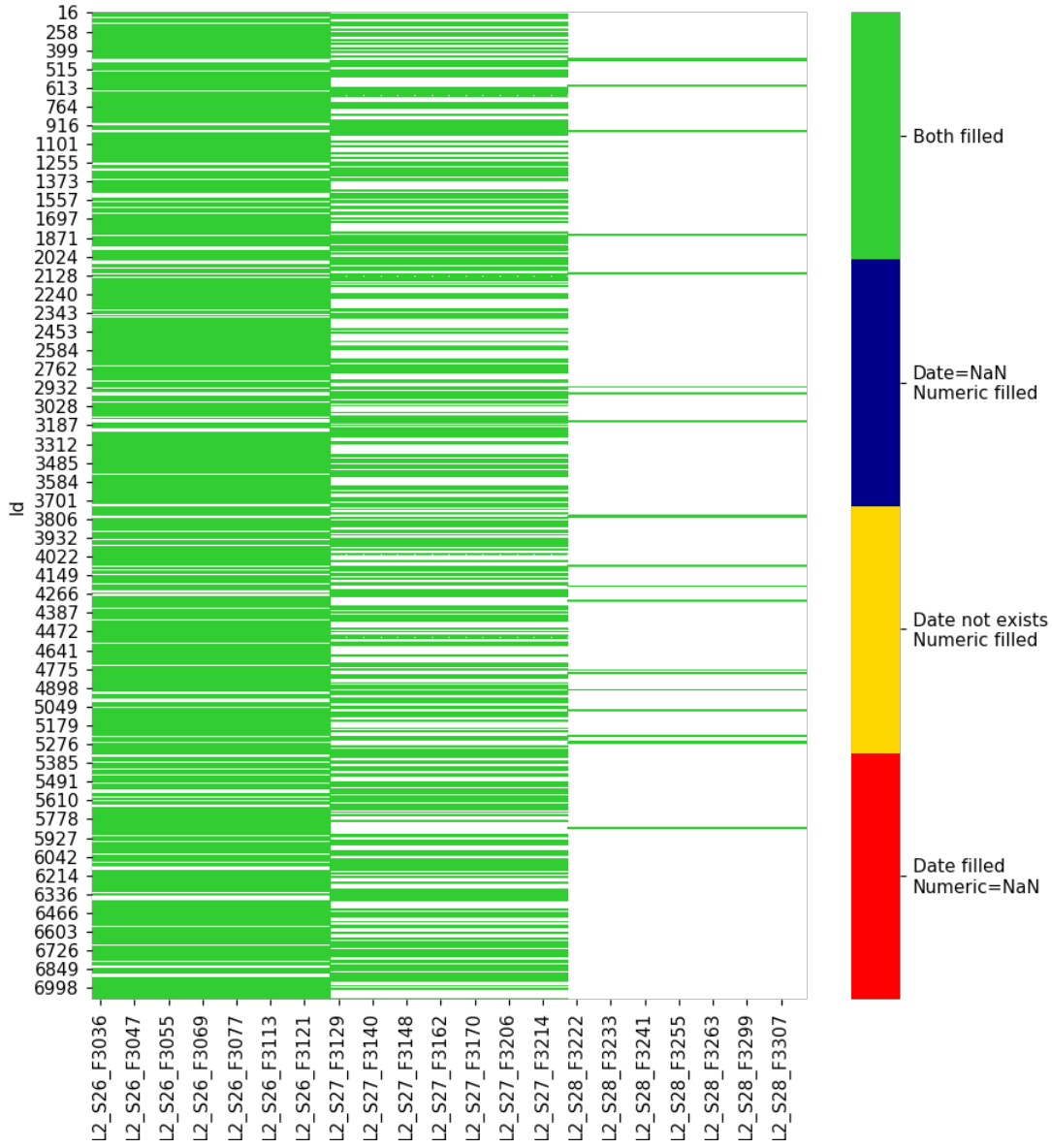


Figure 2.48: L2Numeric vs L2Date heatmap

a correspondent timestamp. Since L2Date has less columns than expected, this means that all missing columns are related to L"Categorical

L2Categorical vs L2Date

The purpose of this heatmap is to check if each value in L2Categorical has a correspondent timestamp in L2Date. As seen before, comparing L2Categorical to L2Date dataset, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L2Categorical and L2Date cells contain not NaN values (Green)
- L2Date=NaN, L2Categorical filled: L2Categorical cell contains a not NaN value, while the corresponding cell in L2Date is NaN (Blue)
- L2Date not exists, L2Categorical filled: L2Categorical cell contains a not NaN value, while the corresponding cell in L2Date doesn't exist (Yellow)
- L2Date filled, L2Categorical=NaN: L2Categorical cell contains a NaN value, while the corresponding cell in L2Date contains a not NaN value (Red)
- L2Categorical=Nan, L2Date=NaN/not exists: Both cells contains NaN or L2Categorical contains NaN, while L2Date cell doesn't exists (No color)

In Figure 2.49 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, the situation "both filled" appears very rarely, while the the case "Date filled and Categorical=NaN" appears in the most of the cases. This situation can be explained only supposing that categorical sensor can give NaN as output. This means that in Categorical dataset we can find two different "kind of NaN":

- Case 1: NaN means "no data have been collected". In this case, the corresponding value in Date is also "NaN

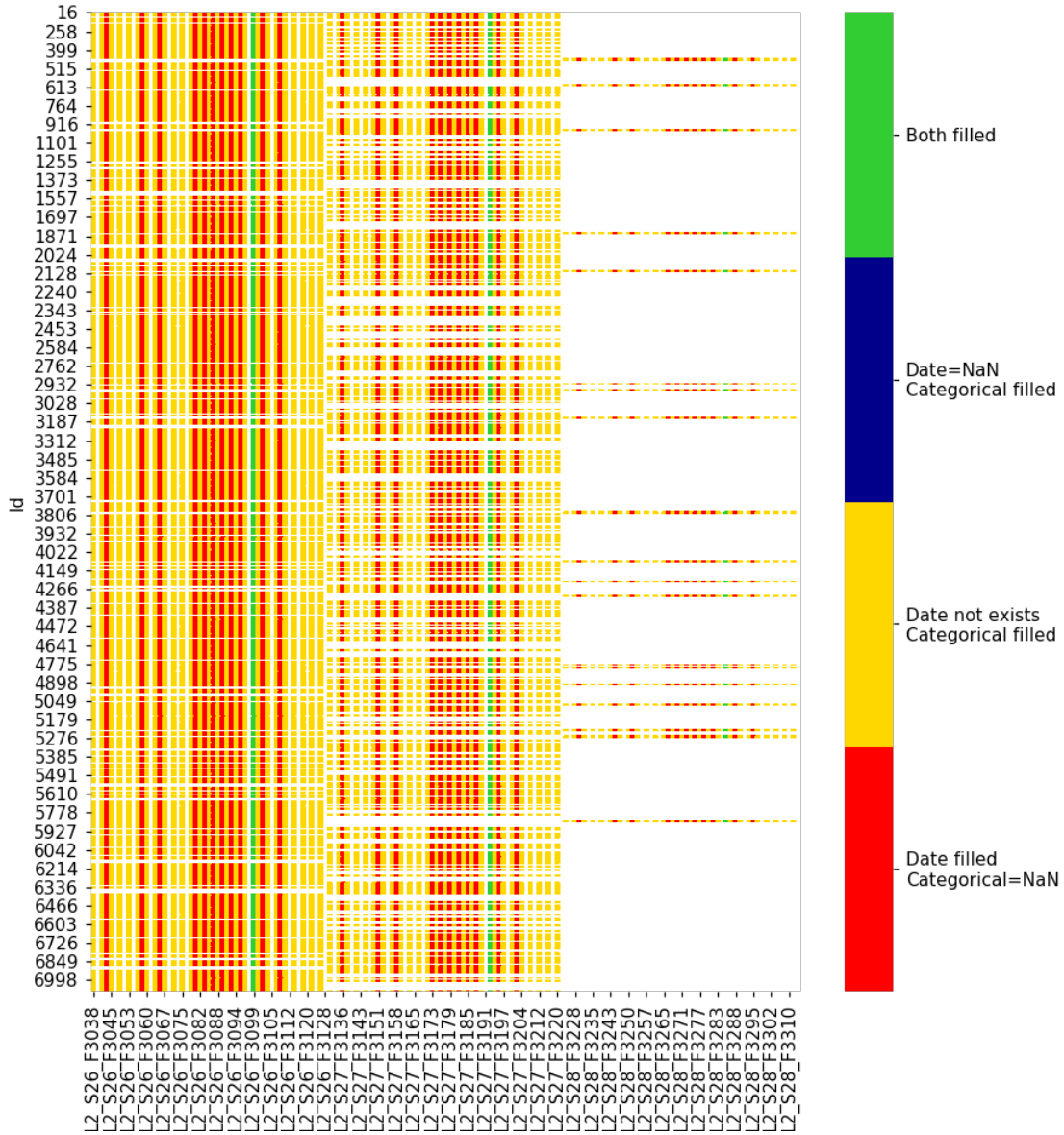


Figure 2.49: L2Categorical vs L2Date heatmap

- Case 2: NaN means "data with NaN value has been collected". In this case, the corresponding value in Date is not NaN

In a situation of perfect information, we could easily distinguish the two different kind of NaN by checking whether the corresponding value in Date is NaN or not, but in this case, a part of the feature haven't a corresponding column in Date, making a perfect discrimination impossible.

L2Date vs L2Numeric & L2Categorical

The purpose of this heatmap is to check if each value in L2Date is the timestamp of a correspondent value in L2Numeric or L2Categorical. We need remember that the three datasets only have *Id* as common column, so each column of L2Date can refer only to one column in L2Numeric or L2Categorical, but not to both of them. As seen before, comparing L2Date to L2Numeric and L2Categorical datasets, we can find five possible situations, that are represented with different colors:

- Both filled: L2Date and one of L2Numeric or L2Categorical cell contain not NaN values (while the other doesn't exist) (Green)
- L2Numeric or L2Categorical=NaN, L2Date filled: L2Date cell contains a not NaN value, while one corresponding cell in L2Numeric or L2Categorical is NaN (while the other doesn't exist) (Blue)
- Both L2Numeric and L2Categorical do not exist, L2Date filled: L2Date cell contains a not NaN value, while the corresponding cell in L2Numerical and L2Categorical doesn't exist (Yellow)
- L2Numeric or L2Categorical filled, L2Date=NaN: L2Date cell contains a NaN value, while the corresponding cell in L2Numeric or L2Categorical contain a not NaN value (Red)
- L2Date=Nan, L2Numeric and L2Categorical=NaN/not exist: L2Date cell contains NaN, while L2Numeric and L2Categorical contain NaN or don't exist (No color)

In Figure 2.50 is shown the result obtained plotting the first 1000 rows.

Green columns are due to the result obtained in "L2Numeric vs L2Date", while the blue ones are explained in "L2Categorical vs L2Date". We can see that there are not red cells: it means that every not NaN value in L2Numeric and L2Categorical has a correspondent not NaN timestamp in L2Date, if the related column exists.

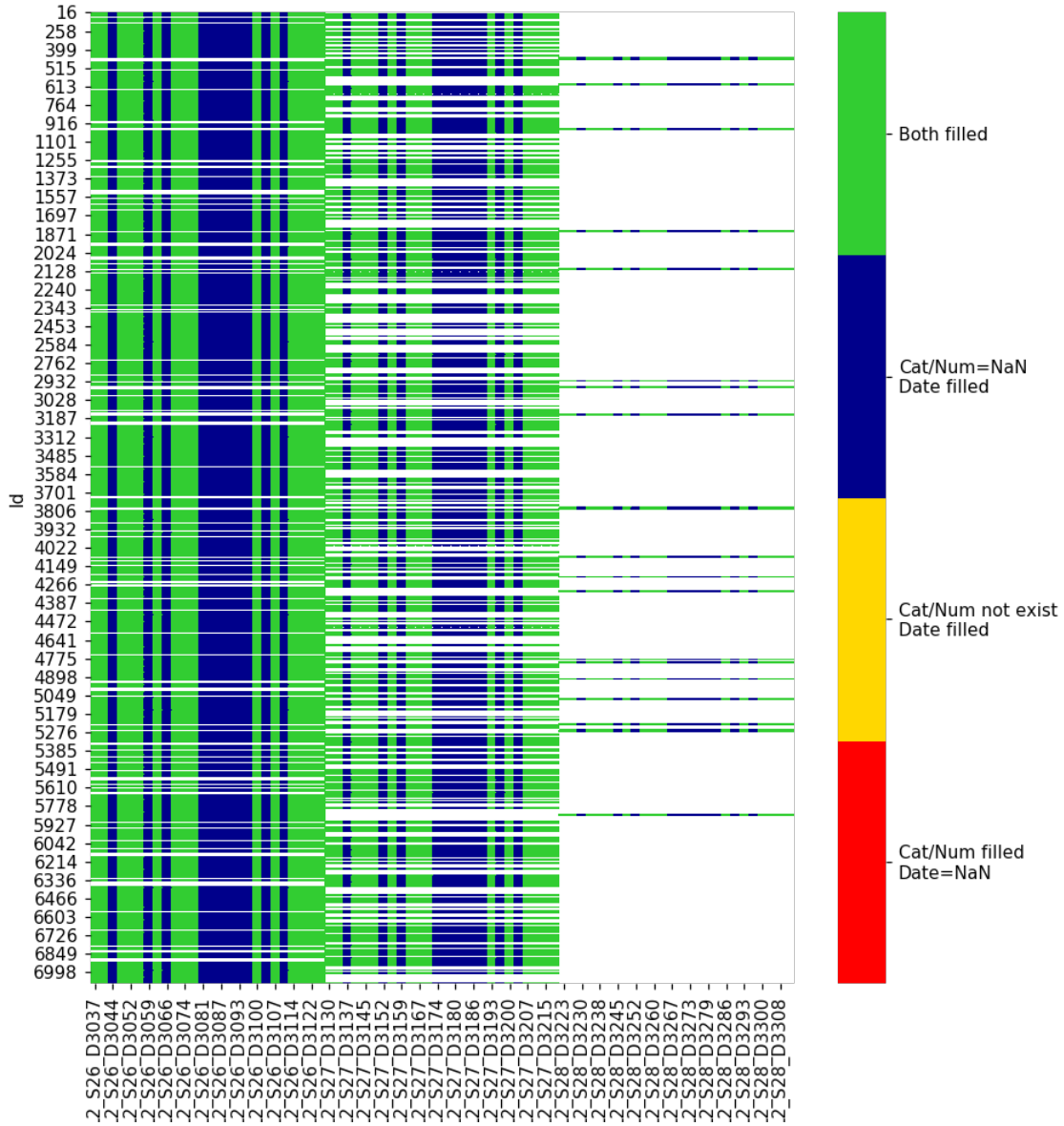


Figure 2.50: L2Date vs L2Numeric & L2Categorical heatmap

Correlation test

The purpose of this analysis is to find any kind of correlation between different Sections. Since almost each Section has features in both L2Numeric and L2Categorical, each dataset will be studied separately, then results will be compared.

Given a and b two Sections, the index called $confidence$ is defined as

follows:

$$conf(a,b) = P(a|b) = \frac{\#\{a,b\}}{\#\{a\}} \tag{2.9}$$

It defines the ratio between the times Sections a and b are active in the same row and the times Section a is active. In particular, $conf(a,b)$ measures how much $b \Rightarrow a$ namely, how much the presence of a implies the presence of b. This index has values in [0,1] and is used for finding association rules between a and b: if $conf(a,b) \sim 1$, then the presence of a means that b has high probability to appear, while $conf(a,b) \sim 0$ means that the presence of a means that b has low probability to appear. We need to observe that the index isn't symmetric, in fact $conf(a,b) \neq conf(b,a)$.

The characteristic of confidence index is that it is influenced by relative frequencies of the events a and b. For example, if $\#\{a\} = 1000$, $\#\{b\} = 10$ and $\#\{a,b\} = 10$, we can see a strong relation between a and b, because b happens if and only if happens a, then $conf(b,a) = 1$, while $conf(a,b) = 0.01$, so we can understand that $b \Rightarrow a$, but $a \not\Rightarrow b$.

Since Line 2 contains only three Sections, studying correlation for each dataset is unnecessary, because the result obtained is always the same.

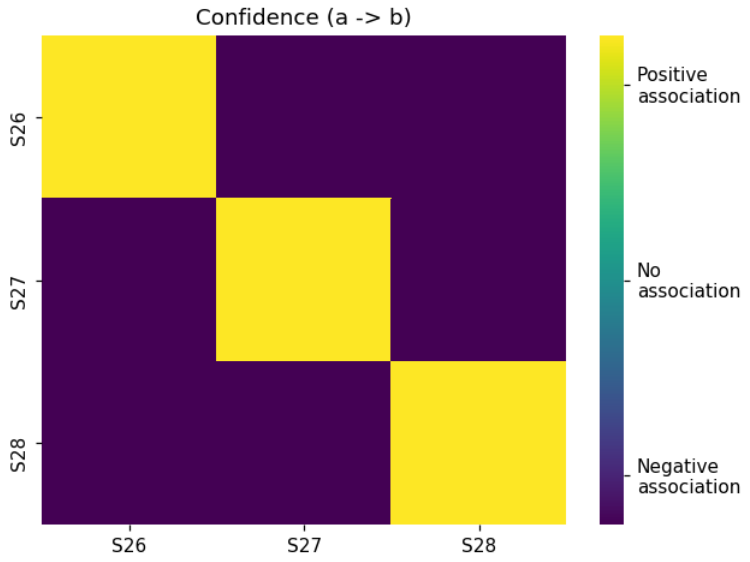


Figure 2.51: Confidence between Sections in L2Numeric

In this preliminary analysis will be studied when each Section assumes

not NaN values, not studying the particular values assumed, but only considering the cases active/not active.

The confidence index (2.12) between each Section is plotted using an heatmap in Figure 2.51. The three Sections show a strong negative correlation. We can suppose that these Sections could represent machinery that work in parallel or machinery with mutually exclusive utility (i.e. paintings of different colors). There are only three sections, thus studying correlation of L2Categorical and L2Numeric dataset would give exactly the same results.

2.3.4 Line 3

After the cleaning process, each dataset contains 1183158 rows related to *Line*3, indexed by the common primary key *Id*. Each file will be initially studied separately, then will be studied how the three dataset are each other related.

Line 3 Numeric

The dataset obtained processing PreProcNumeric and isolating information related to Line 3, will be called L3Numeric. L3Numeric contains 246 columns. In Figure 2.52 is shown the result obtained calculating the percentage of not NaN values and Response=1 rows in this particular Line.

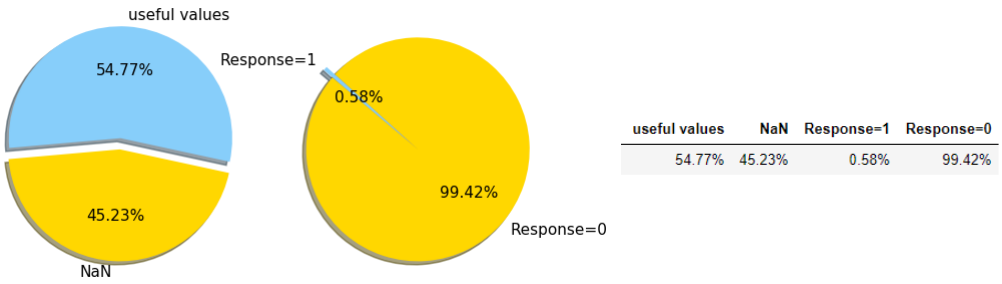


Figure 2.52: Percentage of not NaN values and Response=1 rows in L3Numeric

This Line contains the highest percentage of not NaN values. To visualize the distribution of NaN values over the dataset, has been used an heatmap: a different color is assigned for each numeric value, while the NaN values are left without color. In Figure 2.53 is shown the result obtained plotting the first 100 rows.

Not null data are more densely distributed, compared to the other lines and NaN values are not random distributed. In fact, features look to be gathered in clusters within which, they have a similar behavior: or all they are NaN, or all they assume a numerical value. Looking at the identification codes, we can understand that features with the same behavior belong to the same Section. We can suppose a Section to be a particular machinery, containing various sensors: each sensor is represented by a specific feature

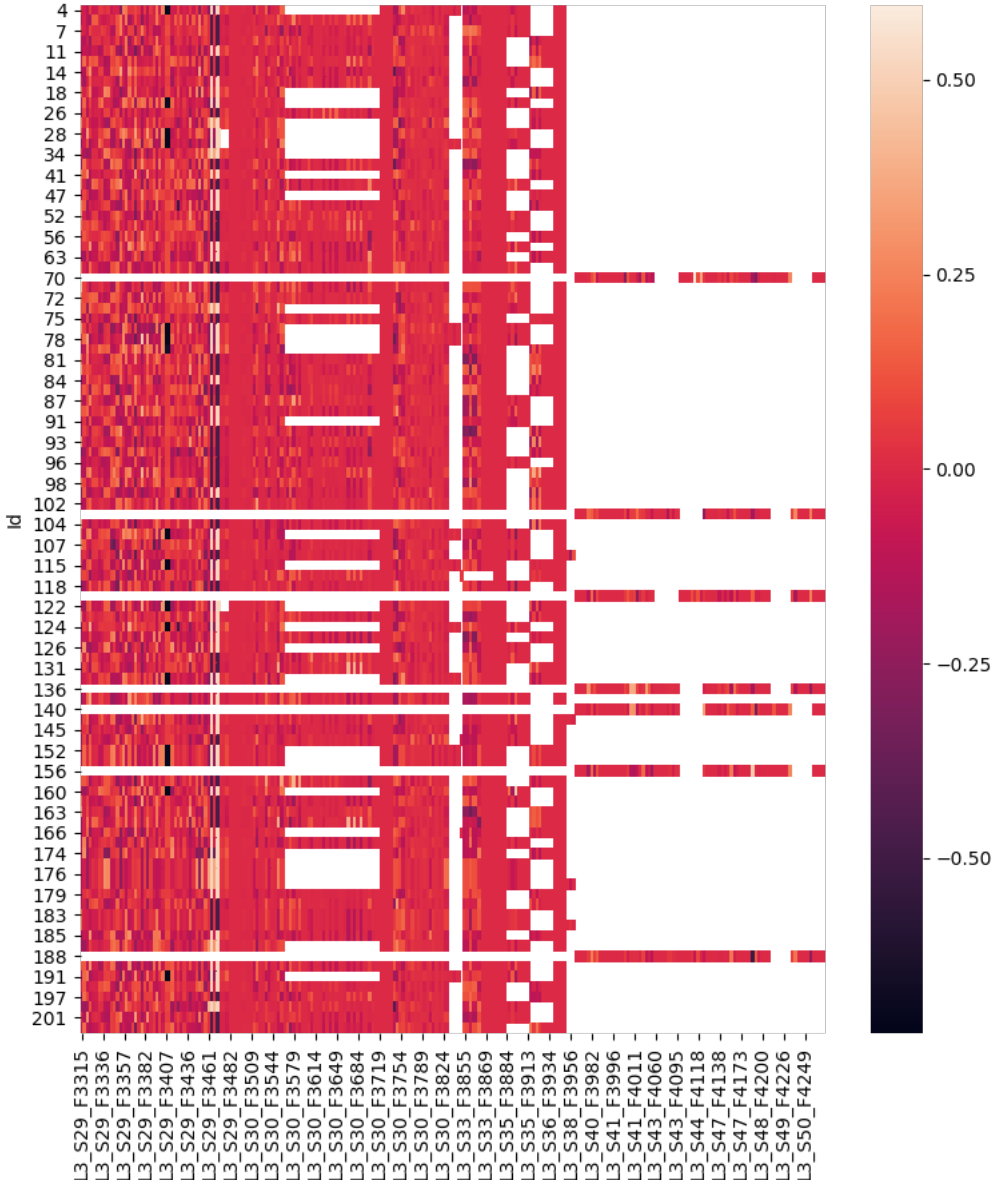


Figure 2.53: Heatmap of the first 100 rows of L3Numeric

in the dataset. This theory explains why features in the same section have a common behavior: if a certain part passes through the machinery represented by a certain Section number, all the features in that Section will assume numeric values, while if the part doesn't pass through that machinery, all the features will assume NaN value. With this mind, we can study the relation between Sections and Response.

A Section affects the value of Response only if the part passes through the machinery it represents, i.e. only if its features assume not NaN values. In order to find a correlation between the error and the activation of a particular section, features have been grouped by Section, then can be defined an index, that will be called *error rate* of the Section S_i:

$$err(S_i) = \frac{N_1(S_i)}{N(S_i)} \tag{2.10}$$

Where $N_1(S_i)$ indicates the number of rows where the Section S_i has not NaN values and Response=1, $N(S_i)$ indicates all the rows where the Section S_i has not NaN values.

The average error rate $\overline{err}$ of the dataset coincides with the percentage of Response=1 described above. If $err(S_i)$ is bigger than the average, the probability of Response=1 is higher when the Section S_i is active (i.e. assumes not NaN values), vice versa the probability is lower if the $err(S_i)$ is lower than average. In Figure 2.54 has been calculated the $err(S_i)$ for each Section in Line 3 and it has been compared with the average $\overline{err} = 0.00581$ (red line).

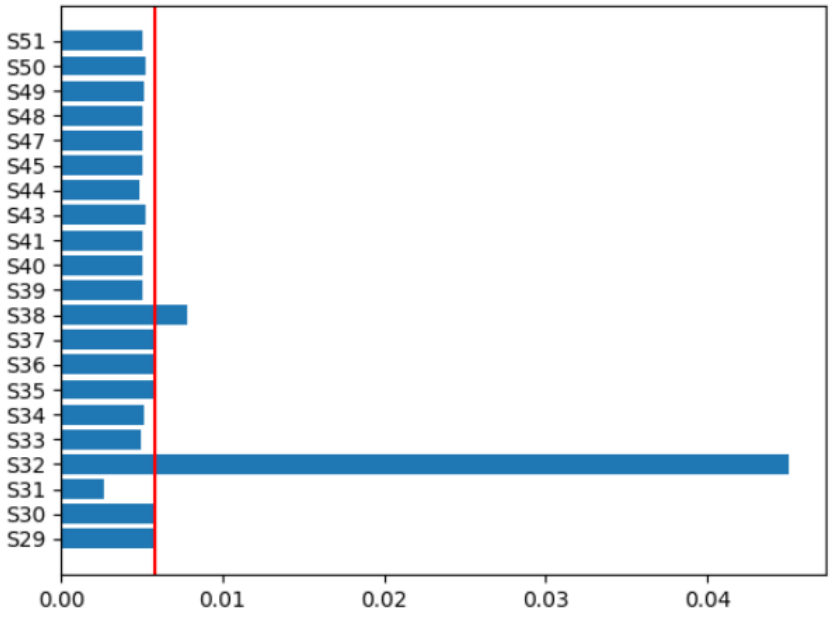


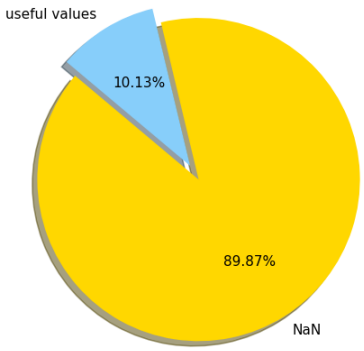
Figure 2.54: Error rate for each Section

It is easy to notice that Section 32 has a really high error rate. This great

error can be explained assuming that this Section describes a machinery dedicated to rework damaged parts or to perform a particularly delicate operation.

Line 3 Categorical

The dataset obtained processing PreProcCategorical and isolating information related to Line 3, will be called L3Categorical. L3Categorical dataset contains 418 columns. In Figure 2.55 is shown the percentage of not NaN values in this particular Line.



useful values	NaN
10.13%	89.87%

Figure 2.55: Percentage of not NaN values in L3Categorical

The percentage of useful values is not so low, compared to the other Lines. Since this dataset contains categorical values, it is necessary to study how the categorical labels are distributed. Different features can assume the same label value. In Figure 2.56 is shown the frequency of appearance for each label value, excluding NaN.

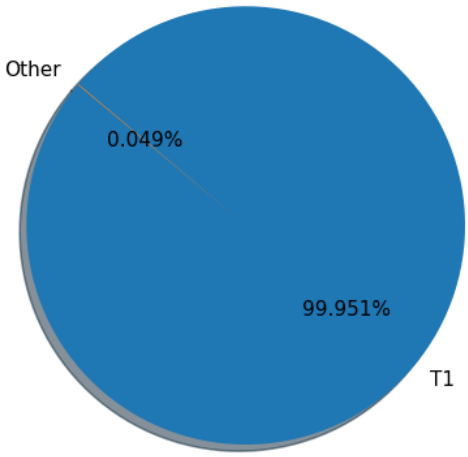


Figure 2.56: Labels with relative frequency

There are 35 different labels, but only one appears in the most of the cases, while the others appear very rarely. To calculate the *error rate* for

each label, has been used the same formula used for features in Numeric dataset:

$$err(L_i) = \frac{N_1(L_i)}{N(L_i)} \tag{2.11}$$

Where $N_1(L_i)$ indicates the number of rows where the label L_i appears and Response=1, while $N(L_i)$ indicates all the rows where label L_i appears. The result shown in Figure 2.57 is very different from the one obtained studying L3Numeric.

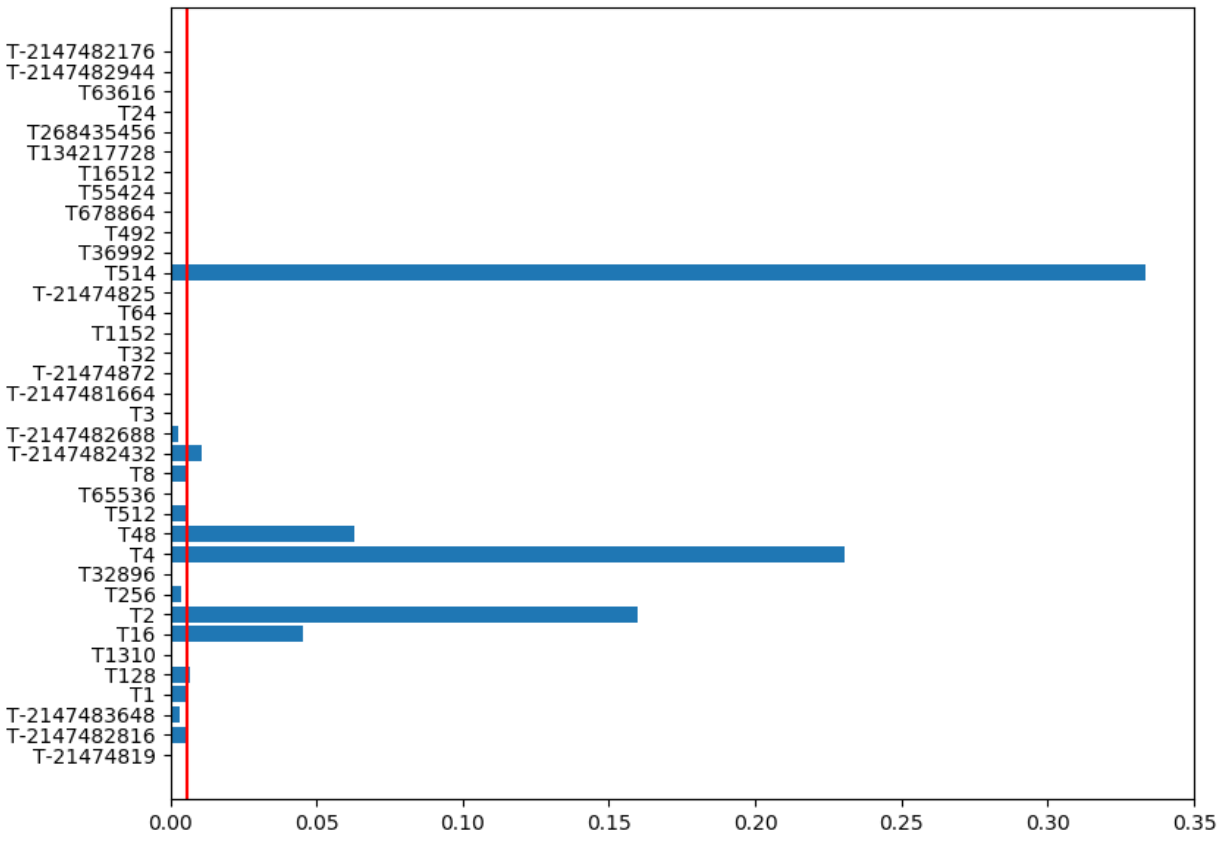


Figure 2.57: Error rate for each label

To explain the difference, we need to consider that the dataset contains only a little percentage of not NaN values. Since also $Response = 1$ appears very rarely, we can suppose that the $error\_rate = 0$ of the majority of the

labels is due to the fact that the simultaneous presence of two rare events is very improbable. With this presupposition, we can see that some label have interesting error rates, compared to the average of 0.58%:

- $T514$: $err(T514) = 33\%$, it is the highest error rate ever found, but $N(T514) = 9$ (i.e. it compares only 9 times in all the dataset). It's not very useful.
- $T48$: $err(T48) = 6\%$, ten times the average and $N(T48) = 556$. It may be useful.
- $T4$: $err(T4) = 23\%$ and $N(T4) = 775$. It has an error rate a bit lower than $T514$, but it appears more frequently.
- $T2$: $err(T2) = 16\%$ and $N(T2) = 4831$. It appears quite frequently, it is very useful.
- $T16$: $err(T16) = 5\%$, about an half of the average and $N(T16) = 3800$. It appears less frequently than $T2$ and it has a smaller error rate, but it may be useful.

In Figure 2.58 we can see the error rate of each Section, calculated using (2.10).

Since each Section has a few rows with not NaN values, we need to compare the results with the number of useful rows in the dataset, indicated in Figure 2.59 by the column 'Samples'.

The most interesting Sections are:

- $S49$: $err(S49) = 16\%$, but it appears very rarely, $N(S49) = 43$
- $S38$: $err(S38) = 33\%$, very high, but $N(S38) = 3$
- $S35$: $err(S35) = 16\%$, but $N(S2) = 61$
- $S32$: $N(S32) = 21588$, is the section that appears more often and $err(S32) = 5\%$, ten time the average, it probably is the most useful

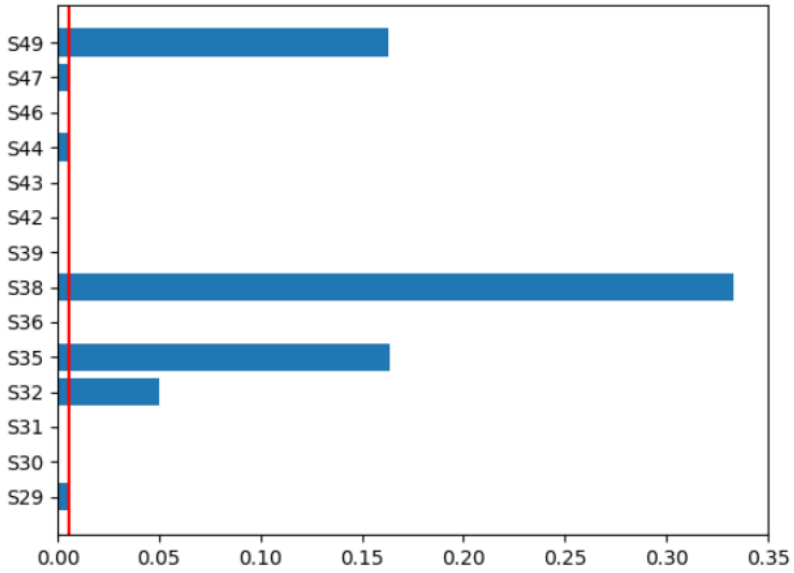


Figure 2.58: Error rate for each Section

	Features	Samples	Error_Rate
S29	63	785437	0.005820
S30	204	16	0.000000
S31	8	7	0.000000
S32	3	21588	0.050352
S35	18	61	0.163934
S36	8	1	0.000000
S38	6	3	0.333333
S39	8	3	0.000000
S42	24	15	0.000000
S43	24	283	0.000000
S44	8	2093	0.006689
S46	1	1	0.000000
S47	11	59955	0.005070
S49	21	43	0.162791

Figure 2.59: Error rate and useful rows for each Section

Line 3 Date

The dataset obtained processing PreProcDate and isolating information related to Line 3, will be called L3Date. L3Date dataset contains 273 columns. In Figure 2.60 is shown the percentage of NaN values in this particular Line.

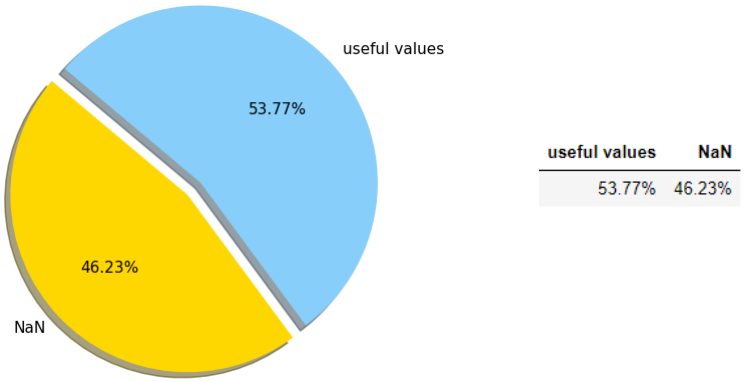


Figure 2.60: Percentage of NaN values in L3Date

The percentage is coherent with the other two datasets. An heatmap has been used to visualize data distribution. In Figure 2.61 is shown the plot of the first 100 rows.

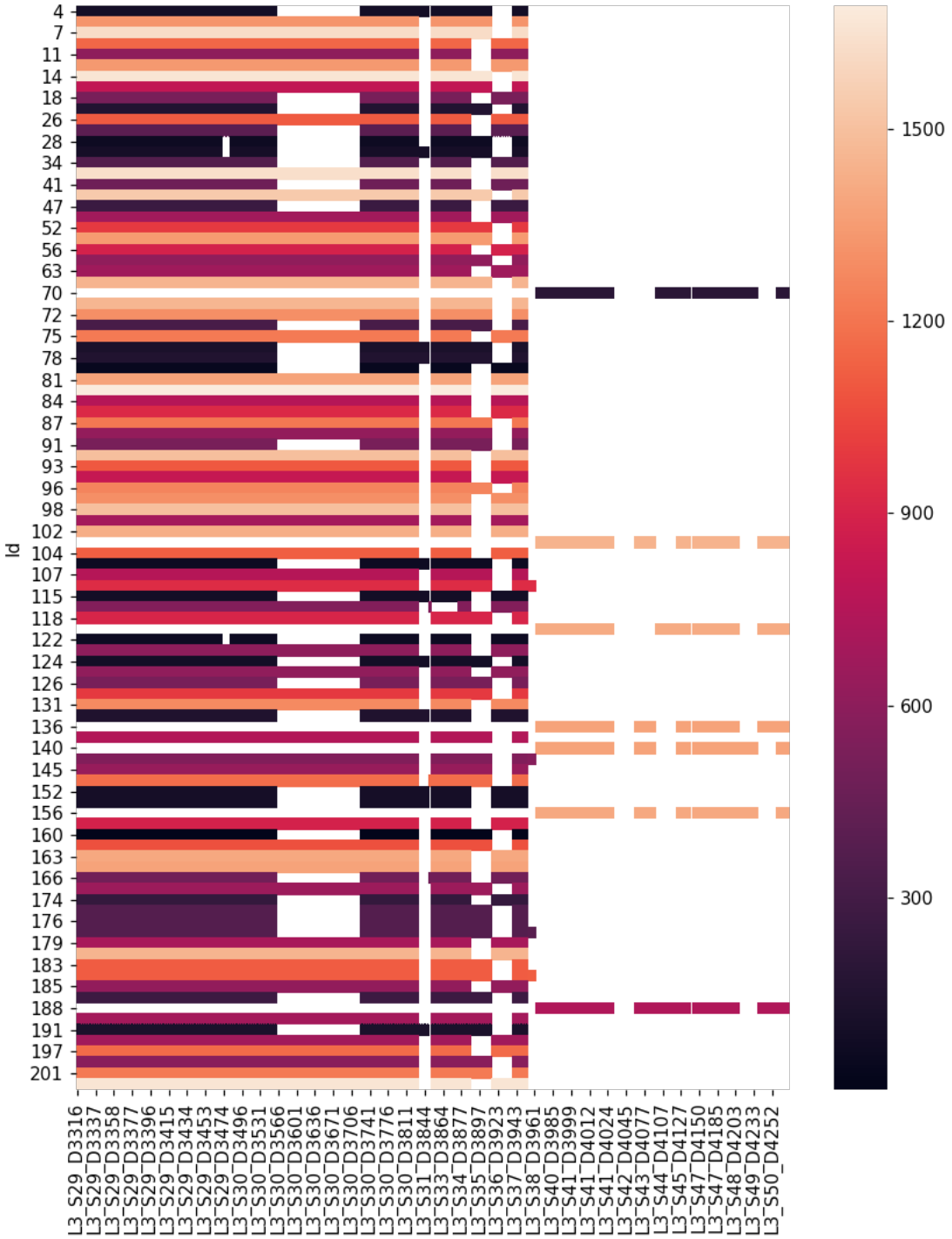


Figure 2.61: Heatmap of the first 100 rows of L3Date

As already noticed in L3Numeric, there is a clusterization by Section. It is also possible to notice that each features in a Section has exactly the same value and it means that they have been recorded exactly in the same time. This fact supports the hypothesis that each Section contains the values of the sensors of a certain machinery. We can also see that values of different Sections in the same row defer a little, and this can mean that the part passes quite quickly from a machinery to the following. In Figure 2.62 is shown the trend of the average value calculated row by row,over the first 100 Ids.

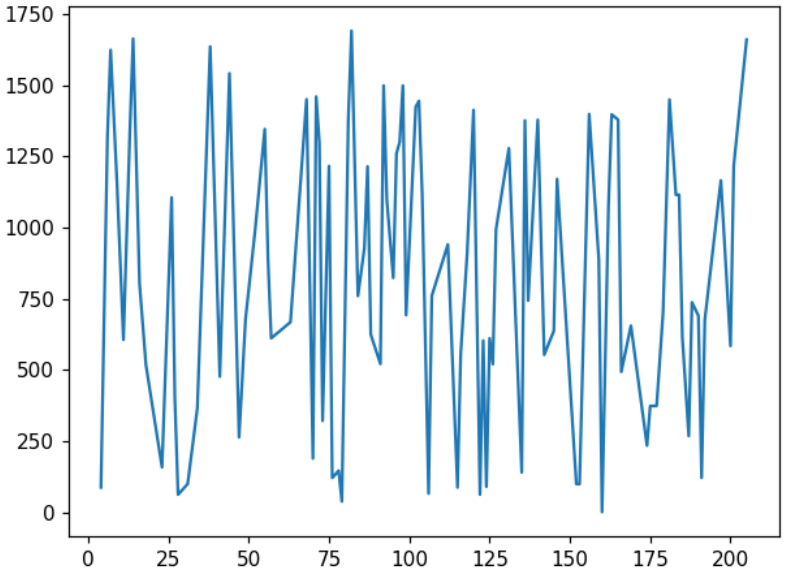


Figure 2.62: Plot of the average values of the first 100 rows of L3Date

There is a great variance, that can be due to the different elaboration time for different products.

Heatmaps

L3Date dataset should give a numeric time value for each value recorded in L3Numeric and L3Categorical, but dimensions suggest that a lot of data are missing. L3Numeric contains 246 columns and L3Categorical contains 418 columns, hence L3Date should contain $246 + 418 = 664$ columns, while it only contains 273. In this section the three datasets will be compared qualitatively, using heatmaps and studying the presence/absence of data.

L3Numeric vs L3Date

The purpose of this heatmap is to to check if each value in L3Numeric has a correspondent timestamp in L3Date. Comparing L3Numeric and L3Date datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L3Numeric and L3Date cells contain not NaN values (Green)
- L3Date=NaN, L3Numeric filled: L3Numeric cell contains a numeric value, while the corresponding cell in L3Date is NaN (Blue)
- L3Date not exists, L3Numeric filled: L3Numeric cell contains a numeric value, while the corresponding cell in L3Date doesn't exist (Yellow)
- L3Date filled, L3Numeric=NaN: L3Numeric cell contains a NaN value, while the corresponding cell in L3Date contains a not NaN value (Red)
- L3Numeric=Nan, L3Date=NaN/not exists: Both cells contains NaN or L3Numeric contains NaN, while L3Date cell doesn't exists (No color)

In Figure 2.63 is shown the result obtained plotting the first 1000 rows.

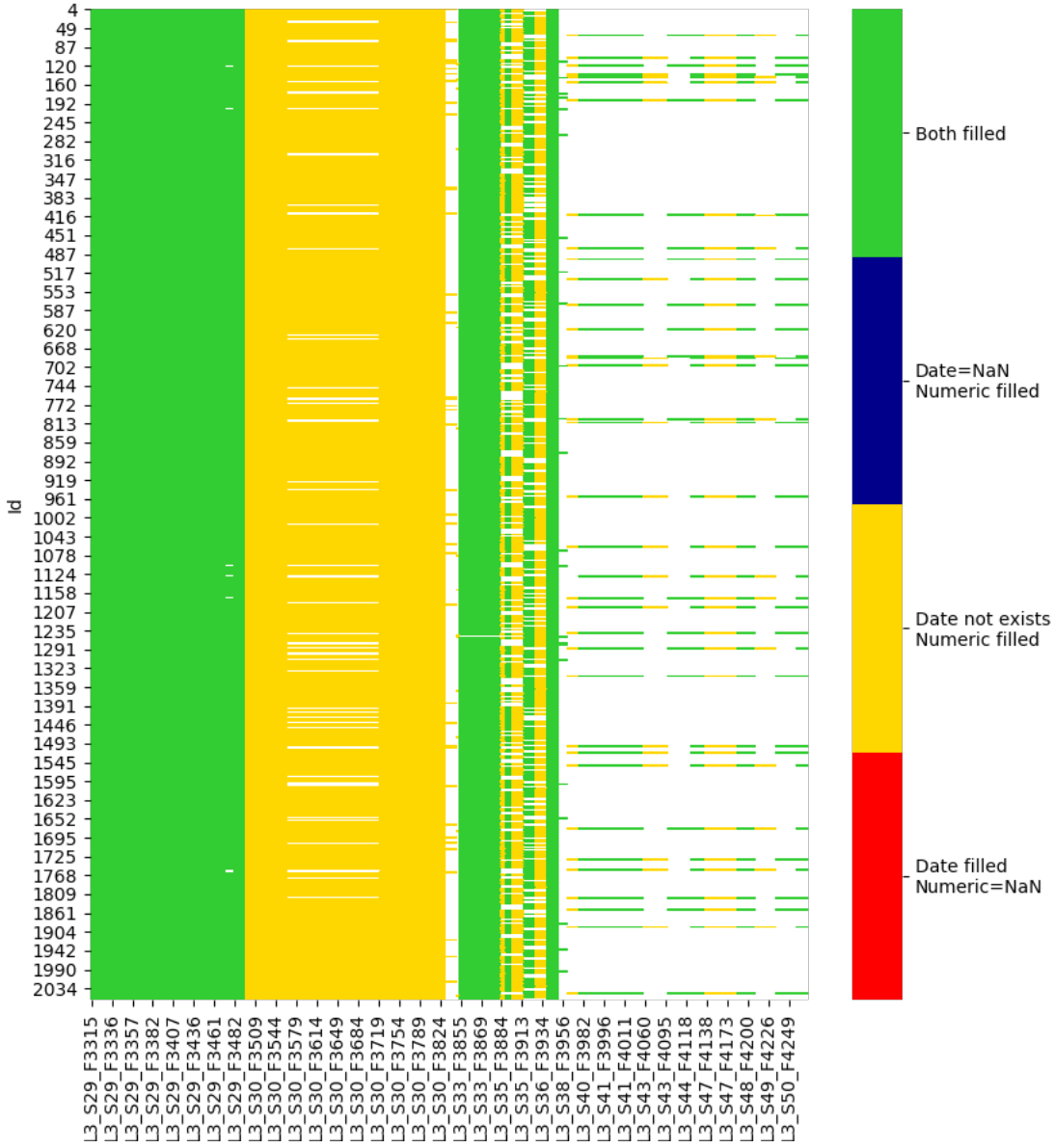


Figure 2.63: L3Numeric vs L3Date heatmap

Beside the both NaN case, there are only two situations: "both filled" and "Numeric filled and Date doesn't exist". In a situation of perfect information, we should only find the "both filled" case, but as we can see, not every feature in L3Numeric has a correspondent timestamp. This may be due to the fact that recording a timestamp for each feature is wasteful, since a lot of features are recorded in the same time or simply because a

part of the instrumentation is old and it hasn't the possibility to record a time value.

L3Categorical vs L3Date

The purpose of this heatmap is to check if each value in L3Categorical has a correspondent timestamp in L3Date. As seen before, comparing L3Categorical to L3Date dataset, is possible to find five possible situations, that are represented with different colors:

- Both filled: both L3Categorical and L3Date cells contain not NaN values (Green)
- L3Date=NaN, L3Categorical filled: L3Categorical cell contains a not NaN value, while the corresponding cell in L3Date is NaN (Blue)
- L3Date not exists, L3Categorical filled: L3Categorical cell contains a not NaN value, while the corresponding cell in L3Date doesn't exist (Yellow)
- L3Date filled, L3Categorical=NaN: L3Categorical cell contains a NaN value, while the corresponding cell in L3Date contains a not NaN value (Red)
- L3Categorical=Nan, L3Date=NaN/not exists: Both cells contains NaN or L3Categorical contains NaN, while L3Date cell doesn't exists (No color)

In Figure 2.64 is shown the result obtained plotting the first 1000 rows.

Beside the both NaN case, the situations "both filled" and "Categorical filled and Date doesn't exist" appear very rarely, while the the case "Date filled and Categorical=NaN" appears in the most of the cases. This situation can be explained only supposing that categorical sensor can give NaN as output. This means that in Categorical dataset we can find two different "kind of NaN":

- Case 1: NaN means "no data have been collected". In this case, the corresponding value in L3Date is also "NaN

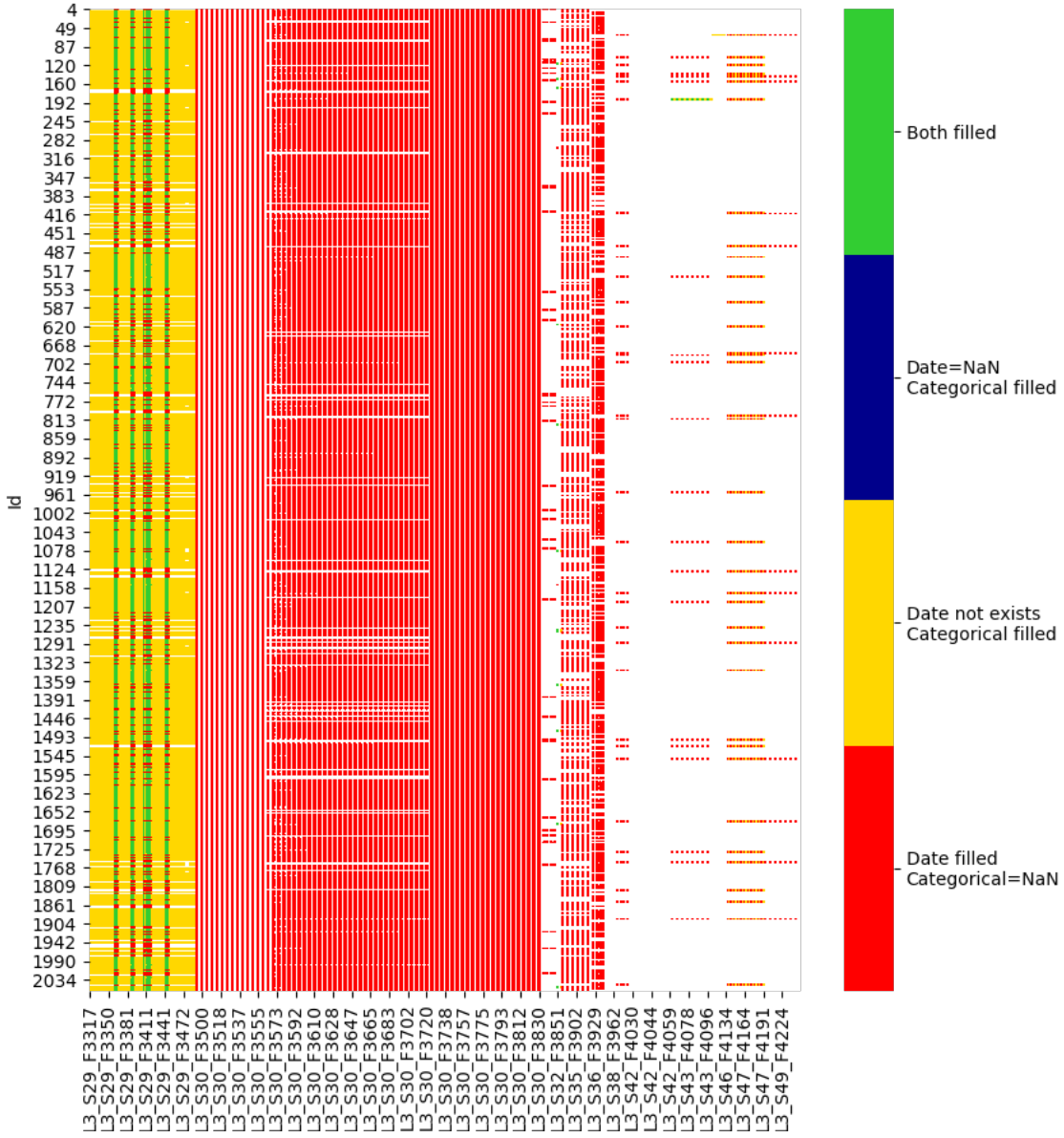


Figure 2.64: L3Categorical vs L3Date heatmap

- Case 2: NaN means "data with NaN value has been collected". In this case, the corresponding value in L3Date is not NaN

In a situation of perfect information, we could easily distinguish the two different kind of NaN by checking whether the corresponding value in Date is NaN or not, but in this case, a part of the feature haven't a corresponding column in Date, making a perfect discrimination impossible.

L3Date vs L3Numeric & L3Categorical

The purpose of this heatmap is to check if each value in L3Date is the timestamp of a correspondent value in L3Numeric or L3Categorical. We need remember that the three datasets only have *Id* as common column, so each column of L3Date can refer only to one column in L3Numeric or L3Categorical, but not to both of them. As seen before, comparing L3Date to L3Numeric and L3Categorical datasets, is possible to find five possible situations, that are represented with different colors:

- Both filled: L3Date and one of L3Numeric or L3Categorical cell contain not NaN values (while the other doesn't exist) (Green)
- L3Numeric or L3Categorical=NaN, L3Date filled: L3Date cell contains a not NaN value, while one corresponding cell in L3Numeric or L3Categorical is NaN (while the other doesn't exist) (Blue)
- Both L3Numeric and L3Categorical do not exist, L3Date filled: L3Date cell contains a not NaN value, while the corresponding cell in L3Numerical and L3Categorical doesn't exist (Yellow)
- L3Numeric or L3Categorical filled, L3Date=NaN: L3Date cell contains a NaN value, while the corresponding cell in L3Numeric or L3Categorical contain a not NaN value (Red)
- L3Date=Nan, L3Numeric and L3Categorical=NaN/not exist: L3Date cell contains NaN, while L3Numeric and L3Categorical contain NaN or don't exist (No color)

In Figure 2.65 is shown the result obtained plotting the first 1000 rows.

The result obtained is almost what could be expected looking at the heatmaps before. The green columns are mainly due to the result obtained in "L3Numeric vs L3Date", while the blue ones are explained in "L3Categorical vs L3Date". We can see that there are not red cells: it means that every not NaN value in L3Numeric and L3Categorical has a

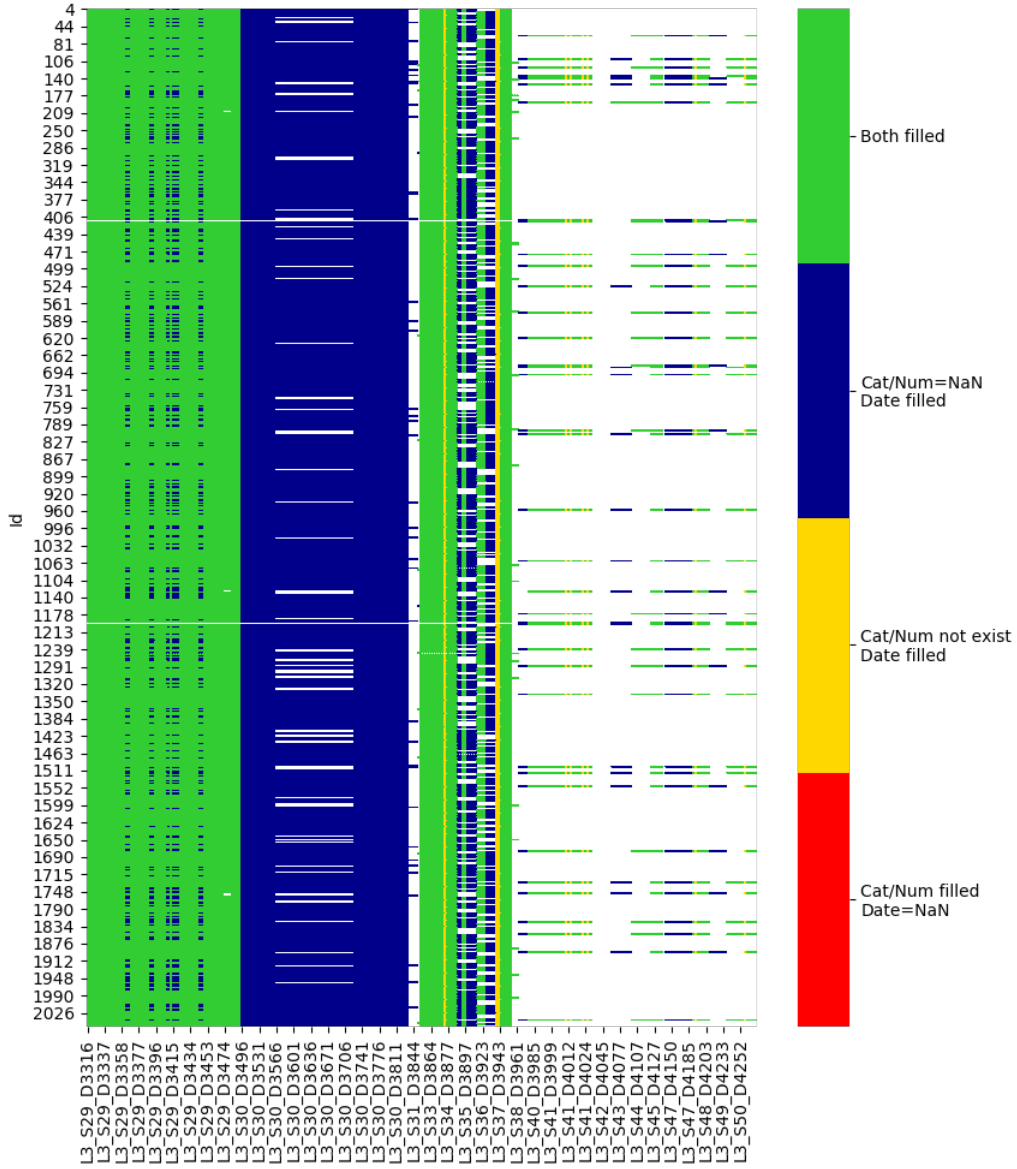


Figure 2.65: L3Date vs L3Numeric & L3Categorical heatmap

correspondent not NaN timestamp in L3Date, if the related column exists. The only exception we can see are some yellow cells that represent columns in L3Date without any correspondence in L3Numeric or L3Categorical. At the moment can't be find any good explanation for this phenomenon except that data inconsistency.

Correlation test

The purpose of this analysis is to find any kind of correlation between different Sections. Since almost each Section has features in both L3Numeric and L3Categorical, each dataset will be studied separately, then results will be compared.

Given a and b two Sections, the index called $confidence$ is defined as follows:

$$conf(a,b) = P(a|b) = \frac{\#\{a,b\}}{\#\{a\}} \tag{2.12}$$

It defines the ratio between the times Sections a and b are active in the same row and the times Section a is active. In particular, $conf(a,b)$ measures how much $b \Rightarrow a$ namely, how much the presence of a implies the presence of b. This index has values in [0,1] and is used for finding association rules between a and b: if $conf(a,b) \sim 1$, then the presence of a means that b has high probability to appear, while $conf(a,b) \sim 0$ means that the presence of a means that b has low probability to appear. We need to observe that the index isn't symmetric, in fact $conf(a,b) \neq conf(b,a)$.

The characteristic of confidence index is that it is influenced by relative frequencies of the events a and b. For example, if $\#\{a\} = 1000$, $\#\{b\} = 10$ and $\#\{a,b\} = 10$, we can see a strong relation between a and b, because b happens if and only if happens a, then $conf(b,a) = 1$, while $conf(a,b) = 0.01$, so we can understand that $b \Rightarrow a$, but $a \not\Rightarrow b$.

L3Numeric

In this preliminary analysis will be studied when each Section assumes not NaN values, not studying the particular values assumed, but only considering the cases active/not active. The confidence index (2.12) between each Section is plotted using an heatmap in Figure 2.66.

This Line appears separated in two block with a strongly negative relation, probably dedicated to processing different kind of parts. Inside each

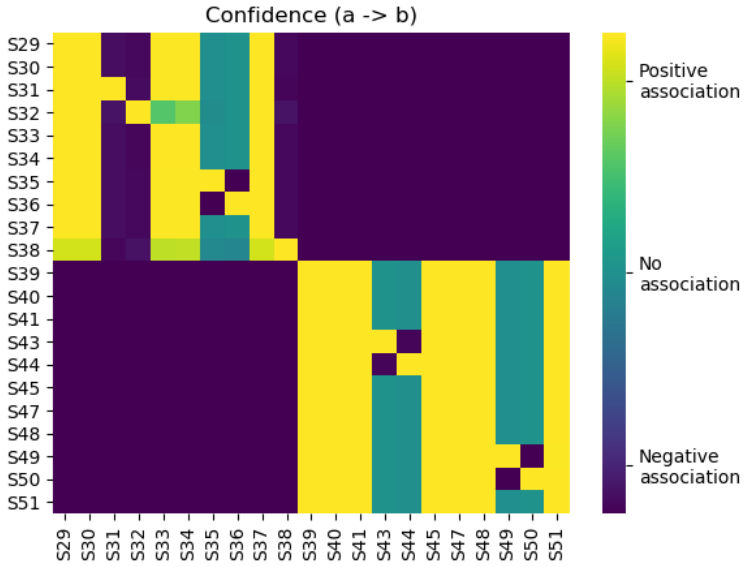


Figure 2.66: Confidence between Sections in L3Numeric

block, some Sections are negatively related, that could represent machinery that work in parallel or machinery with mutually exclusive utility.

L3Categorical

In Figure 2.67, relation between Sections in L3Categorical is studied using (2.12).

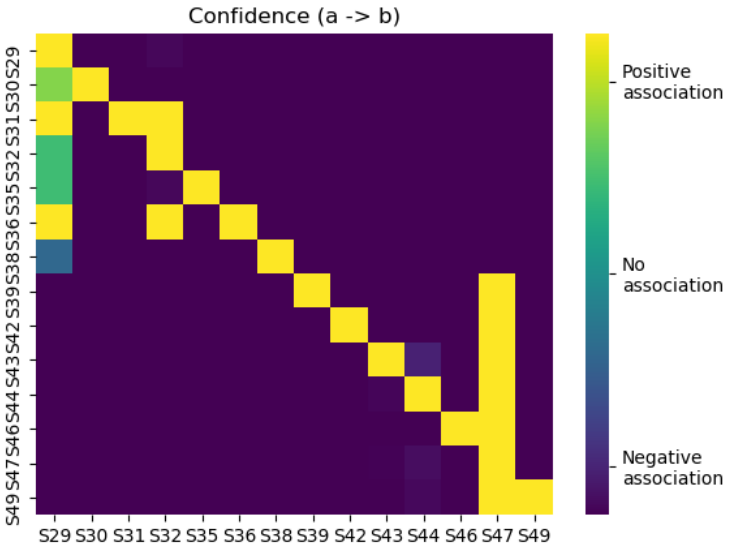


Figure 2.67: Confidence between Sections in L3Categorical

The result is less significant than the one obtained with L3Numeric, since there is a greater number of NaN values.

L3Date

The relation between Sections in L3Date is studied in Figure 2.68 using (2.12).

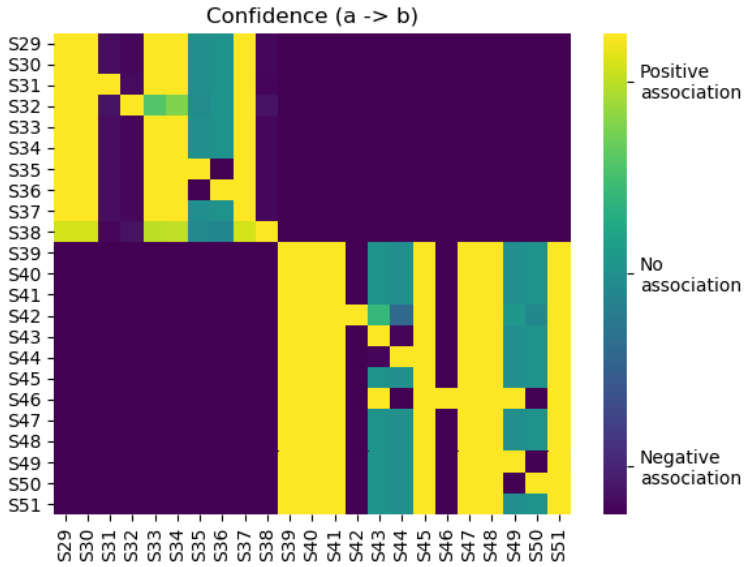


Figure 2.68: Confidence between Sections in L3Date

As expected, the result obtained confirms what already noticed studying L3Numeric. In conclusion, Line 3 contains the higher error rates, this means that it will be one of the most useful Lines in creating a well performing model.

Chapter 3

Objectives

In order to perform a Root Cause Analysis is necessary to create a predictive model capable of explaining data behavior with a good approximation and then to extrapolate the most significant features. According to the exploratory analysis, data are not 'ready to use' but they are very heterogeneous, large dimensional and with a lot of missing values. In the following chapter are described methods used to solve all the problems encountered.

Chapter 4

Methods

To make a good prediction, it is necessary to consider that different types of part pass throught the production lines. Parts of the same type or with similar features are processed with the same machineries, namely they have not null values in the same Sections in the dataset under exam. The discipline dedicated to part clustering is called *Group technology* or cellular manufacturing.

4.1 Group technology

Group technology started in 1920s with the purpose to improve manufacturing efficiency. The aim, as described in [1], was to classify parts, so that parts with similar features could be manufactured together with standardized processes.

More generally, GT can be defined as a theory of management, where similar products are clustered together. Nowadays, this technology is used by factories in two different ways: groping machines according to the parts manufactured by them (this application is called Cellular Manufacturing) or grouping parts according to the machines they pass through. Both approaches bring significant benefits such as:

- Decreased setup time

- WIP quantity reduction (WIP means Work In Progress, refers to parts partially processed)
- Material-manipulation cost reduction
- Decreased direct and indirect work cost
- Better quality
- Optimized material flow
- Improvement in machine utilization
- Optimized space utilization

For example, in Figure 4.1 is represented a typical factory organization, where machines are grouped on the basis of their functionality.

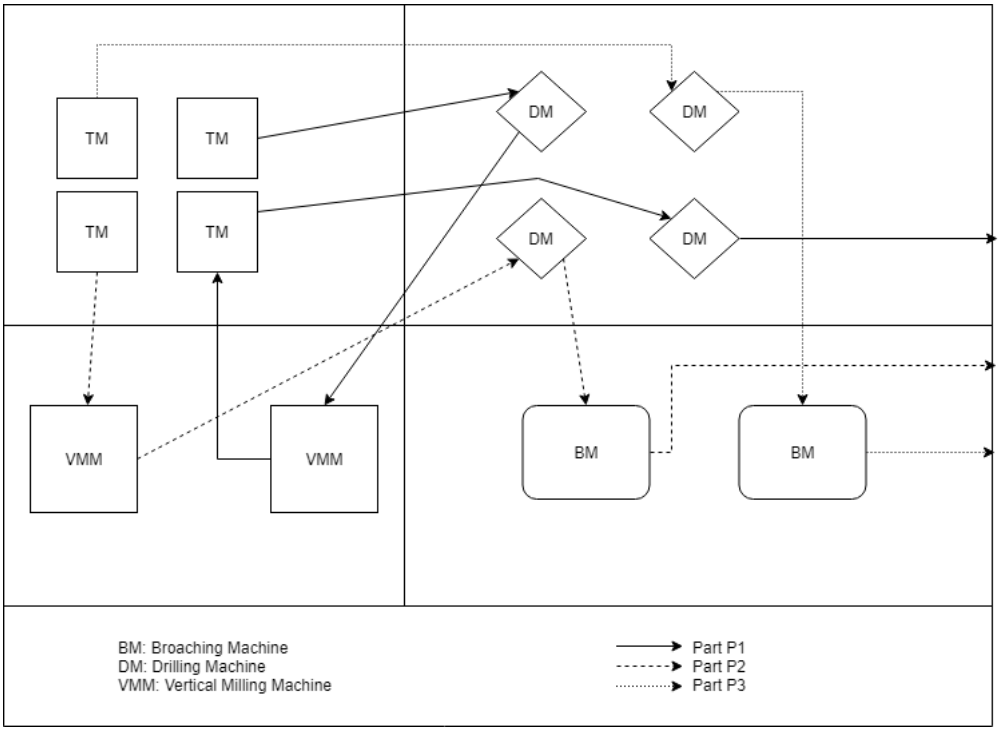


Figure 4.1: Typical organization

Part flow is very intricate and this involves a big loss of time and efficiency. Using the theory of group technology, machinery can be organized in cells, where each cell is dedicated to the processing of a particular family of parts, as shown in Figure 4.2.

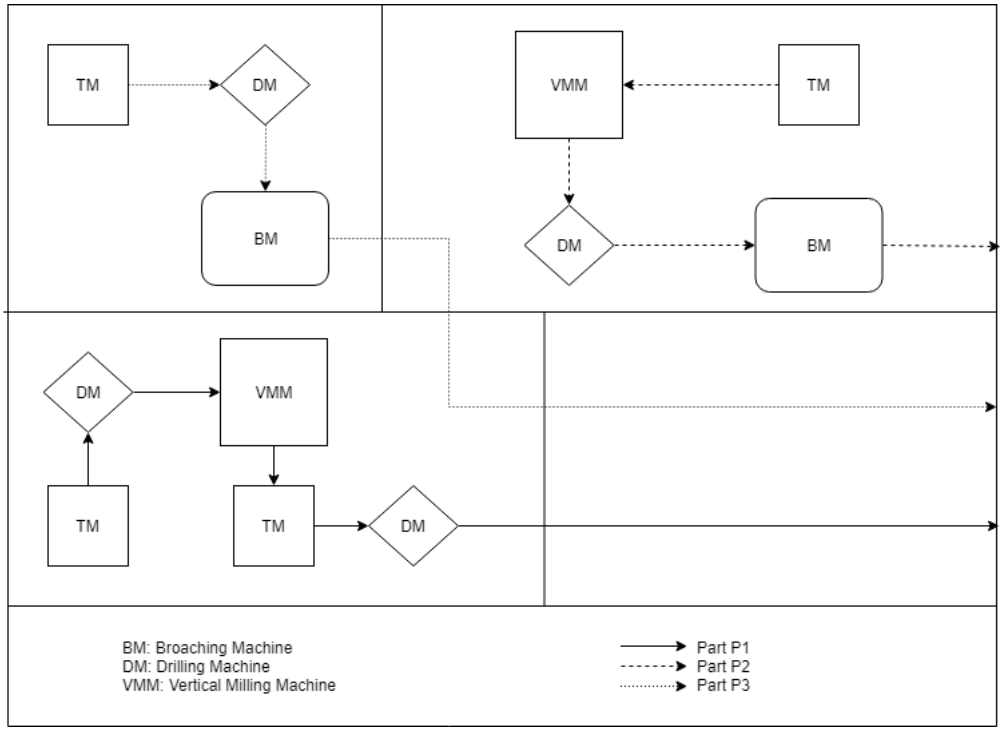


Figure 4.2: Cellular Manifacturing organization

The result is a reorganization where flows are shorter and more linear. In more complex situations, where there are more parts and machinery are not dedicated to a single family of parts, cells need to interact with each other, but the aim of Cellular Manufacturing is to minimize inter-cellular interactions.

In the context in exam, Group Technology can be used to divide all the different parts produced in small groups with similar machinery path. Since datasets are very large, it is necessary to find an algorithm able to perform a good clustering in a short time. Different algorithms will be tested on L3Numeric, one of the bigger and more interesting datasets in exam. All the techniques described have been found in [4]

Data preparation

Group technology algorithms doesn't need data generated by parts passing through machines, but they only need to know whether or not a certain part passes through a certain machine. As discovered in previous analysis, each Section contains the values of the sensors of a certain machinery, so

it is possible to reduce data granularity grouping by Sections. L3Numeric is thus modified as follows: it has a single column for each section, from which it takes its name, and its entries can assume 0 and 1 values only. If we find a 1 in row i and column j, it means that part i passed through the machinery j, while a 0 means that the part is not passed through that machine. The dataset obtained in this process is called $L0NumericS$, in Figure 4.3 is shown its header.

Id	S29	S30	S31	S32	S33	S34	S38	...	S40	S45	S47	S48	S49	S50	S51
4	1	1	1	0	1	1	0	...	0	0	0	0	0	0	0
6	1	1	0	0	1	1	0	...	0	0	0	0	0	0	0
7	1	1	0	0	1	1	0	...	0	0	0	0	0	0	0
9	1	1	0	0	1	1	0	...	0	0	0	0	0	0	0
11	1	1	0	0	1	1	0	...	0	0	0	0	0	0	0

Figure 4.3: Header of L3NumericS

This is the data format used in the algorithms described below.

4.1.1 Rank Order Clustering

ROC algorithm, first published in [6], is one of the most used in Group Technology problems. Considering L3NumericS as a rectangular matrix with 0,1 values, with n rows and m columns, the algorithm sorts rows and columns in order to create a block-diagonal matrix, where each block identifies a distinct cluster.

The algorithm assigns a value to each column making a scalar product with an array of powers of 2, sorted in ascending order (i.e. $[1\ 2\ 2^2\ 2^3\ 2^4\ \ldots]$), obtaining a score for each column. Columns are then sorted in ascending order according to their score. The same process is applied to the rows.

Algorithm stops when sorting steps don't change rows and columns order anymore or when the maximum number of iterations $maxIter$ is reached. Steps of the algorithm are shown below:

- *Step* 1: Create an array of powers of 2 with size m, called PR, where the $i-th$ entry assumes the value: $(PR)_i = 2^i$

- *Step* 2: Create an array of powers of 2 with size n, called PC, where the $i-th$ entry assumes the value: $(PC)_i = 2^i$

- *Step* 3: For each row j, whose entries are $\{a_{ji}\}_{i=1,\ldots,m}$, calculate the corresponding score RS_j making a scalar product with PR:

$$RS_j = \sum_{i=1}^{m} (PR)_i * a_{ji} = \sum_{i=1}^{m} 2^i * a_{ji} \tag{4.1}$$

- *Step* 4: Sort the rows in increasing order of their score RS_j

- *Step* 5: For each column j, whose entries are $\{a_{ij}\}_{i=1,\ldots,n}$, calculate the corresponding score CS_j making a scalar product with PC:

$$CS_j = \sum_{i=1}^{n} (PC)_i * a_{ij} = \sum_{i=1}^{n} 2^i * a_{ij} \tag{4.2}$$

- *Step* 6: Sort the columns in increasing order of their score CS_j

- *Step* 7: If the number of iterations is equal to $maxIter$, stop. If during *step* 4 or *step* 6 any rearrangement was necessary, go to *step* 3, else stop

Before applying ROC algorithm on L3NumericS, it is interesting to have a graphical interpretation of the distribution of 0 and 1 in the dataset. In Figure 4.4 is shown an heatmap of the first 100 rows of L3NumericS, where green cells represent the presence of a 1, while red cells represent the presence of a 0.

After the execution of ROC algorithm, rows and columns of L3NumericS are sorted as shown in Figure 4.5.

The algorithm generates a block-diagonal matrix, where it is possible to see two main groups or family of parts, where parts in each group are quite similar each other and are very different from parts of the other group.

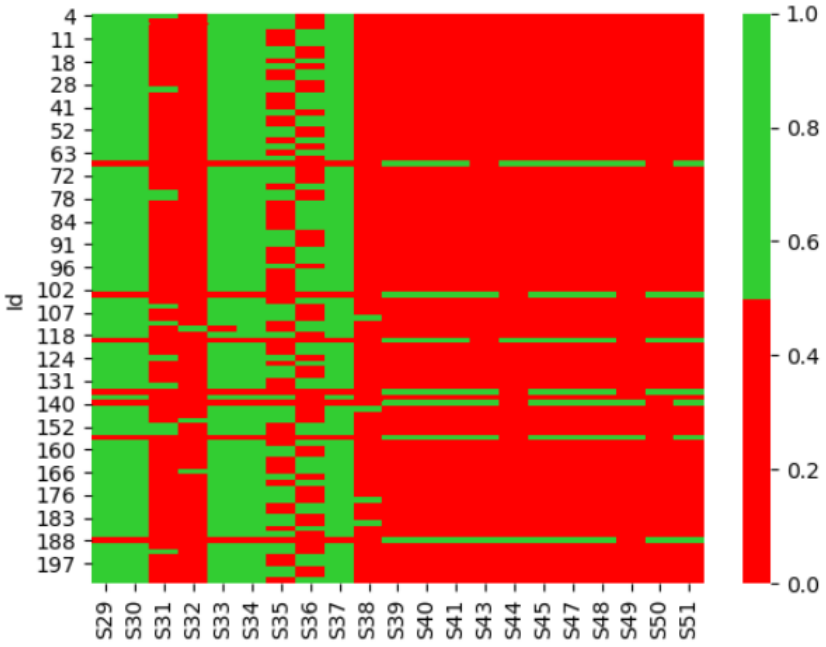


Figure 4.4: Heatmap of L3NumericS

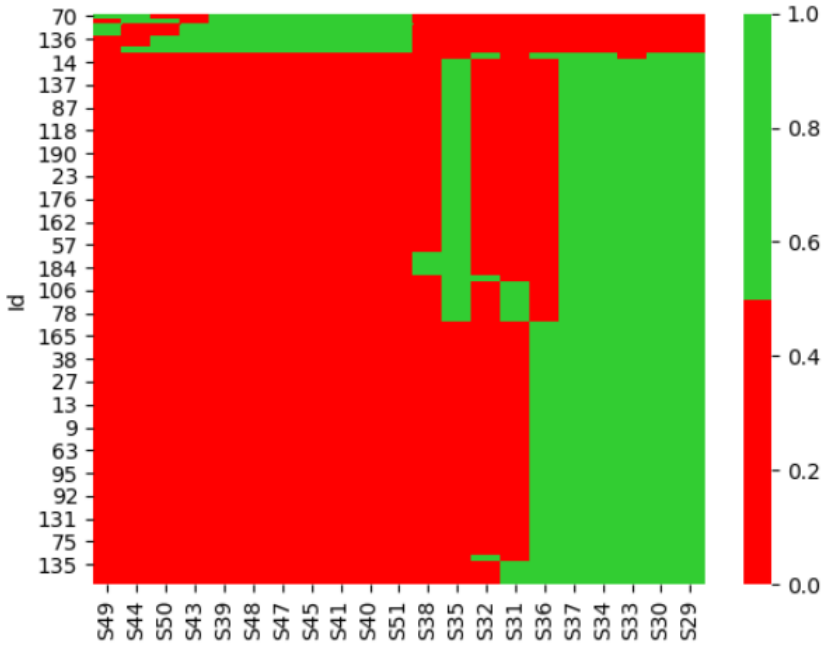


Figure 4.5: Heatmap of sorted L3NumericS

While ROC algorithm generates a very good graphical result, it has three main technical problems:

- Difficult to convert the graphical result to a real classification, indeed it doesn't produce as output groups of row ids, but only a 'sorted list' containing all the ids.

- Impossible to set 'how much similar' rows in the same groups are.
- Impossible to use the algorithm on a large matrix. Since it is necessary to generate two vectors containing increasing powers of 2 of length n and m respectively, it means to save very large numbers in each cell of the array. This algorithm fill computer's memory very quickly, making impossible to proceed.

These issues make necessary to reject the possibility to apply ROC algorithm on the whole dataset in exam.

4.1.2 Similarity Coefficient Algorithm

Another algorithm used in Group Technology problems is SC Algorithm. It works with a matrix filled with 0 and 1, the same format of the dataset L3NumericS used before. The algorithm generates groups of similar rows using a *similarity coefficient*. Differently from ROC algorithm, the output is not a sorted matrix, but a list of groups of row ids (i.e. a *dictionary*). A row belongs to a certain group if the similarity coefficient between the row and the group is higher than a fixed threshold and there is not a group that can generate an higher coefficient.

The most used coefficient is the *Jaccard similarity coefficient*: given two rows representing two distinct parts, it measures the ratio between the number of machines visited by both parts and the total machines visited by at least one of the parts. Similarity coefficient S_{ij} calculated between two rows i, j of the matrix L3NumericS with n rows and m columns, where a_{ik} is the entry in the $i-th$ row and $j-th$ column, is calculated in (4.3):

$$S_{ij} = \frac{\sum_{k=1}^{m} a_{ik}a_{jk}}{\sum_{k=1}^{m}(a_{ik} + a_{jk} - a_{ik}a_{jk})} \tag{4.3}$$

In SC Algorithm, similarity $S_{G_iG_j}$ between a certain group of rows G_i and another group G_j is defined as the minimum of the similarities S_{hk} calculated between each row of the first group $h \in G_i$ and each row of the second group $k \in G_j$:

$$S_{G_iG_j} = \min_{\substack{h \in G_i \\ k \in G_j}} \{S_{hk}\} \tag{4.4}$$

With these assumptions, defined a threshold of minimum similarity *minSim* required to a row to belong to a certain group, is possible to describe the SC algorithm:

- *Step* 1: Place each row in a distinct group, creating n groups G_i, $i = 1, \ldots, n$
- *Step* 2: For each group G_i, calculate the similarity coefficient $S_{G_iG_j}$ with all the other groups. Merge G_i with G_j if:

$$S_{G_iG_j} \geq minSim \quad and \quad S_{G_iG_j} = \max_k \{S_{G_iG_k}\}$$

This algorithm has some important characteristics:

- The solution is a real classification, since the result is a dictionary of groups, rather than a graphical classification
- Classification is based on a coefficient of similarity, that can be set arbitrarily
- The algorithm doesn't use a lot of memory, then it can be used on a very large matrix

The only problem found using the SC Algorithm is computational time. In Figure 4.6 is shown how much time the algorithm needs to classify different numbers of rows of L3NumericS.

Since the pattern is quadratic, classification of the whole dataset (more than a million of rows) is feasible, but very time expensive. A solution is to customize the original SC Algorithm, making it more efficient in analyzing the dataset in exam.

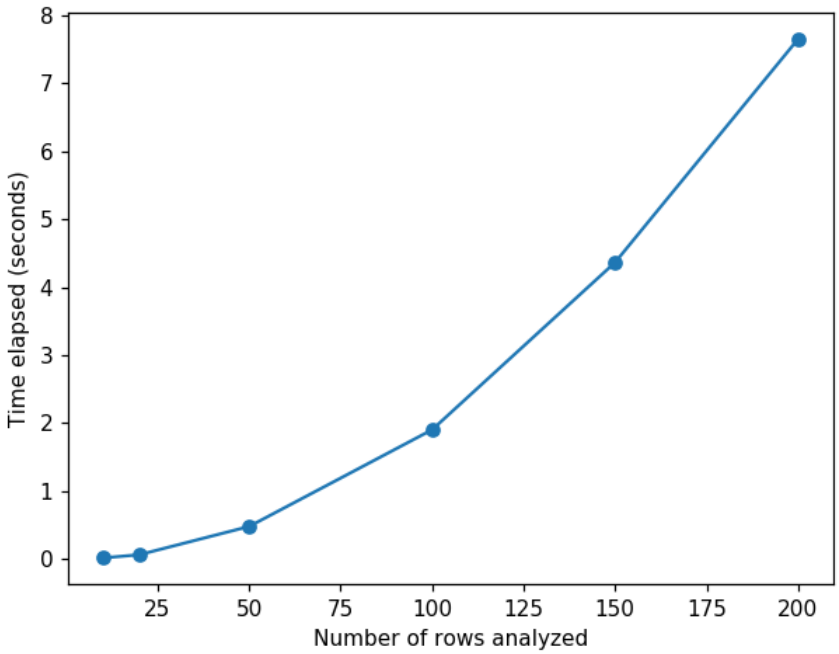


Figure 4.6: Computation time of SC Algorithm used on different numbers of rows of L3NumericS

4.1.3 Customized Similarity Coefficient Algorithm

Similarity Coefficient Algorithm can be adapted, in order to reduce the computational time. Since industrial manufacturing is based on production of a great number of identical products, is logical to suppose that grouping by Sections on the datasets in exam and replacing numerical values with 1 and NaN with 0 as done with L3NumericS, a lot of rows will assume exactly the same values. With this consideration, it follows that SC Algorithm spends a lot of time in classifying one by one a lot of identical row. Since checking if two rows are identical is much faster that calculating Similarity Coefficient between a row an the rest of the dataset, SC Algorithm can be Customized as follows:

- *Step* 1: Create groups of identical rows
- *Step* 2: From each group, select a single row
- *Step* 3: Apply classical SC Algorithm on the selected rows
- *Step* 4: Assign all the identical rows to the same group

Standard SC Algorithm and his customized version produce exactly the same result, while computational time is considerably different, in particular when working on a large number of rows. In Figure 4.7 is shown how much time Customized SC Algorithm needs to classify the same numbers of rows of L3NumericS analyzed by the classical version.

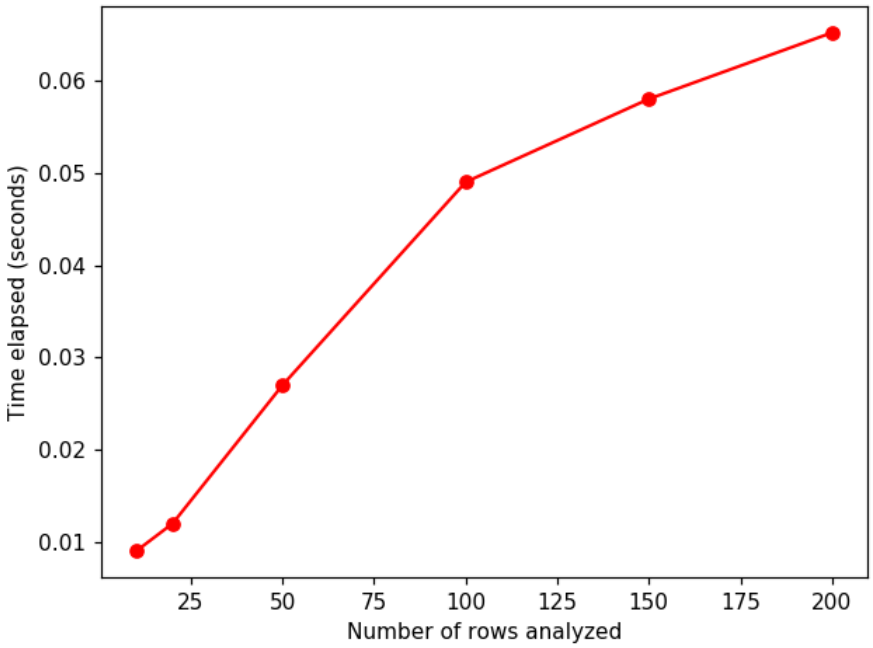


Figure 4.7: Computation time of Customized SC Algorithm used on different numbers of rows of L3NumericS

Pattern is initially quadratic, but assumes a sub-linear trend when the number of rows becomes bigger. In Figure 4.8 are compared the results obtained by the standard and the customized version.

Customized SC Algorithm is extremely less time-expensive, thus it has all the features required to solve the problem in exam.

Before applying CSC Algorithm on the whole data in exam, it is necessary preparing each dataset of each Line (i.e. L0Numeric, L0Categorical, ...,L3Date) as described in 'Data preparation' and joining all them together. Since these operations are very memory and time expensive, the problem can be optimized using only one dataset for each Line. Categorical datasets are not suitable for this purpose, since they are mostly filled with NaN; Numeric and Date dataset have a similar percentage of NaN, but

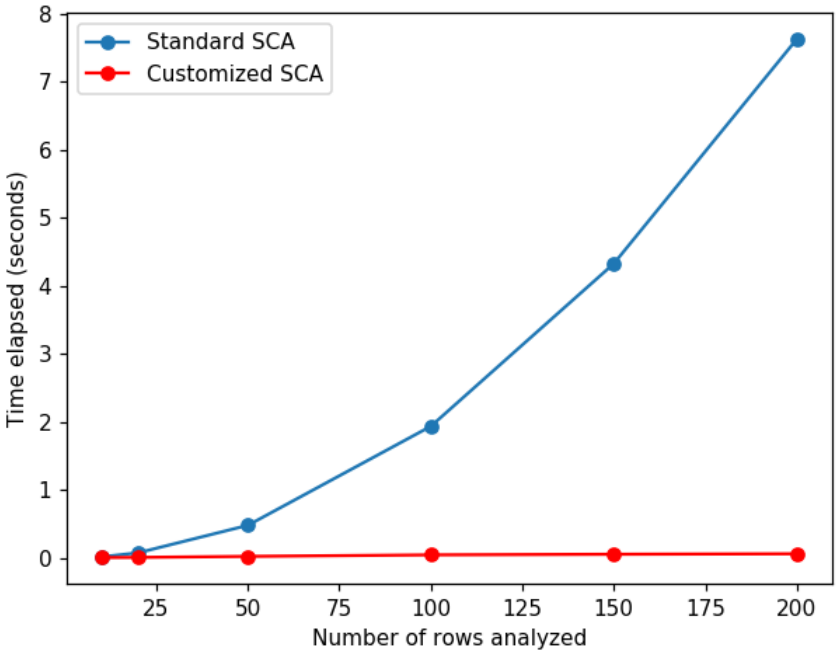


Figure 4.8: Comparation of computation times of Standard and Customized SC Algorithm

Numeric are more appropriate, because they contain more columns.

Established to apply CSC Algorithm on the dataset obtained joining L0NumericS, L1NumericS, L2NumericS and L3NumericS, it is necessary to set the Jaccard similarity coefficient: with a value of 0.75, the algorithm generates 1182 different groups. Despite the great number of elements generated, more than 70% of the whole data is contained in the 29 bigger groups.

4.2 Random forest

Each group generated by CSC Algorithm, contains data related to a different family of parts produced by Bosch factory. With the purpose to explain at least 70% of the data, is necessary to consider the 29 bigger groups. For brevity, will be considered only the biggest and the smallest ones. The names of the groups depend on the order they have been generated by CSC Algorithm:

- *Group 21*: is the biggest group generated, it contains 38219 rows. The total number of not null columns obtained joining all the datasets, selecting only the rows related to this group and converting Categorical values using contingency tables is 955. Only 0.46% of the row have Response=1 label.

- *Group 48*: is the smallest group generated, it contains 9257 rows. The total number of not null columns obtained joining all the datasets, selecting only the rows related to this group and converting Categorical values using contingency tables is 782. Only 0.36% of the row have Response=1 label.

All analyzes will be performed on both groups, using a software called 'R', dedicated to statistical purposes. Algorithms described in this section, have been found in [3].

Since Random Forest algorithm can not manage NaN values, it was decided to replace each null entry of the dataset with an outlier, that is a very anomalous value. Considering v_{ij} as the null value found in row i and column j of the dataset, it will be replaced with the value:

$$v_{ij}^{*} = max_j + 10 * (max_j - min_j + 1)$$

where max_j and min_j are respectively the maximum and the minimum value found into the column j.

4.2.1 Standard approach

Description

Random forest is a classification algorithm, very used in machine learning contexts. It is composed by a certain number (settable with the command 'ntree') of Decision trees, from which it derives its particular name. This algorithm is used both for classification and regression: since the problem is to distinguish between Response=0 and Response=1 parts, is necessary to use the classification version. A Decision tree is composed by nodes and leafs, in each node data are splitted in two child nodes, according to the value of a certain variable, with the purpose to separate the labels in the best way possible. A leaf is a child node that doesn't need splitting anymore, because all the elements (or at least the great majority), have the same label. At each node, the feature that separates better the elements is selected among a certain number of 'candidates' (settable with the command 'mtry'). To measure how well data are splitted, is necessary to define an index, that can be calculated for each node or leaf of the three, called *Gini Index*:

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{4.5}$$

Where c is the number of different classes and p_i is the proportion of elements of class i in the node in exam. Defining n_i the number of elements of class i i the node, p_i can be easily obtained as follows:

$$p_i = \frac{n_i}{\sum_{j=1}^{c} n_j}$$

The best splitting feature for the node k is the one that maximizes the *Gini index decrease*, D_k, calculated as the difference between the Gini index of the node k, $Gini_k$, and the weighted sum of the Gini index of his two child nodes, $Gini_{k_1}$ and $Gini_{k_2}$:

$$D_k = Gini_k - \left(\frac{N_1}{N_1 + N_2} Gini_{k_1} + \frac{N_2}{N_1 + N_2} Gini_{k_2} \right) \tag{4.6}$$

Where N_1 and N_2 are respectively the number of element in the first and second child node. Another possible setting is the maximum depth of a tree (settable with the command 'maxdepth'), that impose a maximum number of consequent splittings for the tree. An example of Decision Tree is shown in Figure 4.9.

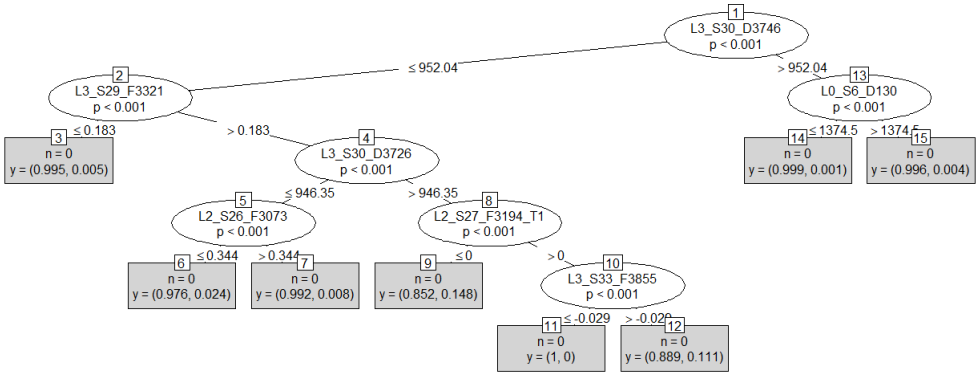


Figure 4.9: Example of decision tree

To classify an element using a Random forest means classifying the same element with all the trees generated and then using all the votes collected to assign a final label: usually is assigned the most frequent label (*majority vote*).

Application

The first approach is to apply a simple Random Forest. In Figure 4.10 is shown how to train a Random Forest with 20 trees, that considers 30 features at each split, with the purpose of predicting the binary value of Response, using all the other features of the dataset called *trainData*.

```
RF = randomForest(Response~. , ntree=20, mtry=30,data = trainData)
RF
```

```
##
## Call:
##  randomForest(formula = Response ~ ., data = trainData, ntree = 20,      mtry = 30)
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 30
```

Figure 4.10: R code: Random Forest with 20 trees, that considers 30 features at each split

The model is then applied on another dataset, called *testData*. A widely used tool used to evaluate the quality of a classifier is the *confusion matrix*, which compares model's prediction to the real solution

		Actual value	
		0	**1**
Prediction outcome	**0**	True negative	False negative
	1	False positive	True positive

Defining Response=0 as the negative outcome and Response=1 as the positive one, the four boxes of the matrix represent different results of the classification:

- True negative (TN): number of negative elements correctly classified
- True positive (TP): number of positive elements correctly classified
- False negative (FN): number of positive elements wrongly classified as negative
- False negative (FP): number of negative elements wrongly classified as positive

Many evaluation indexes can be calculated using these four values, the most useful in this context are:

- True positive rate (TPR): also called *Sensitivity*, measures the proportion of positive labels correctly identified. Is calculated as $\frac{TP}{TP+FP}$
- False positive rate (FPR): measures the proportion of positive labels wrongly identified as negative. Is calculated as $\frac{FP}{FP+TN}$

One of the possible outputs of a Random Forest is the probability of the element under analysis to belong to the positive class. A good way to

evaluate a classifier is to study the behavior of *TPR* and *FPR* when varying the value of the probability threshold necessary for the assignation of positive class label. The curve describing the behavior of these indexes is called ROC (Receiver Operative Characteristic) curve. The overall performance of a classifier is given by calculating the *Area Under the Curve* (AUC): an area near to 0.5 indicates a nearly random classification, while an area near to 1 indicates an almost perfect classification.

In Figure 4.11 and Figure 4.12 are shown the ROC curves obtained applying standard random forest algorithm on the test dataset of *Group 21* and *Group 48* respectively.

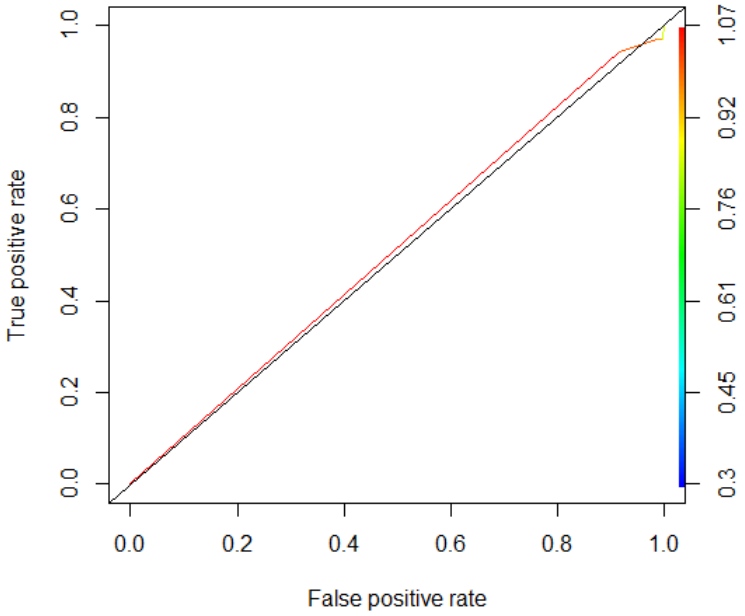


Figure 4.11: ROC curve obtained applying random forest algorithm on Group 21 test dataset (AUC=0.5121)

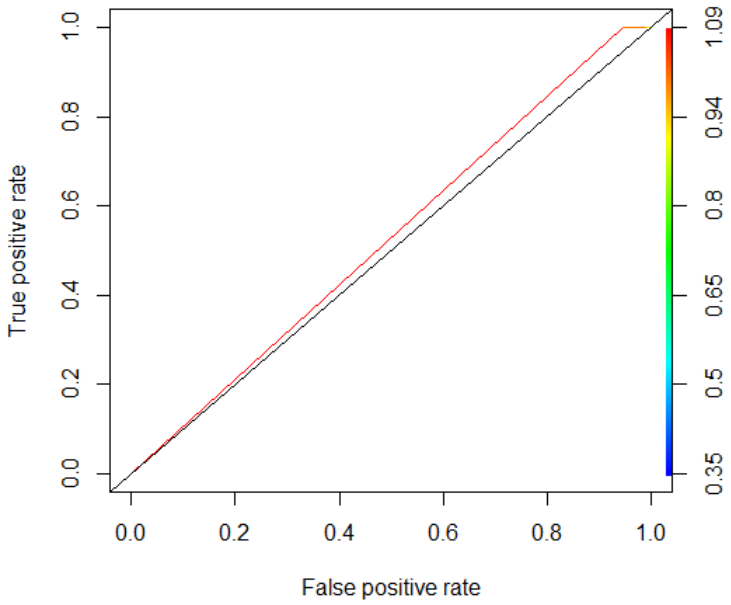


Figure 4.12: ROC curve obtained applying random forest algorithm on Group 48 test dataset (AUC=0.5274)

The classifiers obtained have a very bad performance. One of the causes of the poor classification obtained may be the strong unbalance of data, since both datasets contain less than 1% of Response=1 rows. Moreover, any kind of parameter tuning process is very time expensive, because of the high dimensional data. A different approach is necessary.

4.2.2 Downsampling approach

Description

Downsampling approach consist in training a Random Forest model on a subset of the original data. The subset contains all the element of the minority class (Response=1) and a randomly selected subset of the majority class (Response=0), so that class frequencies match, creating a balanced dataset. The dataset obtained contains less rows than the original one, thus this technique solves both the problems of high dimensional and unbalanced data.

Data balancing technique is a very powerful tool, but it needs to be used very carefully, in order to get a correct evaluation of model's performances, as described in [2]. First of all, only train data need to be balanced, while model obtained must be tested on a dataset were classes have the original unbalanced ratio. The same approach must be maintained while applying the cross-validation method: data must be split between validation and training set before class balance process. A wrong application of the downsampling algorithm can bring a not correct estimation of model's ability of class recognition, while a wrong application of oversampling algorithm (i.e. creating duplicates of minority class elements) can bring to even worse consequences, such as overfitting.

As an example, in Figure 4.13 is shown a wrong application of the oversampling algorithm, where duplicates of the minority class are generated before the execution of the cross-validation.

Is possible to notice that the same element appears in both validation and train set, causing the phenomenon called overfitting. In Figure 4.14 is shown the correct application of the oversampling algorithm, where data duplication is performed after the creation of validation set, in order to avoid any kind of overfitting.

The procedure described is automatically performed setting *sampling = "down"* inside the *trainControl* command, while training a Random Forest,

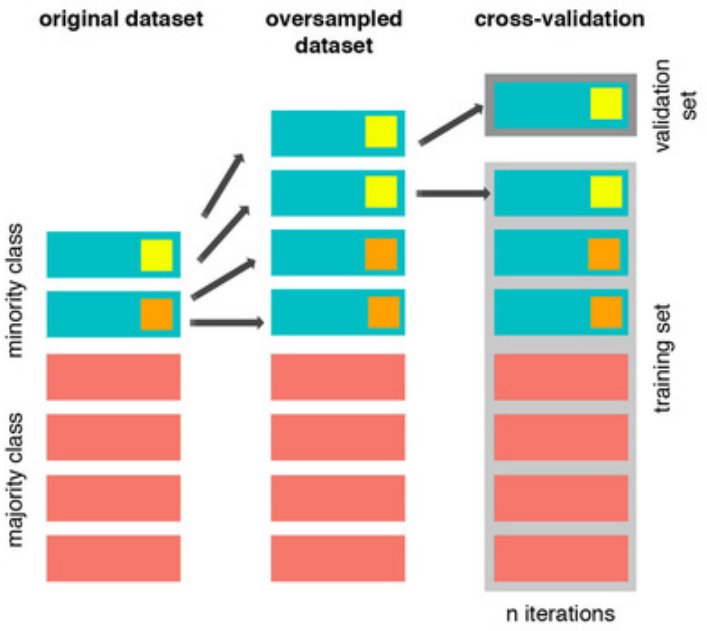


Figure 4.13: Wrong application of oversampling algorithm during cross-validation

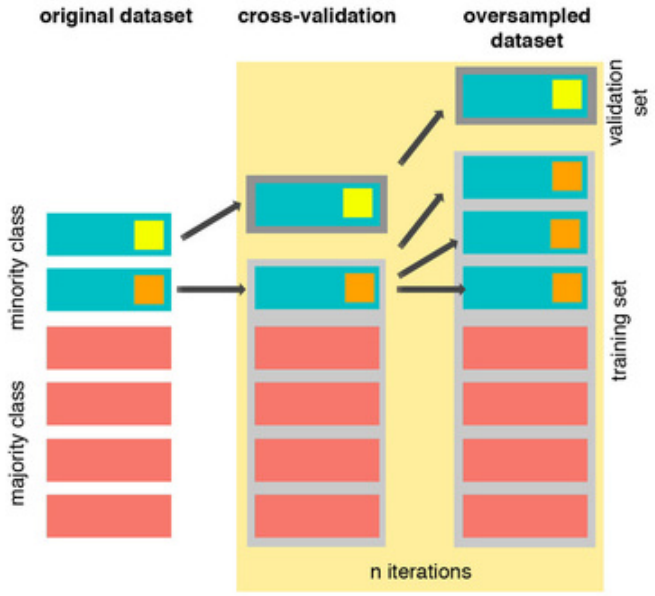


Figure 4.14: Correct application of oversampling algorithm during cross-validation

using the *train* function contained into the *caret* package.

Application

In Figure 4.15 is shown how to train a Random Forest using the downsampling option provided by caret package.

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     sampling = "down")


grid <-  expand.grid(mtry = c(40,43,46))

set.seed(42)
RF_under <- caret::train(Response ~ .,
                                 data = trainData,
                                 method = "rf",
                                 tuneGrid=grid,
                                 trControl = ctrl)
```

Figure 4.15: R code: training a Random Forest with downsampling

Looking at the settings contained in *trainControl*, is possible to notice that it has been disposed to perform a 10-fold cross validation, with the purpose to evaluate model's performances. In order to get a better result, different values of mtry have been tested. The resulting model has slightly improved performances: this means that downsampling approach is a good solution, but further tuning is necessary. One of the output provided by Random Forest algorithm is the 'variable importance', namely how much, on average, each variable decreases Gini index when used to split a node. As an example, in Figure 4.16 is shown the variable importance plot obtained by Group 21 dataset.

In order to get better performance, it is necessary to study interactions between variables: this can be easily performed replacing the model equation "$Response \sim .$", with a second degree equation "$Response \sim .\hat{}2$". Since studying every interaction consistently increases computational time, it is necessary to make a feature selection, in order to study only the most significant variables. Selecting only features with mean decrease Gini

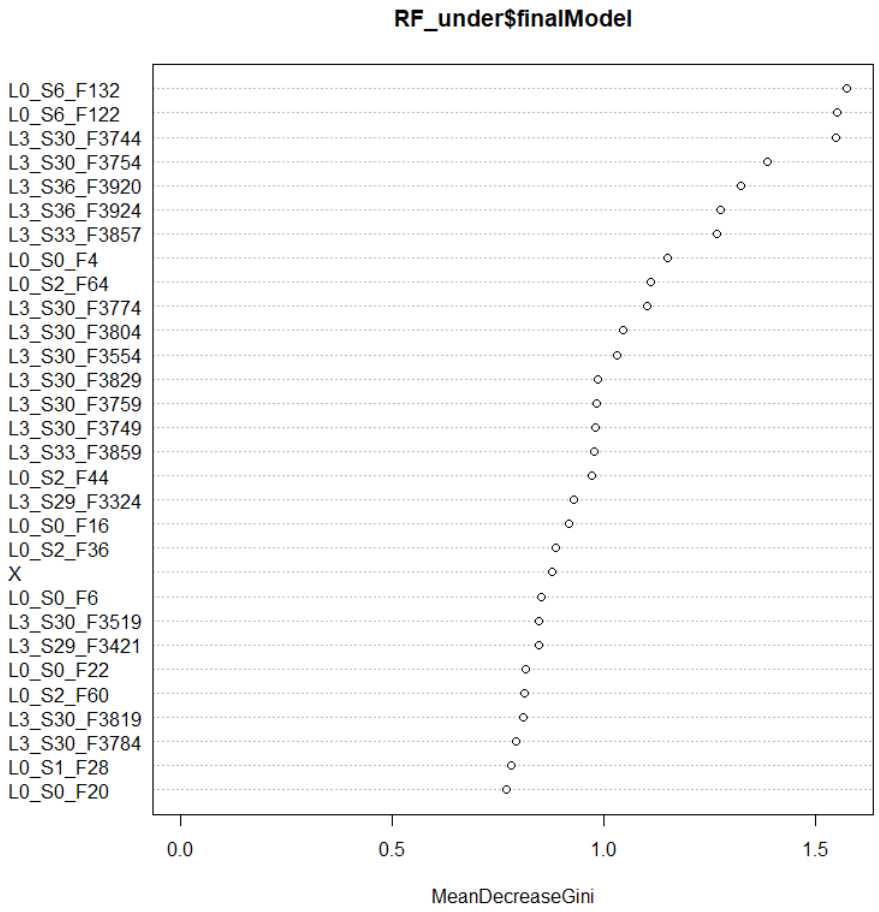


Figure 4.16: Variable importance plot of Group 21

greater than 1, a smaller dataset called *trainSmall* is obtained. In Figure 4.17 is shown how this dataset is used to train a Random Forest considering interactions.

The first two lines of codes are used to start a parallel computation using 5 cores (terminated in the last line), in order to reduce computational time. Besides mtry, more different parameters have been tested, such as *min_ samples_ leaf*, that specifies the minimum number of elements contained in a leaf node, *min_ samples_ split*, that specifies the minimum number of elements that a node must contain in order to be split again and *max_ depth* that specifies the maximum dept of the trees generated. All the models obtained with different parameters have been compared, with the purpose of selecting the best performing model. Utilizing the technique just described over the train datasets of Group 21 and Group 48, is possible to

```
cl <- makePSOCKcluster(5)
registerDoParallel(cl)
modellist <- list()

ctrl2 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 10,
                      verboseIter = FALSE,
                      sampling = "down")


grid <-  expand.grid(mtry = c(5,8,10,12))

set.seed(42)

for (min_samples_leaf in c(1,2, 4)) {
  for (min_samples_split in c(2,5,10)) {
    for (max_depth in c(NA,20, 60, 100)) {

      RF_under2 <- caret::train(Response ~ .^2,
                                data = trainSmall,
                                method = "rf",
                                metric='accuracy',
                                min_samples_leaf=min_samples_leaf,
                                min_samples_split=min_samples_split,
                                max_depth=max_depth,
                                trControl = ctrl2,
                                tuneGrid=grid)

      key <- paste(toString(min_samples_leaf),"/",
                   toString(min_samples_split),"/",
                   toString(max_depth))
      modellist[[key]] <- RF_under2
    }
  }
}
stopCluster(cl)
```

Figure 4.17: R code: training a Random Forest with downsampling, interactions and parameter tuning

create models with good performances: in Figure 4.18 and Figure 4.19 are shown the ROC curves obtained.

Final models obtained can explain quite well data behavior, thus is possible to analyze variable importance in order to detect root causes of the failures.

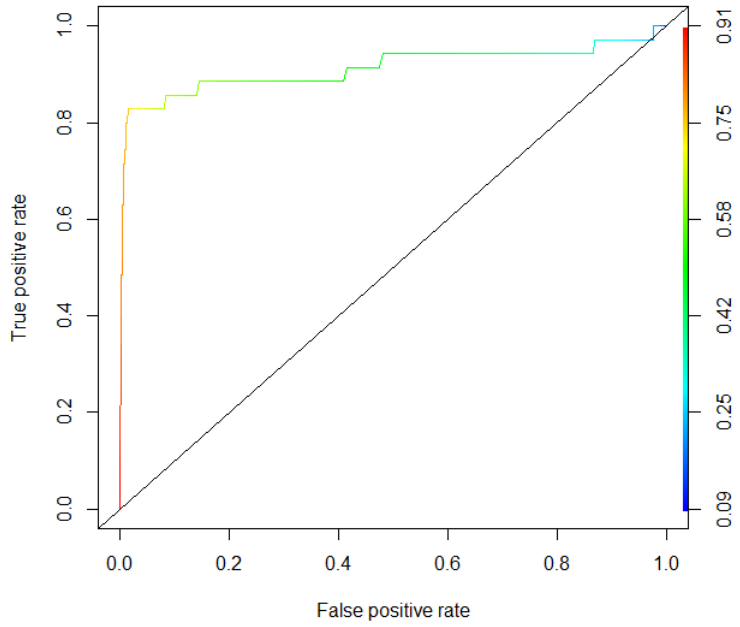


Figure 4.18: ROC curve obtained applying the best performing model on Group 21 test dataset (AUC=0.9129)

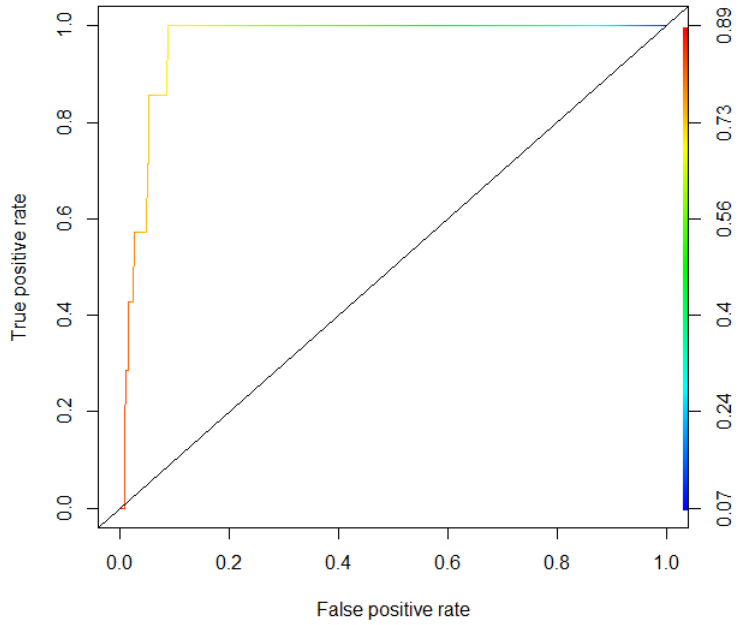


Figure 4.19: ROC curve obtained applying the best performing model on Group 48 test dataset (AUC=0.965)

Chapter 5

Results

Looking at the variable importance plots obtained by the best performing models, is possible to find the root causes of the failures during the production chain.

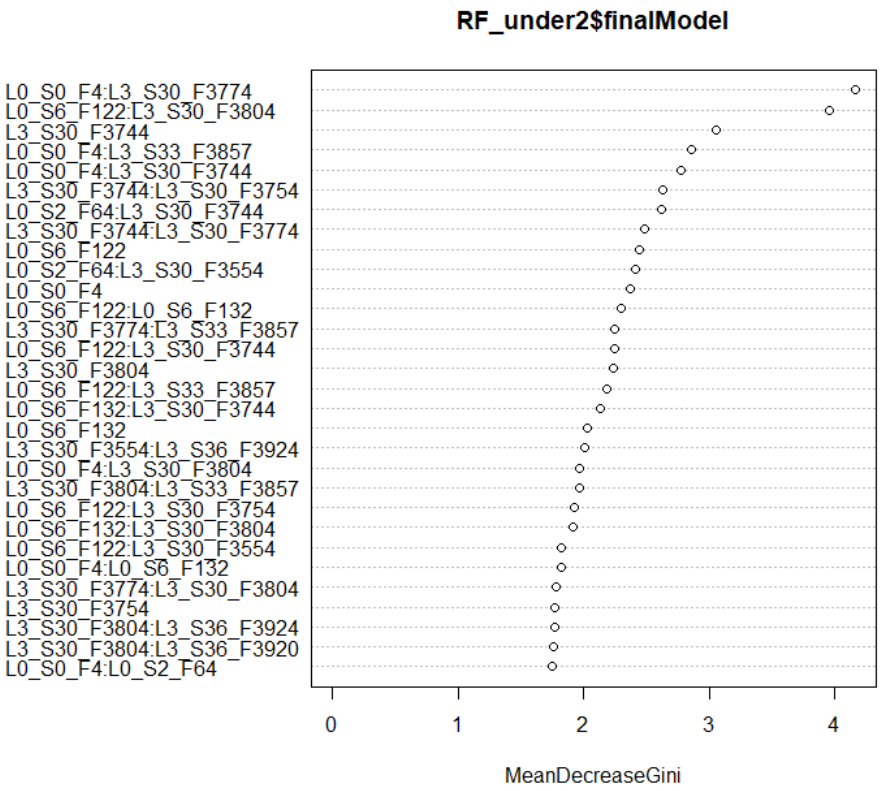


Figure 5.1: Variable importance plot obtained by the best performing model of Group 21

In both Figure 5.1 and Figure 5.2 is possible to notice that the most

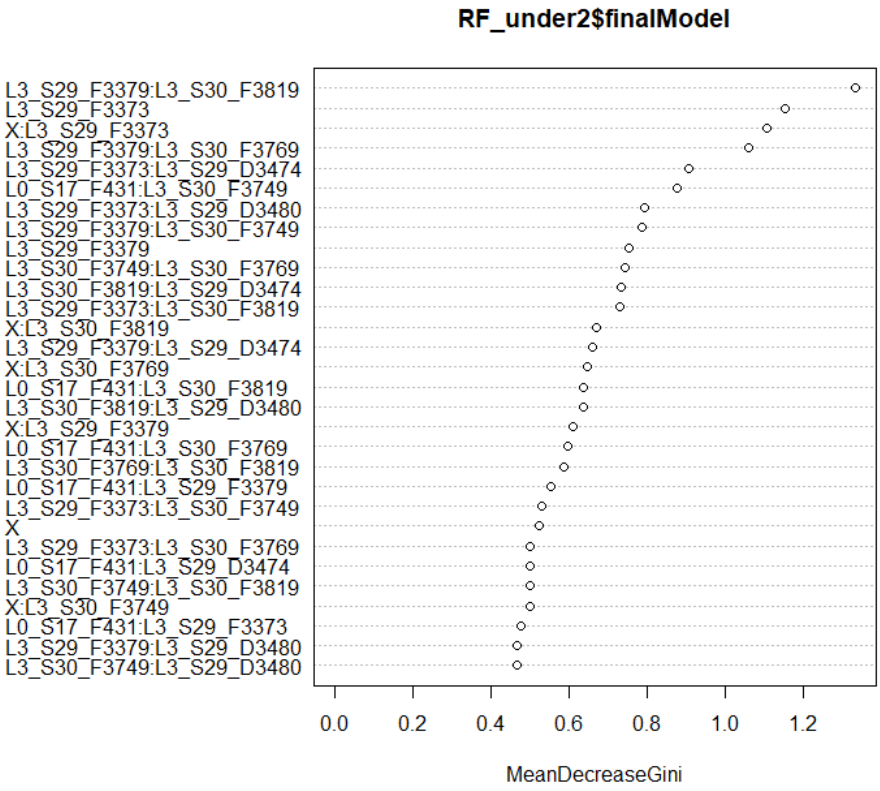


Figure 5.2: Variable importance plot obtained by the best performing model of Group 48

important causes of failures are related Line 3, in particular to Section 29,30 and 33. The following step in a real Root Case Analysis problem, would be the inspection of machinery related to these Sections, in order to detect any sort of malfunction.

5.1 Application

Results obtained studying Bosch datasets can be applied to many industrial situations. One of the most suitable cases of application are data provided by one of the companies participating in the FDM project. The dataset in exam contains the output generated by a machinery while processing raw food, in order to reduce its humidity level. Data are confidential, so the name of the company and the kind of food processed will not be declared.

The drying process analyzed is still a prototype, then only 15 runs have

been recorded at the moment. Actual process consists in the following steps:

- *Step 1*: Test in laboratory the humidity level of the food
- *Step 2*: Start the drying process
- *Step 3*: Every 2 hours stop the machinery and test humidity
- *Step 4*: If humidity percentage is lower than 6%, stop the process, else restart from *Step 2*

The goal of the analysis is to skip *step 2*, reducing the number of laboratory analysis predicting food humidity only using machinery outputs and discovering which variables affect humidity levels the most, in order to optimize the process.

Since the process stops when ideal humidity level is reached, only a few rows describe the ideal state of the product, this means data are very unbalanced. Data produced by machinery's sensors have already been pre-processed and organized, in order to make possible inspection activities. The model obtained applying methods described in previous chapters explains quite well data behavior: in Figure 5.3 is shown the variable importance plot.

As it could be expected, the variable that influence humidity level the most are *time*, that indicates how much time has passed since the beginning of the process, *umidita_iniziale*, that indicates the initial humidity level, and *HMI_grammi_persi_filt*, that indicates the total loss of weight of the food during the process. All machinery's sensors have a similar importance, this means that there are not bad-working parts, then no correction of the process is needed.

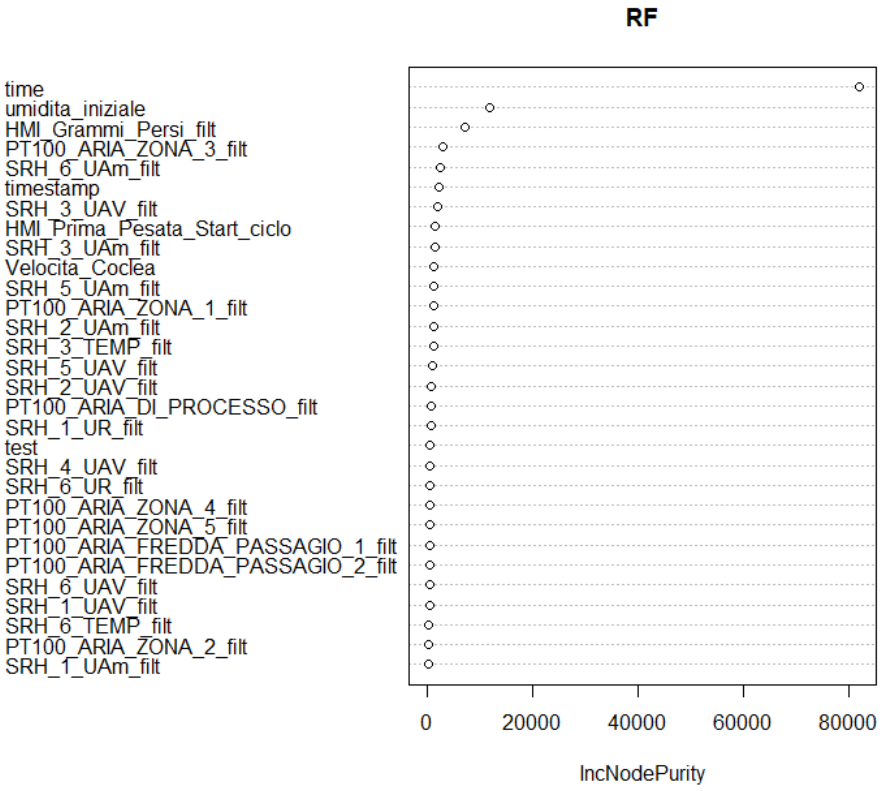


Figure 5.3: Variable importance plot obtained by the best performing model of FDM dataset

5.2 Conclusion

In this Thesis various issues like data unbalance and heterogeneity have been faced while studying Bosch dataset. The innovative resolution pattern formulated can be applied to many different industry 4.0 contexts, in order to detect and prevent failures, reducing wastes and improving efficiency and quality.

Food manufacturing is a very suitable field of application, thus Food Digital Monitoring project, despite being still under development, will certainly get great benefits from the application of the pattern created and will provide the opportunity for further developments.

In conclusion, research and improvement of manufacturing digitization can provide more information and assets to ensure new data-driven business models.

Bibliography

[1] R. G. Askin and C. R. Standridge. *Modeling and Analysis of Manufacturing Systems*. Ed. by John Wiley and Sons. 1993.

[2] *Dealing with imbalanced data undersampling oversampling and proper cross validation*. `https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation`. Accessed: 2019-03-20.

[3] T. Hastie et al. *Introduction to Statistical Learning*. Ed. by Springer. 2013.

[4] Sunderesh S. Heragu. *Facilities Design*. Ed. by Taylor & Francis Group. 2008.

[5] *Kaggle*. `https://www.kaggle.com/c/bosch-production-line-performance/data`. Accessed: 2019-03-20.

[6] J. R. King. "Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm". In: *International Journal of Production Research* 18 (1980).

[7] Robert J. Latino, Kenneth C. Latino, and Mark A. Latino. *Root Cause Analysis*. Ed. by Taylor & Francis Group. 2011.

[8] Li Da Xu, Eric L. Xu, and Li Ling. "Industry 4.0: state of the art and future trends". In: *International Journal of Production Research* 56 (2018).

[9] *mise*. `https://www.mise.gov.it/images/stories/documenti/Piano_Industria_40.pdf`. Accessed: 2019-03-20.