POLITECNICO DI TORINO

Master degree course in Mechatronic Engineering

Master Degree Thesis

A Machine Learning Technique for Predictive Maintenance and Quality in Cut Glass Machinery

POLITECNICO
DI TORINO

Supervisors
prof. Edoardo Patti
Andrea Acquaviva
Lorenzo Bottaccioli
Luciano Baresi

Candidate
Giacomo Ornati

2019 October

Abstract

The study presented in this thesis work is based on the development of a machine learning project applied to the particular case of the *548 Lam* machine, a cut glass machine produced by the company Bottero s.p.a., which collaborates with this thesis. The work describes how to develop a project based on machine learning and how it can be applied in a real case. The thesis is in fact divided into two distinct parts, the first more didactic in which are explained the various steps to be followed to prepare the data and apply different algorithms of machine learning to maximize the results, the second instead shows how to use in a concrete way the results of the study of machine learning in order to effectively increase productivity. To do this, two problems have been selected to be solved, indicated by the company.

The first study tries to predict machine stops and anomalous failures. The work includes an accurate understanding of the machine and the starting dataset, and then concentrate on the data preparation. Therefore, all the various steps to be followed during the pre-processing phase are listed: how to perform data merging, correlational and statistical analysis to find important information from the data, feature selection, how to manage categorical features. When the initial data have been properly manipulated, we proceed to the evaluation phase of the selected algorithms based on the characteristics of the dataset, evaluating which model obtains the best results in predicting machine errors. The results obtained show that the probabilistic algorithms based on the tree classification are those that best predict errors. In details, the Random Forest Classifier proves to be the best model obtaining an F1 score of about 50% on the positive prediction.

The second part deals with the prediction of machining times based on the characteristics of the work to be performed. This second part want to show how a machine learning model can be transposed from the mere field of study to a real application, so there is a reconstruction of the prediction function inside the machine itself, in order to make real time predictions based on the data selected by the user. The linear regression model was first trained on a PC thanks to the data contained in the dataset and then reconstructed in the software interface of the machine thanks to the coefficients listed in an orderly JSON file that is first created during the training of the model and then passed to the machine that is at this point able to make predictions. The results show that the time prediction based on this model achieves an error of less than 10% with respect to the actual measured values.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope of this work and problem formulation

This thesis work was born from the will of the company Bottero s.p.a. to expand its knowledge in the Internet of Thinks (IoT) and Artificial Intelligence (AI) field. Specifically it acts on data collected by one of their machines: the *548 Lam* for cutting laminated glass. The purpose of this thesis is to try, thanks to the analysis of the data collected by these machines, to predict its possible behavior in a perspective of predictive maintenance and predictive quality.

More in detail, the work that will be presented consists of two main parts: **Error Prediction** and **Time Prevision**. Both are works related to the world of Machine Learning (ML) that use as a starting point the Bottero database full of information from machines already produced and in operation in the possession of various customers around the world.

The initial purpose of the database was to collect information for statistical purposes of the company. Bottero then realized the potential of the collected data that can be used for more complex and productive purposes such as increasing the quality of the machine, avoiding errors or adding functionality in a simple way thanks to the fact that the IoT ecosystem was already complete from the point of view of data collection, to complete it was missing only a prudent use of the database to achieve improvements and optimizations. This prudent use is a correct application of artificial intelligence that allows to obtain substantial achievements only thanks to data analysis.

This work therefore represents common parts for both Error Prediction and Time Prevision that are: a study of the machinery, of the working

process and of the database containing the information related to the latter in such a way as to be able to find the machine learning algorithms that best behave with the object of study in question. Then a detailed study for each of the two sections will be performed: it will analyze the main characteristics of various algorithms known in the literature, highlighting their pros and cons. Finally, the results obtained by implementing these algorithms in the case study will be presented.

1.2 Error Prediction

This first part of the thesis tries to solve an uncommon problem on the *548 Lam* but potentially very harmful to the production process. At the moment it can happen that the machine stops and the whole work chain stops, or that the machine produces pieces out of standard. The problem is currently solved a posteriori, i.e. when the abnormal stop has occurred or the failed piece is produced an operator blocks the production line and manually restarts the machine. This involves a huge waste of time and work, moreover, it can happen that the machine ruins the workpieces, being a machine for cutting glass, a failure of this type can mean the replacement of the entire glass plate being processed, causing great frustration and loss of time, money and energy to restart the line.

The proposed novelty seeks then to obtain correlations between input to the machine and consequent output, so as to avoid out-of-standard or otherwise unacceptable outcomes, or even when this combination of inputs results in an interruption of the process that causes inconvenience to the entire production line. This is what can be defined as a **predictive quality**, in fact on the basis of certain features known a priori or set by the user it is possible to try to anticipate the operation of the machine and to predict whether a certain combination of inputs may result in a bad piece or may lead to a stoppage of the machine with consequent loss of work hours and money. This concept goes beyond machine maintenance and is independent of it, because even a new or revisioned machine can lead to damage if the sequence of input parameters is critical for a certain specific task. Given the amount of information and parameters at stake, it was decided to rely on artificial learning because in most of the negative events in question it is impossible or very difficult to understand the reason of the anomaly. In this optics a sort of artificial intelligence can be able to carry out the correlations necessary to avoid this type of behaviour.

This means that if you are able to effectively predict a possible crash or a badly finished piece, you will have a significant saving of time and money because instead of restarting the production line by manually reloading

the machine with a new glass plate, you only have to change the set of inputs provided to the machine which will then resume the work process with a minimum loss of time. Alternatively, if the initial inputs would create a situation of risk, the software would be able to signal it before the processing starts.

Technically the error prediction output can lead on machine learning problems called **binary classification**, where the result of the algorithm will be only 0 or 1, this information has to be taken into account when choosing the prediction algorithms.

1.3 Time Prevision

Even the time forecast part is based on machine learning algorithms, this time we want to predict the machining time of a machine in order to anticipate the customer the duration of the operation. Up to now, the problem of predicting processing times has been partially covered by a deterministic approach. Partly because the *548 Lam* machine has many operating modes and parameters to set. All of this information together creates a huge combination of possibilities. A deterministic approach requires the creation of formulas for each of them. The state of the art selects the most promising parameters and estimates on the basis of these processing times. This deterministic approach is repeated only for the most used operating modes, demonstrating great shortcomings also from the point of view of performance, reaching errors greater than 50% between the prediction and the real value.

The proposed novelty here is linked to the prediction of each single sub-processing performed (from now on defined step). The steps are linked directly to the parameters, the variation of these affects the machining time. The machining modes combine different steps and in different numbers, once the machining mode is known, the individual steps are predicted and then added to have the total machining time for any combination. Since the prediction is to be made in a continuous range of possibilities, this second part of the work will be based on algorithms on a part of the machine learning called **regression**.

Unlike the first part, this second work will also explain how to directly integrate the time prediction algorithm into the machine software, so that the operator, even before starting to work on a glass plate, will know how long the machine will take to finish the job.

1.4 How the work was structured

The developing of this thesis took few mounts because of the knowledge required for a good understanding of the problem. So first step was to know the machinery and the team who developed it. The machine as we will see in the next chapter is able to perform really complex tasks and manage multiple and completely different types of situations. The criticality in this step was to associate the data provided from the machine with the actions the machine performs. Then was possible to start viewing at the database collection of Bottero s.p.a. that is quite new and still under developing. This non stationarity comported lot of adaptations to every change in the database format. After having met all the tools needed for the work was the time to study the literature of the possible machine learning techniques to apply for this case study. Then it is possible to go in a more concrete direction describing the setting up of the developing environment that is related, just for introduction, with the use of SQL Server by Microsoft, with Jupyter Notebook and Pandas for managing data structures and for rearranging them and with the use of some ML libraries such SKlearn. Then it is possible to apply some of this techniques and get the results, obviously the rearranging of the implementation is crucial to obtain good outcome, so in the chapter related with this part the reiteration is part of the develop. Finally will be the time of conclusions and suggestions for eventually improve the work based on different type of informations obtainable from the machine. In details this work is structured as follow:

Chapter 2 : Chapter dedicated to the description of the machine, its interface and the peculiarity of the production process.

Chapter 3 : Here are descriptions of the database format, structure and tips on its use.

Chapter 4 : Description of the main algorithms in the literature and search for possible machine learning techniques both for classification and forregression problems.

Chapter 5 : Error Prediction pre-processing and unification of the various data in the database, eliminating discordant values or repetitions.

Chapter 6 : Error Prediction algorithms evaluation and results obtained for this first part, specifying which problems there were in the adaptation of the algorithms and in the initial data.

Chapter 7 : Time Prevision solution, here the preprocessing phase is less important because in part already done, so the goal is not only

to obtain the best ML algorithm to use, but is to create a real possible implementation in the machines able to predict the time of operations. So here the approach is more concreate also because the problem is more complex and with possible further applications.

Conclusion : final observations, comments and reflexions on this work.

Chapter 2

The machinary and the process

Bottero has been manufacturing machinery and plants for glass processing for over 50 years and is now a world leader in the sector. It is organized into 3 business units: flat glass, large plants and hollow glass. Over the years the company has developed many machines and the one under study in this thesis is the result of decades of experience and innovation. The machinery in question belongs to the flat glass business unit, its trade name is *548 Lam*, as the name indicates its purpose is the cutting of laminated glass sheets.

Under certain conditions it is fully automatic machinery. The automatic mode is particularly suitable with the use of *optimizations*: given the input glass format, the machine software is able to generate the output pieces in order to optimize the cuts minimizing the waste of glass from the plate, so it is able to calculate the best sequence of cuts to be made to obtain all the desired parts. Once the plate is loaded on the machine, it is automatically managed and processed obtaining in output all the pieces cut and transported elsewhere. For special cuts or for less frequent requirements it is possible to use the machine also in manual mode. Unlike the automatic cuts that work on the entire glass sheet, the manual cuts are set on the single piece that is loaded manually on the machine, when set it will perform the cut. The task of the *548 Lam*, in addition to cutting the glass, is to move, align, rotate the glass sheets in order to make the process fast and efficient. This has led to a great reduction in the time required for these operations.

The *548 Lam* derives from a model capable of performing very similar tasks, the model *558*, which can be defined as the father of the machinery

in question. Compared to the *558*, the *548 Lam* is cheaper, more efficient and easier to maintain, which has allowed it to be widely used and appreciated by customers.

The machine has been designed to work in a continuous production line integrated with other Bottero machines, such as loaders, overhead cranes, conveyors, etc ...so that every single machine can be seen as an element of a production line able to manage all the processing of the plates including loading from the warehouse, transport, positioning, cutting and unloading.

Figure 2.1: Basic setup of a *548 Lam* machine

2.1 Functional groups

The machine is made up of various macro parts that can be identified according to the task they perform. Some of these parts are standard and normally combined with the central body, even if there are many setups to satisfy every kind of customers. In any case, all the optional parts can be integrated with the main body of the machine. The parts described are those related to the normal setup of the machine, here are not described all the other possibilities offered by the company for simplicity and also because they have no influence on this work. The main parts are:

T-T-S Module *(Taglio Troncaggio e Stacco* in italian) is the heart of the machinery, here is concentrated most of the technology used and developed and patented by the company over the years. This part is composed of two fixed bridges characterized by:

- A locking system of the plate to keep it in position during cutting, this result is guaranteed by special rubberized air chambers that once inflated ensure an even distribution of forces on the plate.
- A highly efficient heating element (lamp) for heating the plastic inside the laminated glass. This is composed of a single mobile

element of small size easy to replace and much cheaper than the solution adopted on the *558*.

- A support surface for the module with belts capable of carrying the piece under the bridge for cutting.
- A gripper that working in symbiosis with an air suction cup mounted on the work bench is able to rotate the workpiece to perform transverse or diagonal cuts.
- A blade for precision cutting of the plastic inside the laminated glass sheet.
- Two plate engraving wheels create the line of weakness where the glass will be cut off.
- The breakout bar and wheels that hit the glass and cause the glass to split.

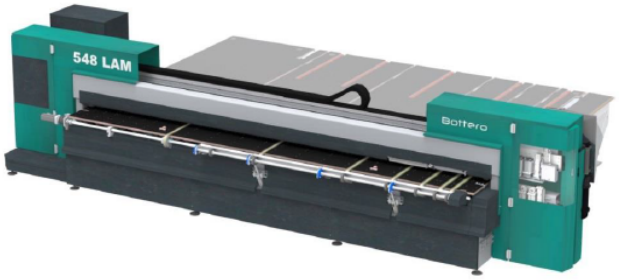

Figure 2.2: TTS module of a *548 Lam* machine

The figure 2.2 shows the TTS module.

Loading table It is a table used to transport the glass sheet to the bridge where it will be cut. It is composed of transport belts and can sometimes be replaced by other Bottero machines for in-line work. The loading table is shown in 2.3.

Workbench This is another work table that serves for the correct positioning of the piece of glass under the bridge, it is an essential component of the machine. Here there is also the suction cup that is used together with the bridge clamp to rotate the piece. The rotation takes place through a connecting rod-hand crank system that keeps the length between the two points of contact with the glass fixed and in the meantime moves them causing the rotation of the piece. In the figure 2.4 there is a picture of the workbanch.

The machine is sold in different layouts and sizes. The equipment are *basic, semi-auto, fully-auto, LTM* where the services offered are increasingly greater and with a higher level of automation, while the

Figure 2.3: The loading table of a *548 Lam* machine

Figure 2.4: Workbench of a *548 Lam* machine

sizes are *548 Lam - 38, 548 Lam - 49, 548 Lam - 61* where the last number indicates the size of the bigger glass plate achievable (they stay for 3800, 4950 and 6100 mm respectively, so it indicates the size of the machine). The minimum cut is about 20mm from the edge, that is a very little measureand require lot of expedients that will be cited after in this chapter while speacking about the process of cutting the glass.

2.2 Phases of interest and criticalities

As seen, the operational possibilities of the machine are many and variable depending on the chosen equipment and the operations to be performed. Briefly summarizing the main actions that the machine can perform are:

- Positioning
- Cut
- Diagonal cut
- Train discharge
- Transport

- Rotation
- Reverse transport
- Scraping

The details of these operations and others more remain omitted for simplicity. The important message to be leaked is that of all the possibilities offered by the machine has chosen to focus on the main ones. Among the various, the most complex, delicate and essential operation is the cutting. This is the fundamental step to be taken into account, moreover in addition to the most important it is also the most complex because it consists of various sequential phases. The research work of this thesis will start from this operation, trying to predict if the cut operation will have a good output or not. The second part will be focused on the time prevision not only for this step but for all the steps that the machine perform in an automatic manner.

Let's now analyze some of the various failures that may occur to the machinery. These failures are real eventualities and have been selected from a long list in the Bottero software documentation. Here are only the most important and easiest to guess for what has been explained so far on the machinery, are omitted those that require a thorough knowledge of the machinery:

- Vacuum missing in sucker for the movimentation of the glass.
- Glass search failed, the glass is bad positioned on the table, some photocells do not read the glass.
- Wheel worn has to be replaced.
- Invalid position for cut.
- Servo error.
- Truncation cycle error.
- Exhaust error.
- Pression lost.
- Error reading workpiece height.
- Trim not fallen in trim box.
- Rotation error.
- Blade broken.
- Glass mismatch.
- ...

As you can see in every phase there may be errors and failures of various kinds that normally lead to the block of the machine for safety. Here, however, you can see that most of the possible problems occur during the cutting phase, confirming what has been said above, the cutting phase is the most delicate and the errors in this phase concern problems during the breakout, the take-off, the blade, the clamping pressures, ...

After this further confirmation of the importance of the cutting phase, we will now analyze in detail the general process of separation of laminated glass sheets to better understand how the machine works.

2.3 The process of cutting the laminated glass

In order to better understand the glass cutting phase that takes place in the TTS module of *548 Lam* (cutting, breakout and take-off), the process of separation into parts of laminated glass is described in general terms.

Laminated glass consists of two or more layers of monolithic glass permanently and thermally bonded under pressure with one or more plastic interlayers of Polyvinyl Butyral (hereinafter *PVB*). It is possible to combine different glass thicknesses with different layers of PVB to obtain the desired properties. Usually laminated glass is identified by three digits corresponding to the mm of thickness of the glass and the number of layers of PVB used to join the parts. Each layer of PVB has a thickness of 0.38mm so a glass described by the acronym 3-2-3 is composed of two outer sheets of 3mm glass interspersed with 2 layers of PVB of 0.38mm for a total thickness of 6.72mm. The *548 Lam* machine is able to work with glass in the thickness range between 2-1-2 and 8-12-8 glasses (i.e. between 4.38mm and 20.56mm). A general structure of laminated glass is shown in fig. 2.5. Now it is possible to understand why laminated glass is also called stratified glass. This laminated glass composition ensures safety, so laminated glass is widely used in window and door frames and flooring. Other properties of this glass are therefore the greater resistance to impact and stress than normal monolithic glass and the seal in case of breakage, in fact when broken this glass does not collapse but its fragments remain attached to the layer of PVB reducing the risk of injury and increasing the safety against injury. Finally, laminated glass generally increases the level of soundproofing and blocks a high percentage of ultraviolet rays, these are two other qualities for which this glass is now widely used.

The downside of the medal is that compared to monolithic glass, it requires longer processing and is more expensive. Even the cut is much more complex, now we will analyze in detail the process of separation.

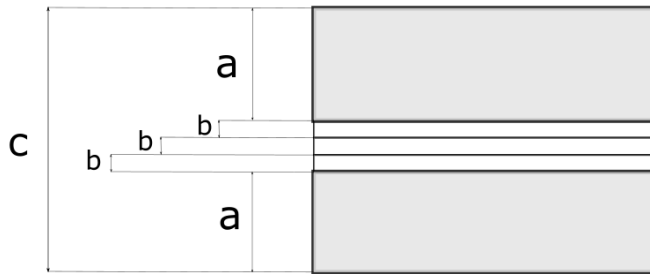

Figure 2.5: Laminated glass general structure. Usually dimension a is equal for both the glasses but can vary in range 2-8mm while dimension b is fixed at 0.38mm, in this case the number of layers of PCB can vary. C is the total high of the glass sheet.

The steps to be taken in order to obtain a correct, clean and chipless cut of the laminated glass are now described in sequence.

1. The first step is a **engraving** on the surface of the upper and lower glass: this engraving is performed by two toothed wheels, similar to the pinion of a bicycle, and create micro grooves that weaken the glass along that line that in jargon is defined as the cutting line. The result of this operation is the creation of an engraved path that has a dashed shape almost like a paper card prepared to be removed manually.
2. The second phase is the **breakout** of the glass. The separation of laminated glass is not a real cut is just a breakage of the glass along a line of weakness. The line of separation is, in this case, the line engraved in the previous step. There are two truncations, upper and lower. They can be made by means of a roller or a bar. The simplest and most immediate is the one with the bar in which you give a sudden and violent blow to the glass throughout its length. The breakout is immediate, the glass breaks exactly on the line of weakness. With the wheel instead the process is different: a wheel starts from one side of the glass giving a sudden blow on the opposite side of the engraving, so a local separation of the glass takes place, then the wheel maintaining a certain pressure runs along the entire length of the sheet making the breakage propagate till the opposite edge. For technical reasons related to the necessary space, the *548 Lam* uses both methods of breakout. It is important to note that the terms breakout upper and lower refer to the glass that is separated, to have a breakout you have to create pressure from the opposite side of the sheet so that you create on the glass a zone of tension that splits the glass. It is to be noted instead that the glass resists at compression so in the part of the blow the glass does not collapse. Moreover, for

obvious reasons, the glass must be supported on the opposite side to where it is hit. The support distance and the thickness of the glass affect the pressure to be exerted to obtain the separation.

3. Now the glass is detached from the top and bottom but the two parts are still joined by the middle layer of PVB that must be cut. This phase is the **separation and heating phase**:you have to enlarge the two sheets to allow then to pass a blade that will cut the plastic. To obtain the best results, the separation does not take place cold, but a lamp formed by an electric heater heats the glass along the separation line, melting the plastic and making it more elastic. At this point it is possible to separate the two parts. On the *548 Lam* the separation takes place thanks to the work table that is able to move away a few millimeters from the bridge. The glass is kept in position on the table by the two air chambers that press it and do not make it slide on the table, as a result of which the two parts are now spaced.

4. Finally the **cutting phase**: a razor blade descends from the machine in this enlarged position, cutting the heated plastic along the entire length of the piece of glass. When cut, the PVB does not fall inside the edges of the glass but is in line with the surface, which results in a clean and aesthetically pleasing cut, as well as preserving all the properties of the glass unaltered.

To allow a perfect cut, not chipped and without smudges caused by the plastic, there are various parameters to be adjusted on the machine, which further complicate the learning. The various phases of the process described above are qualitatively illustrated in the figure 2.6.

As you can see, cutting laminated glass is not a simple task and the possibility of breaking the glass during separation is more than real. In addition, each glass manufacturer creates glass with different chemical characteristics, which can influence the cutting parameters. For all these reasons, the *548 Lam* machine takes into account the needs of all customers for each type of cut and glass used, so the parameters of pressure, speed, heating are fully customizable and adaptable to every need. This versatility generally goes against the simplicity of the algorithm to be used for the purpose of the first part of this thesis, which reminds us to find correlations between the various inputs that create incorrect or non-compliant outputs, within a view of predictive maintenance. About this complexity of manage the data will be discussed in the database paragraph.

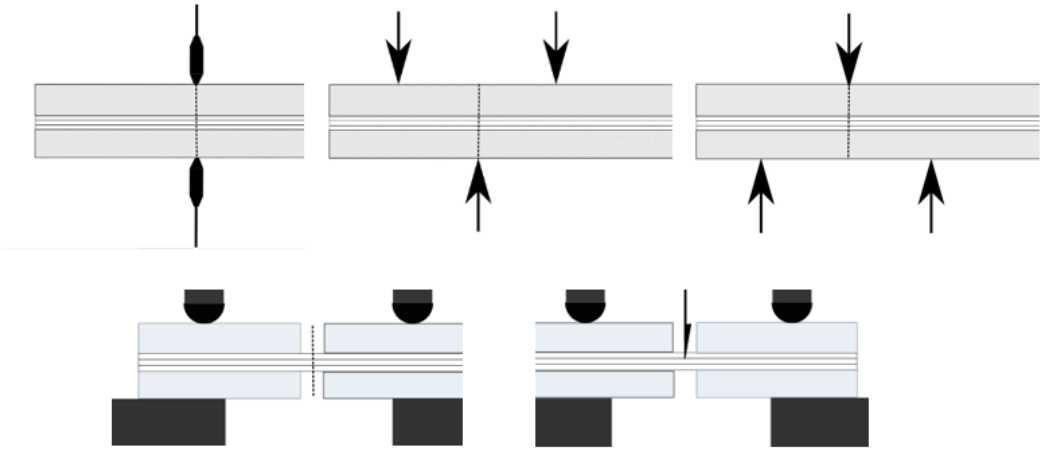

Figure 2.6: Steps for cutting laminated glass: in order Engraving of the surface, Breakout of the two glasses, Heating and Separation, Cut of PVB.

2.3.1 Trim cuts

For the sake of completeness, it should be noted that the *548 Lam* can also make so-called *trim* cuts. Normally, as far as the breakout is concerned, the glass must be placed on the surface of the table or crushed by the inner tube in order to be truncated correctly, i.e. it must have a support on the opposite surface in order to be deformed and to apply the force necessary for the controlled breakage. *Trim* cuts are cuts in which the cutting line is very close to the edge of the glass. In these cases it happens that the surface of the piece cannot rest on the other edge of the cutting table, so for the breakout of these pieces the glass is placed on appropriate tools that the machine extracts when it recognizes that the piece can not rest on the table nor be pressed by the air chambers. Moreover, even the separation and heating phase is different from the classic one explained above, even if the process is the same. Now the heating is done with the part to be cut that does not rest on the table but is suspended in air, once the right temperature is reached the slab is moved until the trim piece does not arrive on the table and can be clamped by a special bar, then there is the separation with the piece now moved from one side, finally a second blade also moved from the center of the bridge cuts the PVB. Usually the trim pieces have to be thrown away because they are waste, in this case a moving part is available on the *548 Lam* that opens and throws away the waste. In figure 2.7 is possible to see the difference of a

trim cut with respect a normal cut.

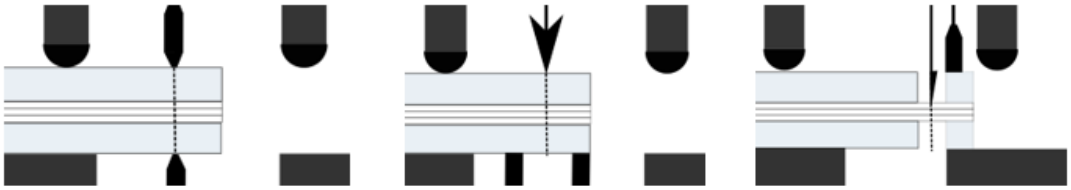

Figure 2.7: Steps for cutting a trim piece: in order Engraving of the surface, Breakout of the glasses, Separation and Cut of PVB.

I wanted to expose also this particular cut in order to make understand even more the complexity of the machine with which we have to deal, obviously this complexity also affects the mass and the diversity of data sent in the database that will be analyzed in the next chapter. Moreover, it should be clear what is meant from now on when we talk about trim cut.

2.4 The machine control software

The *548 Lam* is managed thanks to the apposite integrated computer where there are installed two main softwares: the first one is a high level graphic interface for selection, managing, tune parameters and interact with the user. This is the only part that the worker can see. The second part of the software installed is a low level one. Its scope is to manage the communications (input and output) between the hardware parts of the machine and the interface, where the input are selected. In order to manage the machine it is needed a real time system, so a processor of the pc is dedicated for this task and work in real time like an embedded system. This kind of software is essential to guarantee hard time constraints and to not miss any deadline of the machine tasks. Instead, the interface works on a normal Windows based OS. Unlike the real time software, the interface can be directly managed by the user and its main purpose is to simplify the use and regulation of the machine to anyone who acquires a minimum of familiarity with the various buttons and keys.

All settings chosen by the user through this program are then sent to the low-level software which translates and sends them to the machine. This correspondence is obviously valid also on the contrary direction, in fact when a step of working is finished or there have been some failures,

this software sends to the interface all the useful information to represent the output of the machine, which will be immediately represented to the user and then sent to the database.

Of the two softwares will now be analyzed in more detail the graphical interface so as to better clarify how the machine works and for what purposes it was designed. In addition, this high-level software is also used for recording and sending machine data in the online database.

2.4.1 Interface management

For the purpose of this thesis it is not necessary to examine the type of communication between the software and hardware, so the part to be analyzed is related to the interface. When the machine is turned on, it appears the main screen, which consists of four tabs:

Editor : In the editor you can select or draw cutting patterns (also called *optimizations*). Cutting patterns are representations of properly oriented lines separated by previously selected distances. They indicate where and how a standard glass plate will be cut and therefore the lines contain all the various pieces that will be extracted from the main plate. The dimensions of the glass sheets are standardized while the optimization process can be done with any dimension of glass.[1]

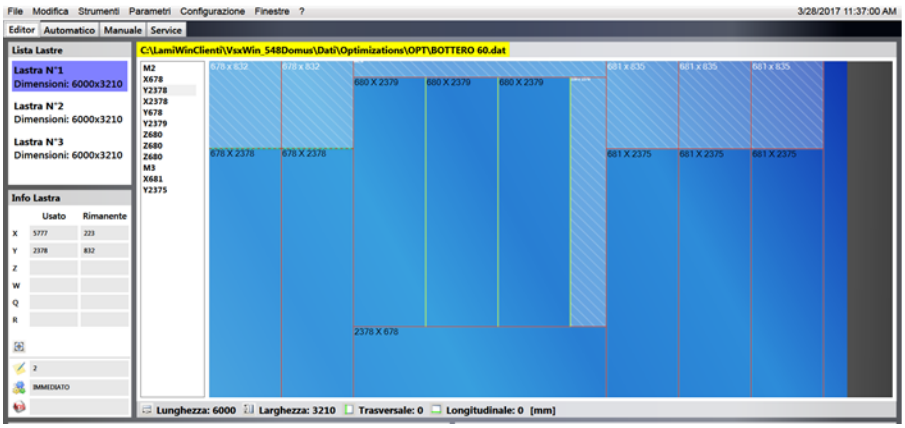

Figure 2.8: Editor menu, here it is possible to select or create a cutting scheme.

[1]The most common sizes of glass sheets are Jumbo (6000x3210mm) and Regular (2250x3210mm although some manufacturers while maintaining fixed the second size change the first in 2550mm or 2000mm)

Automatic : The cutting pattern set in the editor menu is loaded here. Before operating, you can navigate through the cutting pattern and view all the steps of the process that will then be performed physically. In detail you can see how the machine will perform the cuts and in what sequence, which pieces will be moved, discarded, etc. . . all in a clear and illustrative 3D graphic interface. You can also change the type of glass to be used and the parameters set through the special bar of the type of glass used where you can select some options:

- Thickness of the upper glass plate
- Lower glass plate thickness
- PVB thickness
- Engraving head pressure
- Upper truncation pressure
- Lower truncation pressure
- . . .

By pressing the cycle start button, the interface and the machine will work synchronously: the graphics inside the software follow and reproduce exactly the steps being executed. There are some operations (such as diagonal cuts) where the piece in question is shown in yellow, in which case the machine enters semi-automatic mode. This mode puts the machine in stand-by mode, waiting for an operator to perform a manual action that the machine is unable to perform due to mechanical limitations. in this situation the operation to be performed is simple and fast (for example placing a bar on the cutting table to allow positioning before diagonal cut).

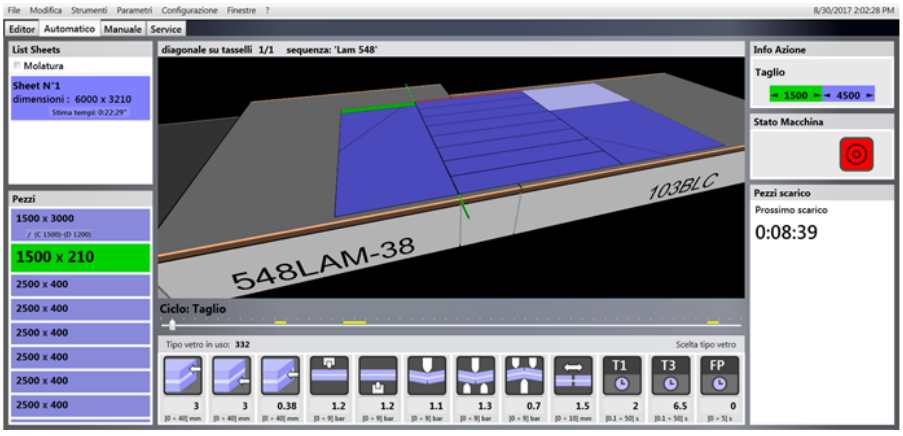

Figure 2.9: Automatic menu. It is possible to see all the phase of the cutting sequence in a clear 3D representation.

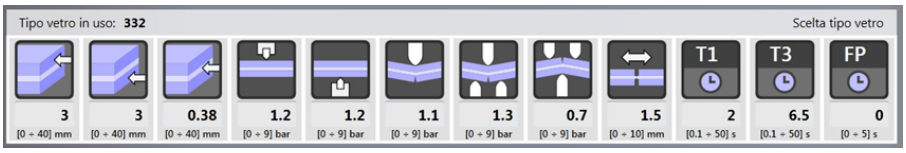

Figure 2.10: Type of glass selection and tuning: some of the parameters related with the type of glass.

Manual : The manual mode, unlike the automatic mode, allows you to make cuts on pieces of glass not set with a cutting pattern. The workpiece must be correctly positioned on the work table. The machine is however able to automatically perform some operations such as understanding the length of the piece to be cut and make the cut. The manual mode can therefore be defined as composed of small automatic cycles that are fast and pre-established. In particular, the following operations can be carried out in this mode:

- Positioning and cutting
- Only cutting
- Only diagonal cutting

Differently from the automatic mode here it is always needed an operator to supervise the operation and interact with the work. Even in this case you can select the type of glass and the parameters to be used accordingly. If any of the settings are changed, an event will be recorded containing information about the new glass type and selected parameters. This event will then be sent to the database. To anticipate what will be said in the chapter on the database, this is an event called *TypeOfGlass* and like all other types of glass is identifiable by a *CodeEvent*. Other events with other *CodeEvent* are sent to the database at different times and in different ways, and obviously contain other types of information.

Service :Allows you to access the screen concerning the controls of the manual movements of the machine. This menu is not very important for the purpose of this thesis.

Summarizing, the machine can work in automatic or manual mode. In automatic mode it works when starting from a whole plate to be cut according to a cutting scheme defined in the editor menu, and it is possible that for some cutting schemes there are operations to be carried out in semi-automatic mode in which the operator must help the correct positioning of the piece of glass that the machine cannot mechanically carry out. In manual mode, instead, cuts are made on single pieces of glass that have not been previously set or calculated. Currently, in manual mode,

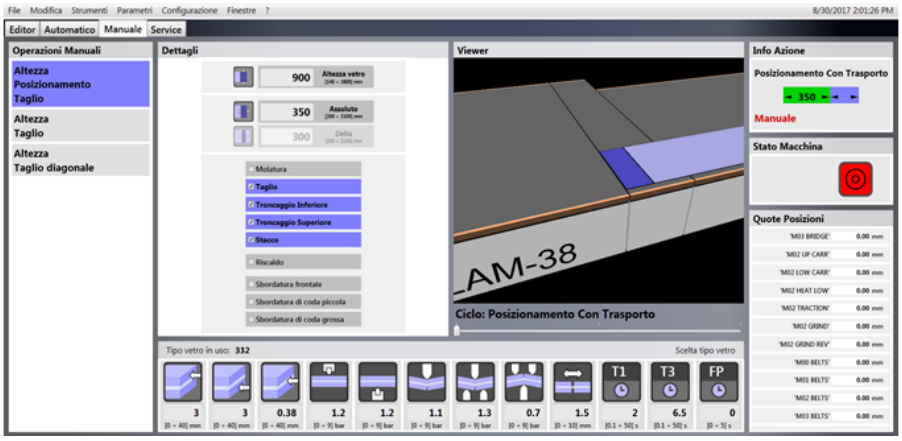

Figure 2.11: Manual menu: on the left there are the three operations a operator can do in this mode. Also here it is possible to configure the type of glass.

three actions are available, all concerning the cut, which is always carried out automatically.

For the sake of completeness, mention is made of the fact that each cut can be broken down into a series of operations as seen in the paragraph 2.3 about the process, some of these operations can be skipped or avoided, all this information is recorded anyway (e.g. it is possible to cut without heating the piece with the electrical resistance, it is only possible to cut the glass, it is possible to cut the glass but not to separate the two parts from the PVB, etc..).

The purpose of the second part of the thesis is to predict the entire time of a machining cycle when a certain optimization is selected, this time changes according to the number of cuts, transports, rotations and in general according to the number of step to be made in the plate. In addition, parameters such as glass thickness, type, selected pressures,... influence the processing times. After this introduction to the machinary it is clear why a machinale learning solution can be hopefully a good alternative to solve also the time prevision problem.

Chapter 3

Database and Data Collection

The company Bottero in recent years has decided to create a database to record the data produced by their machinery. The intent for which the database was born was to make production statistics and keep under control the machinery installed by the various customers also with a view to an easier detection of errors and consequent easier maintenance.The database collects data on various types of machinery produced by the company and installed around the world, not only from the *584 Lam*. So there are many types of machines and many 'copies' of the same machine scattered across the various continents that send their data on the database.

3.1 Structure of the database

In detail, the data is saved on an online AWS database. Each type of data sent by the various machines installed by Bottero customers around the world goes to populate and enrich a single table called *TEvents*. This table collects all types of data sent by various machines, so here inside there are many different types of data, and to use the information inside, it will need to extract the features and classify the data. *TEvents* has a well defined and formatted structure, so each event that populates it has a set of information divided by columns. There are lot of columns for each type of event, although they are often not all used for each event.They are enumerated in table 3.1. You will then have many *null* values that must be taken into account in the preprocessing phase of the data. The data contained in the various columns change meaning according to the recorded event. Each event belongs to a certain well-defined category.

First, however, it is good to define the generic structure of the large *TEvents* table. As I said it is composed by lot ofcolumns. Some of them have fixed meaning for each type of recorded event, others instead change of meaning according to the event. The columns with fixed meaning are: *ID, CodeEvent, PLCIP, DateTime, DateTimePLC, EventData1, EventDataA*.

ColumnName	DataType	Fixed/Variable Meaning	Description
ID	bigint	fixed	index of the data
CodeEvent	int	fixed	event code among 11 possibilities
PLCIP	varchar(50)	fixed	local IP of the machine
DateTime	datetime2	fixed	Date of the registered event
DateTimePLC	datetime2	fixed	as above
EventData1	int	fixed	number of the event from session start
EventData2	int	variable	
EventData3	int	variable	
EventData4	int	variable	
EventData5	int	variable	
EventData6	int	variable	
EventData7	int	variable	
EventData8	int	variable	
EventDataA	varchar(max)	fixed	machine ID
EventDataB	varchar(max)	variable	
EventDataC	varchar(max)	variable	
EventDataD	varchar(max)	variable	
EventDataE	varchar(max)	variable	
EventDataF	varchar(max)	variable	
EventDataG	varchar(max)	variable	
EventDataH	varchar(max)	variable	

Table 3.1: Structure of TEvent table on the Database. At the right side of the column name there is the type of the data contained. All the informations respect this format when arrive at the database. Each event create a single line of this table containing informations in this format.

The population in the TEvents table changes according to the type of event recorded. In other words, in order to extract the data properly, the rows of TEvents must be 'read' appropriately based on the event code. The possible events sent by the *584 Lam* machine and contained in the *CodeEvent* feature are:

- *Session*
- *MachineState*
- *Step*
- *Cut*
- *Glass*

- *Piece*
- *TypeOfGlass* (deprecated)
- *TypeOfGlassBasicParameters*
- *TypeOfGlassAllParameters*
- *TypeOfGlassOnlyDependencies* (deprecated)
- *MachineError*

In the next sub-chapters there will be analyzed in detail all these events, with the meaning to attribute to each variable column defined above, that, as said, vary according to the event type. It is possible to find in *TEvents* different types of event codes, these will be ignored because generated from other machines different from *584 Lam*.

3.1.1 Session event

A session event is generated whenever a new *session* of the machine starts, i.e. when the 548 Lam software is turned on. This event is therefore logically recorded in a much smaller number than the events of type cut or step for example, because usually the machine is turned on at the beginning of the work day even if it is possible that the machines remain turned on several consecutive days or that they are restarted many times throughout the day. In any case, the number of session events reaching the database remains rather limited because this operation is not frequent compared to the number of operations to be performed in each session.

In general and event of type *session* contains informations about the software installed on the machine plus the *datetime* of the start of the session.

3.1.2 MachineState event

An event of type *MachineState* is triggered each time a machine change state. The columns in *TEvents* contain now the information about the machine state, the possible values can be:

- 0: Off
- 1: Ready
- 2: Busy
- 3: Service

. Many columns are not used for this event.

3.1.3 Step event

This is one of the most important events generated by the machine. When an event code is type step the *TEvents* columns collect information about the type of step performed by the machine. The machine can perform many different types of steps including positioning, cutting, diagonal cutting, rotation, etc ... (Others have already been mentioned in the chapter 2.2). This event contains both step input information (selected parameters) and machine output information (step execution time, if the step has ended badly, ...). In detail, the *TEvents* columns assume the following meaning when the *CodeEvent* is related with a step event:

- *EventData2*: Number of the glass plate within the optimization. Normally this number is in the order of tens and grows by one unit each time a new plate starts to be cut in the same optimization.
- *EventData3*: Number of steps in the glass plate: remember that each plate is associated with an optimization, so before the physical start of operations the software already associates at each operation a step number inside the glass plate.
- *EventData4*: It is a decimal number that represents a mask containing a series of operations. it is basically a way of compressing binary information into a single decimal number. The values that this number can assume are between 0 and 7, i.e. it can be represented in binary on 3 bits. Each bit is a boolean key that contain the following information[1]:
  - Bit 0 (LSB):If the step is in automatic mode.
  - Bit 1: If the step has to be performed in semiautomatic mode (chapter 2.4.1)
  - Bit 2 (MSB): if the step is bad terminated (outcome result)
- *EventData5*: Boolean key that indicates if for this glass plat the grinding in on.
- *EventData6*: Measured time for executing the step in *ms* (output result).
- *EventDataB*: String that indicate the type of the step. Among all the most important step is the cut that will be analyzed deeper in this thesis.

[1]LSB stay for Least Significant Bit, it is the first bit starting from the right, MSB stay for Most Significant Bit, it is the last bit starting from the right

3.1.4 Cut event

Whenever there is a *Step* event associated **only** with a step type *StepVsxTaglio5X8* or *StepVsxTaglioDiagonale5X8* , a *Cut* event is immediately sent containing additional informations. In practice this type of event is to be read as an extension of the *Step* event that introduces additional informations when the step is a *StepVsxTaglio5X8* or *StepVsxTaglioDiagonale5X8*.

As for the previous events, the various meanings of the *TEvents* table are now indicated when a *Cut* event is recorded:

- *EventData2*: It indicates the length of the cut to be made calculated according to the optimization and the selected cutting pattern.
- *EventData3*: Indicates whether cutting is in manual mode (boolean key).
- *EventData4*: It is a mask like the *Step* event. it is represented by a decimal number that, if converted to binary, indicates various boolean information through the bits of the binary number. This mask is composed of 8 bits, so the corresponding decimal number varies from 0 to 255. In detail:

  Bit 0: if the piece has to be grinded
  Bit 1: if the piece has to be cut
  Bit 2: if there will be upper truncation
  Bit 3: if there will be lower truncation
  Bit 4: if the piece will be heated during separation
  Bit 5: if the piece will be detached
  Bit 6: if the piece has a diagonal cut to be performed
  Bit 7: if the process is bad terminated (output result)
- *EventData5*:This number represents the measured time to perform the cut (output result)

3.1.5 Glass event

A Glass event is generated when the machine starts to process a new sheet of glass in the optimization process. It is therefore an event that reaches the database only when the machine is working in automatic mode, since in manual mode the glass sheets are not processed but only pieces are cut. For this reason, this event is to be correlated only with steps that take place in automatic mode and not in manual mode. Theinformation contained are related with the optimization glass sheet number from the beginning of the otimization selected, the glass sheet number since the

session started, length of the glass sheet, high of the glass sheet, total length of the cuts in the actual glass sheet, number of pieces to obtained from the glass sheet, name of the optimization.

3.1.6 Piece event

This event provides information on the machining operations to be carried out on a given piece of the plate. Every time the process of machining a new workpiece from a complete or partially machined plate starts, a *Piece* event populates the database. In this case the columns meaning are related with number of the piece in the in the glass sheet,if the piece is worked in automatic mode, length of the piece, high of the piece.

3.1.7 TypeOfGlassBasicParameters event

This event populates the *TEvents* table when the user selects a new glass type from the software or modifies its parameters. The parameters of glass type have already been partially discussed in the section 2.4.1. it is important to note that the machine is able to adjust pressures, speed, forces thanks to the modification of the appropriate commands from the interface. This makes the machine adaptable to any type of glass, not only of different thickness but also of different chemical composition. The columns that go to popular *TEvents* for this type contain information about top plate thickness , PVB thickness , maximum and minimum pressure that can be exerted by the upper head during the engraving phase, maximum and minimum pressure for truncation.

3.1.8 MachineError event

This event is generated when an error occurs during a machine working step. It contains detailed information associated with the type of action that caused the error, its structure in the database is variable and articulated to analyzed. The columns of *TEvents* used are few but not always contain all the information, the content varies greatly for each type of error. The information here inside are related with standard error codes, generic codes valid for multiple errors. In general there is lot of variance in the structure of the error that will be, when needed, analyzed case by case. A possible thing to do is to list all the possible error that can occur. The table 3.2 enumerates these possible errors.

Description	Message Error
Vacuum missing	MSG-ERR-SICUREZZA-VUOTO
Glass search failed	MSG-ERR-MANCA-VETRO
Wheel worn - replace	MSG-ERR-MOLA-ESAURITA
Cycle interrupted by emergency	MSG-ERR-EMERG-BREAK
Bewilderment bridge motor excessive	MSG-ERR-STOP-MOTOR-PHASE
Invalid position for cut	MSG-ERR-POSIZ-ATTESTA
File not found	MSG-ERR-OPENFILE
Servo error	MSG-ERR-SERVO
Servo warning	MSG-ERR-SERVOWARN
Input waiting timeout	MSG-ERR-IN-TOUT
Timeout zeroing	MSG-ERR-AZZERAMENTO
Drive not ok	MSG-ERR-AZZ-WAIT-AZ-OK
Slow down not found	MSG-ERR-AZZ-ACCEL
Stop not out	MSG-ERR-AZZ-WAIT-EXIT-STOP
Stop not found	MSG-ERR-AZZ-WAIT-STOP
Error photocell1	MSG-ERR-FTC1
Error photocell2	MSG-ERR-FTC2
Error upper truncate cycle	MSG-ERR-CICLOTRONCS
Error attesting cycle	MSG-ERR-CICLOATT
Error air presence	MSG-ERR-AIR-PRESENCE
Error reading thickness	MSG-ERR-SPESS
Error cut security	MSG-ERR-SAFETYCUT
Error discharge security	MSG-ERR-SAFETYUNLOAD
Glass load still to execute	MSG-ERR-LOAD-TODO
Glass load in act	MSG-ERR-LOAD-IN-CORSO
Sheet pusher extractable only with high table	MSG-ERR-PUSH-ON-LOW
No input	MSG-ERR-NO-INPUT
Presence Input	MSG-ERR-INPUT
Glass On	MSG-ERR-GLASS-ON
Time out	MSG-ERR-TIMEOUT
Time out transport glass	MSG-ERR-TIMEOUT-TRANSPORT-GLASS
Error transport glass	MSG-ERR-TRANSPORT-GLASS
Grinding wheel operations in progress	MSG-ERR-OPER-ON-GRIND-IN-CORSO
Operations on the clamping cups in progress	MSG-ERR-OPER–ON-CLAMPING-IN-CORSO
Trim presence	MSG-ERR-TRIMONPRESSOR
Errore cut cycle	MSG-ERR-CICLOCUT
Errore high length	MSG-ERR-ALTEZZA
Error piece length	MSG-ERR-LUNGHEZZA
Trim Not Falled in Trim box	MSG-ERR-TRIM-NOT-FALLED
Pezzo fuori squadro	MSG-ERR-FUORI-SQUADRO
Error rotation cycle 6000	MSG-ERR-CICLOROT6
Data corrupted on some EtherCAT slave.	MSG-ERR-MESSAGES-LOST-BY-SLAVE
Error safety tilt	MSG-ERR-SAFETYTILT
Glass under thickness measure	MSG-ERR-TRANSPORT-GLASS-FTC-THICK
Timeout Blade Lowering	MSG-ERR-TIMEOUT-BLADE-LOW
Blade broken	MSG-ERR-BLADE-NOT-OK
Blade Trim broken	MSG-ERR-BLADETRIM-NOT-OK
Trim Box Full	MSG-ERR-TRIM-BOX-FULL
Glass on M03, but not on FTC upper carriage	MSG-ERR-TRANSPORT-GLASS-FTC-M03
Axis setEnabled failed. Axis	MSG-ERR-ENABLE-AXIS

Table 3.2: List of all possible errors for *548 Lam.*

3.1.9 Other events

Events 100007, 100009, 100010 are not analyzed. However, a brief description and justification of this decision is given:

- The event 100007 is no longer used, it was the old format of selection type glass, now replaced by the event 100008.
- The 100009 event contains more precise details about the glass type and is sent together and in extension to the 100008 event. Inside, however, the data is not formatted in columns but a single feature contains a long string with all the information about the glass type, much more than those contained in the event 100008.
- The 100010 event is being studied and is not yet implemented, always on the glass type.

3.2 How and when the data is sent

A number of operations are performed before data is sent to the database. Usually the parameters and the configurations on the use of the machine are chosen before the operation to be carried out through the graphic interface. These parameters are then sent to the real time software that controls the machine, which translates them into a language suitable for communication with the hardware mounted on the *548 Lam* machine. This low-level software controls the process that the machine is running, providing input and obtaining output results. When a certain step or phase ends, the low level software communicates to the interface the outcome of the operation, returning both the inputs previously supplied and, in some cases, the outcome of the operation carried out. At this point the execution of the program continues showing the user the next step that the machine starts to execute, moreover the log files of the operation are recorded and some of the data of return from the low level software are selected and saved in a sort of temporary local backup. When the machine is not too busy, this backup sends the selected and recorded data to the cloud database, going to populate the *TEvents* table discussed in the previous paragraph. The information contained in *TEvents* are therefore only a part of those that the machine uses during its normal operation, and it is possible, by changing the criteria for selecting information to be sent to the database, to change the type of data that reaches *TEvents*.

However, not all events need to be output from the machine and can be sent before the machine physically performs the operation. These are selection data, for example the parameters of the glass type and of the optimizations are saved as events in the database every time the

configurations of the interface changed. It is therefore possible that there are more than one configuration change and therefore various events of the same type in succession during a period in which the machine is not actually operating. This happens for glass events, when changed it is possible that various events related to the various changes reach the database, it is therefore important to select only the last recorded event of glass type change before the machine starts operating, so as to associate the last glass type chosen to the steps and cuts actually made by the machine.

This last reflection will be of extreme importance when, later on in this thesis, we will proceed to combine various types of events together (for example, the *Cuts* events with the *TypeOfGlass* ones, the *Step* with the *Glass* in the optimization, etc...). An accurate and careful unification of the events will lead to the creation of new extended but reliable features for the events of type *Cut* and *Step* that contain the output of interest. Reliability is vital to have concrete and correct results from the machine learning algorithms that will be used to try to predict errors and increase the quality of the output of the machine in terms of correctly cut pieces or successfully performed steps. Also expanding the amount of features to be used with machine learning algorithms can be of great importance if you realize you have problems related to high *bias*. These technicalities will be discussed in detail in the chapter dedicated to the choice of the most suitable algorithms for this work.

3.3 Raw Data Manipulation on the Database for Time Prevision

TEvents is the only table that continuously increases the size of the AWS database. It is still possible to manage and edit data directly online through the Microsoft SQL Server database manager. Through this software you can manipulate the database by creating new tables or views getting the data from *TEvents* filtering them appropriately. In fact the subtables generated by *TEvents* in the AWS server are many, all created for statistical purposes. Among these there are 2 that will be the starting point for the time prevision part, already developed by Bottero. The second part of the thesis will take in input these two tables and after a small pre-processing the data will be well formatted to estimate the processing time for each step.

These two tables are:

- view_StepDetail

- TLamiWinDataToRtx548

The first one is an extension of the Step event to which other information of other events like Session, Glass, TypeOfGlass have been associated so that the table contains a lot of information related to each step taken by the machine. Having a lot of information for each step it is possible to estimate the working time of that operation and finally obtain the total working time of the plate by adding all the steps belonging to a glass plate.

The second table contains additional information for each step. A column is a pointer to the step ID so that the two tables can be merged to create an extended one with even more information, so that the time prediction is more precise.

These two tables were created by Bottero explicitly in order to be functional to the prediction of machining times. To create them Bottero worked, as anticipated, on SQL Server directly connected to the database AWS. Starting from *TEvents* the desired data was searched and associated in the two tables. The programming on SQL Server is done through the T-SQL language very powerful. The two tables will then be extracted and imported in python development environment for faster management and for the application of machine learning algorithms. Of this work I will discuss in chapter related with time prevision.

Chapter 4

Machine Learning: Main Supervised Characteristics and Algorithms

A formal definition of what machine learning is was given by Tom. M. Mitchell and turns out to be highly rated and appreciated:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

It is clear from this definition what is meant by *learning*: a program is said to learn if there is an improvement in performance after completing a task, i.e. with experience. Based on this experience of the program, the objective of machine learning is to successfully complete new tasks that it has never faced. Thanks to many examples collected (and therefore known), the machine that has the task of producing accurate forecasts based on criteria detected independently during the training phase is instructed. From what has been said we can identify the various steps that are necessary to successfully develop a classical machine learning algorithm. These phases are:

- Learning
- Test and Validation
- Prediction

To these it is necessary to add the *pre-processing* phase of the data to provide the algorithm with correct, significant and well-formatted examples, so as to optimize the learning phase and obtain better predictions. The first two phases are iterative until the algorithm is validated. Iterative

means that parameters, features, algorithms are often changed before being satisfied with the result. These iterations are due to the resolution of problems intrinsic to machine learning projects that are related to overfitting, underfitting, bias, variance, size and characteristics of the training and test set, type and complexity of the hypothesis, features selection. In the next paragraphs these concepts will be extended and defined precisely because they will be an integral part of this thesis, in fact we will discuss them again when we will choose the algorithms to be adopted for this work.

Now it is good to define the different types of existing machine learning that can be divided into 4 groups:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Recommender Systems

The most important for this work is the first type. *Supervised Learning* works with examples providing input and relative output, a part of these examples are known during the learning phase. The objective is to extract a general rule that associates the inputs with the correct outputs, when this rule has been found the algorithm is able to use only the inputs so as to be useful for predicting the output. The goal is to be able to trust these predictions with some confidence, to measure the degree of "confidence" of the model there is the testing and validation phase that defines measures regarding accuracy, precision, recall, F1 score, etc...

The other types of machine learning are not important for the purpose of this work. Only *Unsupervised Learning* deserves a mention, in which unlike the previous one the outputs are not provided but are only the inputs. These algorithms have the task of finding correlations between the inputs provided and grouping them, doing what in jargon is called *clustering*.

4.1 Supervised Learning

To better understand this type of learning on which this thesis is based, the typical structure of this type will now be described. In addition, common and known problems will be discussed, together with some evaluation methods and the most used supervised learning algorithms.

4.1.1 General Structure

To be able to use these types of algorithms you must have a so-called training set of examples. Generally, the larger this set is, the better the final result will be, but the slower the training process will be. Sometimes having a huge set of examples is not enough to improve the results of the algorithm, but surely the result will not get worse as the examples grow. The set of examples must always be representative of the problem that you want to solve, that is, the examples that you want to predict later must have similar characteristics to those used during the testing and training phase.

Normally the set of examples is divided into two groups, one for training (the algorithm searches for correlations between input and output) and the other for testing and validation (the trained model is provided only the inputs of the test set, so that you can compare the predictions resulting from the correlations found previously in the training set with the true outputs of the test set, known but not provided). It is common practice to divide the two sets into percentages 70-30 for training and testing, but a variation of $\pm 15\%$ is common. Another method of model validation is to divide the set of examples into three sub-sets instead of two: *training, cross validation and test sets*. The new *cross validation set* works as a temporary test set that indicates what to change when adjusting the algorithm parameters. We use a percentage division of the set of examples in the ratios 70-15-15 or 60-20-20. These percentages are not fixed and generally the tests and cross validation sets reduce the fraction of the total database if the number of examples is very large. In fact, for databases with many entries (millions) it is not necessary to have large test sets because even a small percentage contains all the types of examples needed and are therefore representative of the case study.

While the testing phase is simple and requires no effort to be understood, the training phase is much more complex because it is based on mathematical intuitions. Training is in fact an iterative process of minimizing the error of prediction. Generally this error is calculated on the basis of the difference between the prediction and the true output. With the number of iterations this error tends to decrease asymptotically. Technically the error that is calculated is the error of *fitting* of the data by a *hypothesis function*. Before continuing, it is good to define what is meant by these two terms. The hypothesis function is the mathematical function that predicts the result given the inputs and set the parameters within the function itself. This function can be more or less complex, linear or polynomial, exponential, logarithmic etc... A given example defined by certain features (input) produces an output well defined by the

function and dependent on the chosen parameters. Therefore the hypothesis function *weighs* the various input features to produce output. The weights are just the parameters that therefore become the object to be defined during the learning phase. Each algorithm has a different hypothesis function and therefore different parameters will also be obtained. The value of the weights (the parameters) define the so called *fitting* of the data. The fitting can be defined as finding the multidimensional line or surface that best approximates the examples during training. The choice of weights is defined by the *cost function*, i.e. by a function of minimizing the error between the fitting surface defined by the parameters in the features space and the real values of the output. If this fitting is very precise (i.e. each example perfectly belong at the surface) the algorithm risk running into the problem of overfitting, if it is too rough it risk the opposite problem of underfitting. These two problems will be discussed shortly.

A simple representation of what has just been explained is visible in figure 4.3 where you can observe the fitting line for a linear regression problem, that in the first case represents an underfit condition and in the third case an overfit, while the second is an adequate fit of the data. In figure 4.2 it is instead possible to visualize a graphic representation of a possible and simple cost function that depends only from two parameters. When its value is minimum, there is the best adaptation of the training data with the prediction model, but there is a risk of overfitting.

Only for example a Linear Regression Hypothesis Function (regression problems) has this shape:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 * x_1 + \vartheta_2 * x_2 + \cdots + \vartheta_n * x_n = \vartheta^T X$$

Where the coefficients ϑ are the parameters to be estimate and the features x are the input informations at the algorithm. The cost function to minimize to find the ϑ parameters is instead:

$$J(\vartheta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\vartheta(x^{(i)}) - y^{(i)})^2$$

Finally the iterative process to compute ϑ parameters is of this form:

$$\vartheta_j = \vartheta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\vartheta(x^{(i)}) - y^{(i)}) x_j^{(i)}]$$

As you can see the conversion depends on a so-called *learning rate* α, the tuning of this parameter have some effect on the velocity of the conversion and in its accuracy, in fact if it is too little the conversion will

be precise but very slow and the number of iteration required to converge at the minimum of the cost function will be huge. On the contrary, a big value of α may create problems of oscillations in the convergence or even divergence. So it is crucial to determine it well.

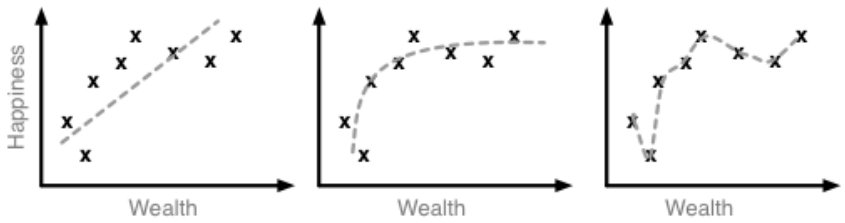

Figure 4.1: Different fit of the data set for a regression problem: an underfitting, a good fit and an overfitting. The different behaviors are due to different choice of parameters in the hypothesis function.

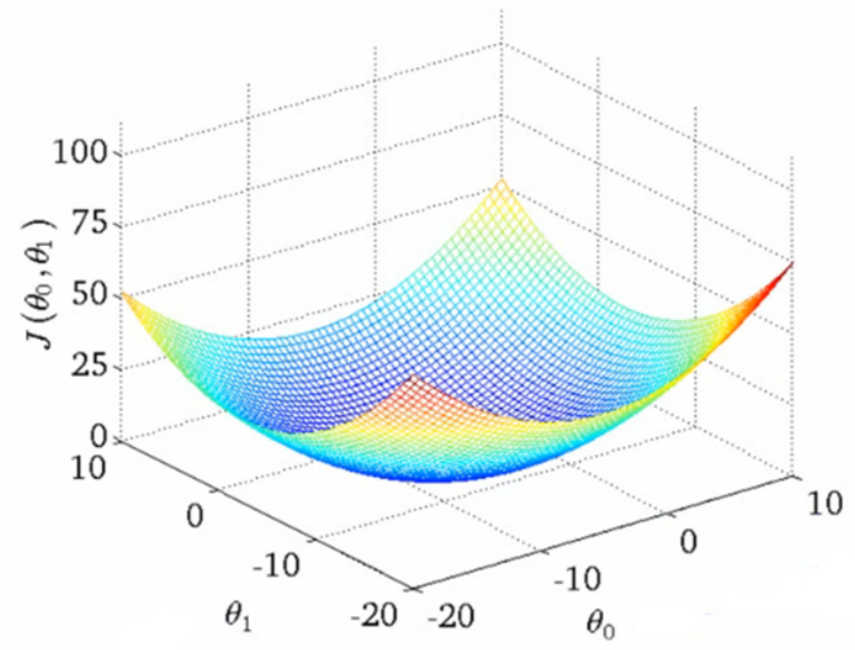

Figure 4.2: A simple representation of a cost function: there is at least one minimum but normally it is possible to find a local minimum trying to minimize the cost function.

Things can be much more complex than what is described with the increase of the number of parameters, of the form of the cost function and of that of the hypothesis function, of the variance and of the average of the data.

4.1.2 Classification and Regression

Before are cited terms such a regression and classification. These are the two categories of problems in witch supervised learning algorithms are divided:

Regression problems : they are problems with continuous output in the domain of real or integer numbers or in any case in an infinite range of possibilities. An example of a regression problem is the prediction of the price of a house based on the number of rooms, square meters,...

Classification problems : the output of the problem is in this case belonging to a small circle of possibilities. This set can be composed by two alternatives (0 or 1) or by a integer number more or less wide. In any case, the size of the integer is known a priori. Some examples of classification problems are the diagnosis of a disease, predicting the gender of a child based on the characteristics of the parents,...

It is easy to guess that for the purpose of this thesis it is necessary to use supervised algorithms for both the parts of error prediction and time prevision, because the output that we will analyze and that we want to predict for the *548 Lam* machine is labeed with a known output. In details classificatioin algorithms are to be used in the first case to understand whether a glass cut of the *548 Lam* machine ends well or badly, while in the second case we need a continuous range of time necessary to make a certain step, this would belong to regression problems. In any case, both these models need to be trained with supervised techniques, i.e. with a complete training set of known output.

4.1.3 Underfitting and Overfitting

In this and in the following sub-section the already mentioned concept of fitting is extended.

Overfitting generally appears in models that train data too well, that is, when a model learns the details and even the noises of the data. These have a negative and wrong impact during the test phase because the model takes into account the fluctuations that normally are present in the training set. Noise is learned as part of the model even if it is not. This problem is mostly found in non-linear models, i.e. as complexity increases, because the fit curve can take on more complex forms and there is a lot of flexibility in fitting. Therefore, some techniques have been developed to limit that the model learns too much detail from the features and thus avoid overfitting.

The problem of underfitting, on the other hand, refers to poorly approximated problems that cannot predict new examples. The function and calculated parameters are not suitable for the chosen model and performance is poor. Underfitting is generally simpler to resolve than overfitting because it is easy to identify with a good choice of performance metrics.

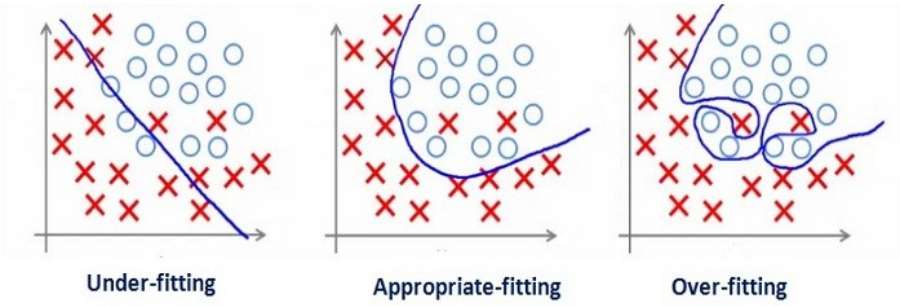

Figure 4.3: Overfitting and underfitting for a simple classification problem: the circles and the cross identify two categories of output, they are correlated with the values of the features in the axis.

Ideally, you should select a good model that is precise but does not include noise during the training phase. The problem is therefore a tradeoff between under and overfitting.

To understand how learning is working with respect to fitting, it is possible to plot the errors of training and test sets according to the complexity of the algorithm. Both start from a big error that initially decrease. The error of the training set continues to decrease incrementing the complexity of the model and tends to converge to zero when there is a perfect fit of the data (including noise), while that of the test set first drops to a minimum value, then if the model is too complicated begins to rise again. When the error of the test set starts to rise again, we are facing with the problem of overfitting. When both errors are large there is instead the problem of underfitting. It is important to note that we are globally interested in the error on the test set because the ultimate goal is to train an algorithm to make predictions on data not yet analyzed. The figure 4.4 is a graphical representation of what just mentioned.

Overfitting is a big problem during a machine learning project. Unlike underfitting, it is difficult to identify because the training model behaves well (too well). There are various methods to limit overfitting including simplifying the model, applying regularization (limiting the importance of certain features) and more.

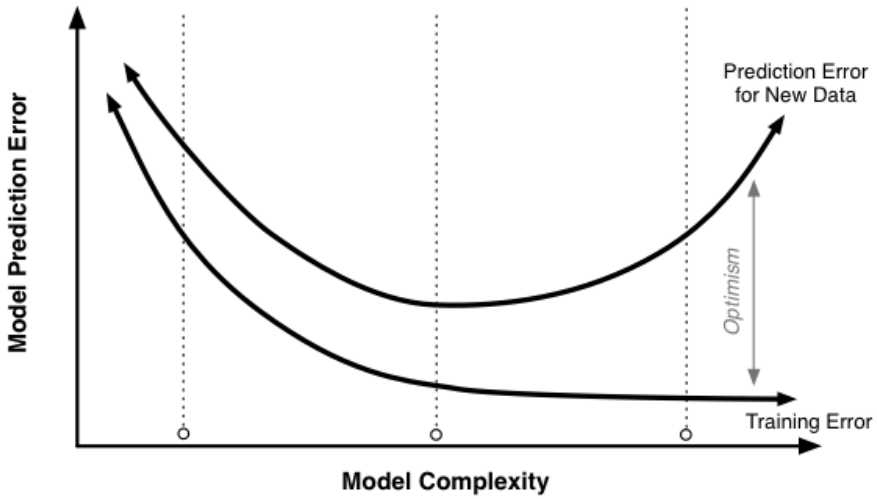

Figure 4.4: Correlation errors/fitting in function of model complexity: for great complexity of the model the test set may suffer of overfitting, while for too easy model it can suffer of underfitting. The correct fitting is a tradeoff of the two.

4.1.4 Bias and Variance

The concept of bias and variance is closely linked to that of fitting, so it is discussed here. Bias and variance are two typical problems of the machine learning algorithms that always arise and therefore must be able to analyze and limit. There is a tradeoff in the ability of a model to minimize bias rather than variance. If you have a lot of data available, an informal definition is as follows:

Bias :can be thought of as the error of the algorithm in predicting the output of the training set from which it was trained and that therefore "knows".

Variance :can instead be thought of as the worsening of prediction between the training set and the test set (it is assumed that the accuracy of the training set is always better than that of the test set).

Using these two definitions is very easy to compare them with the concept of fitting. In fact if the error on the training set is low but I have a big error on the test set there will be an overfitting problem and there will be great variance. On the contrary if the error on the training set is big and is very similar to the error on the test set there will be underfitting and high bias.

Finally it is possible to define a minimum bias not eliminable, in this case the definition of bias changes because it is necessary to take into account this additional part. The unavoidable bias is not erasable by any

kind of algorithm because it depends on the structure of the data. This type of bias is not always easy to recognize, especially in machine learning projects that a human is not able to perform.

There are many method to reduce bias and variance, but as you can see in table 4.1 many of them are in contrast: one reduce a problem but sometimes increase the other.

Reduce Bias	Reduce Variance
Increase the model size	Decrease the model size
Modify input features	Reduce input features
Reduce or eliminate regularization	Add regularization
Modify model architecture	Modify model architecture
Add more training data	Add more training data
Augment complexity of the model	Add early stopping in gradient descent

Table 4.1: Reducing Bias and Variance: some method are common and other are in contrast.

Mathematic Definition

To formalize the definition, it is now proposed the mathematical concept of Bias and Variance related to the world of machine learning. Given a set of m examples each composed of n features x_i and an output y , we can evaluate the performance of the machine learning algorithm through the analysis of the convergence error between training set and real output.The real output y can be described by a function of x plus an error term with zero mean: $y = f(x) + \epsilon$. The error we are interested to compute is the square of the difference between this y and the hypothesis function $\tilde{f}(x)$ that is the predictive function that approximate y:

$$Err(x) = E[(y - \tilde{f}(x))^2] = E[(f(x) + \epsilon - \tilde{f}(x))^2]$$

It is possible to show that the previous formula can be re-written as follow:

$$Err(x) = (E[\tilde{f}(x)] - f(x))^2 + E[(\tilde{f}(x) - E[\tilde{f}(x)])^2] + \epsilon^2$$

where the three separate terms have the following meaning:

$$\begin{cases} Bias^2 = (E[\tilde{f}(x)] - f(x))^2 \\ Variance = E[(\tilde{f}(x) - E[\tilde{f}(x)])^2] \\ Irreducible\,error = \epsilon^2 \end{cases}$$

The important thing to note is that with this definition Bias and Variance depends on $\tilde{f}(x)$ so on the selected model and on the correlated parameters, while ϵ^2 is proper of the system. This means that Bias and Variance are variable parameters that we can modify if selecting an appropriate hypothesis function while the internal error of y is not. This error depends on the goodness of the data and can influence a lot the final result of the machine learning project.

Learning Curves

The learning curves are graphs of performance (y-axis) according to experience or training time (x-axis). The most common in machine learning are curves of loss (cost function) or accuracy with respect to the increase of training examples. In the same graph two curves are usually plotted, one relative to the training set and the other to the test set. In this way it is possible to compare the evolution of the performances with respect to the variation of the experience of the algorithm based on the quantity of examples used during the training phase. In this section we decided to plot the error of the training and test sets, but later in this work you will also see those related to accuracy that are roughly the opposite of those of the error.

The learning curves based on error as a function of the number of examples generally have a decreasing trend with regard to the test set and increasing with regard to the training set. This means that generally the error of the test set decreases with the increase of the number of examples while the error of the training set increases. At best, the two curves asymptotically converge around a low error and their values remain about constant after a number of examples of the training set. In figure 4.5 is shown a learning curve of a model that perform well because the convergence is fast and the final error low.

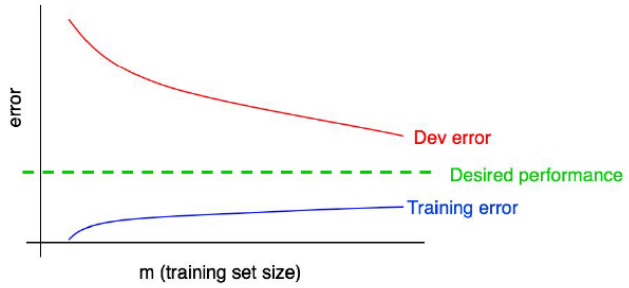

Figure 4.5: Learning curve that seems to perform well: the two curves are converging near the desired value.

If the behavior deviates from that of the figure 4.5 it is possible to identify various problems that can be solved in different ways:

- High Variance
- High Bias
- High Bias and Variance
- Model Performance Mismatch

In order to identify these problems, it is necessary to be able to recognize for your model a desired performance to be displayed on the graph. In the case in which the metrics of evaluation of the learning curve is the error, it is therefore necessary to be able to estimate the error considered acceptable for our machine learning model (e.g. I want my model to be able to fail less than 5% of the output, 5% is the desired error).

Now the possible behaviors of the learning curves will be briefly analyzed:

High Variance

If you are measuring the error a model with high variance will have a big gap between test and training error. The desired performances are still achievable because the error of the training set remains low and therefore increasing the number of examples or using other techniques to limit the variance the two curves will get closer and closer.

Figure 4.6: Learning curve for high variance dataset: the two curves can still converge in the range of desired performance if the model is correctly modified.

High Bias

If there are bias problems, the two curves converge rapidly and at the same asymptote but far from the desired performance. This means that the test set is representative of the dataset but the model has been poorly trained.

High Bias and Variance

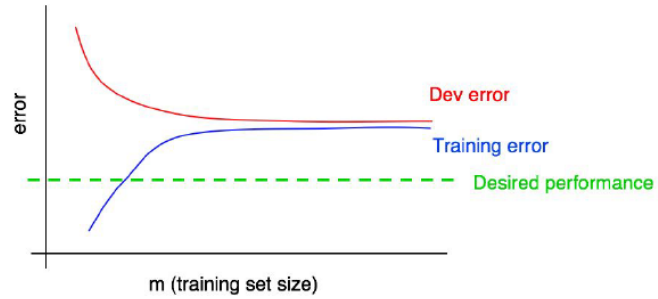

Figure 4.7: Learning curve for high bias: the two curves behave the same and are converged. No more data is useful and we have to modify the model for better performance.

The two curves are quite flat and converged, far from each other and both far from the desired performance. In this case it is necessary to change the model or improve the pre-processing phase. If these methods do not work, there may be a problem of model performance mismatch.

Model Performance Mismatch Learning curves can also be used to diagnose dataset properties and whether the dataset is representative of the problem to be solved. A non-representative dataset does not capture the statistical characteristics of the problem. The term Model Performance Mismatch means that the chosen machine learning algorithm behaves very differently between training set and test set. A small difference is normal because it indicates an inevitable small overfitting but if this discrepancy is too large it compulsory to reduce it to obtain acceptable results. We want to have small discrepancies between training and test set so that we can compare and choose between the various models which is the best. The possible causes of a Model Performance Mismatch are:

Model overfitting: usually is the more frequent cause, solutions were already mentioned.

Quality of the sampling data: it is possible that the training or test set is not representative or that the training set does not cover all possible cases. In this case, increasing a lot the number of examples is very useful to cover all possible cases. To understand if we are facing this problem it is useful to analyze the statistics of the features and look for if there is great variance and standard deviation in the data.

Stochastic nature of algorithms: creates discrepancies in the model score due to a random factor (e.g. algorithm initialization) that affects model accuracy. This problem can be visualized by evaluating the variance of the cross validation set.

Many problems with the Model Performance Mismatch can be avoided by using a more robust test set. To understand if you are using a suitable test set you need to perform an analysis before using it for evaluation. However, this analysis is often complex and time-consuming.

4.1.5 Evaluation Metrics

The problems related to fitting, variance and bias can be identified through evaluation metrics, which, in addition to indicating which are the problems, also make different algorithms comparable. Moreover, the same algorithm can give good results with a certain evaluation metric, and poor results with others. Therefore it is necessary to know all the evaluation techniques in order to fully understand how the model is really behaving.

The most popular evaluation metrics are:

- Accuracy
- Confusion matrix
- Recall and Precisioin
- F1 score
- Mean absolute error
- Mean squared error

A brief discussion about these evaluation metrics are carried on:

Accuracy

Accuracy is the most classic and instinctive method of evaluation based on the ratio between correct predictions and total number of predictions.

$$Accuracy = \frac{Correct\,predictions}{Total\,number\,of\,predictions}$$

A significant problem of this method of evaluation is visible when you have very unbalanced classes in the number of occurrences, this method of evaluation does not give useful information and you risk misunderstandings.

Example: if a machine works well in 99% of the cases and gives an error only 1% of the time, an algorithm that never predicts errors will predict 100% negative cases (usually 0 is attributed to the major value and 1 to the particular case). The accuracy, when calculated, will therefore be 99%, which seems to be a very good value. The problem is that

my algorithm never recognizes a positive case, so it does not work! For this reason, for very unbalanced classes, other methods such as F1 score, Recall and Precision are used.

Confusion matrix

It is an array that completely defines the behavior of a model, used in classification problems. In case of binary classification the confusion matrix correlates the predicted results with the real ones. Its form is shown in figure 4.2. The terms *true* and *false*, *positive* and *negative*, refer to the

	Predicted Negative	Predicted Positive
Real Negative	True Negative	False Positive
Real Positive	False Negative	True Positive

Table 4.2: Confusion matrix for binary classification.

goodness of the forecast after comparing it with the real output in the test set. In detail:

True Positive : positive predicted and positive real value.

True Negative : negative predicted and negative real value

False Positive : positive predicted but negative real value.

False Negative : negative predicted but positive real value.

Positive is usually understood as a minority value to which we tend to assign the value 1 (in other words, the action that happens rarely has a value of 1). You can calculate the previously defined accuracy with the use of the confusion matrix in this way:

$$Accuracy = \frac{True\,positive + True\,negative}{True\,positive + True\,negative + False\,positive + False\,negative}$$
$$= \frac{True\,positive + True\,negative}{Total\,number\,of\,prediction}$$

Recall and Precision

These are two measures based on the values of the confusion matrix. They are especially useful in the case of unbalanced datasets, when the measurement of accuracy is not adequate as described above. The definition of precision and recall is as follows:

$$Precision = \frac{True\,positive}{True\,positive + False\,positive} = \frac{True\,positive}{n.\,predicted\,positive}$$

$$Recall = \frac{True\,positive}{True\,positive + False\,negative} = \frac{True\,positive}{n.\,actual\,positive}$$

These are two measures that are often calculated for the minority class but their definition can be modified to use it also with the majority class.

Accuracy is the reliability of the prediction. When precision tends to one for sure when the algorithm predicts a positive case the real output will also be positive. On the contrary when the precision tends to zero if the algorithm predicts positive the real value will almost certainly be negative. Therefore the more the precision tends to the value one, the higher the reliability is.

The recall instead indicates the amount of positive values found by the algorithm on the total of the real positive values. When the recall tends to one, the algorithm finds almost all the values that are actually positive. If it tends to zero, however, the algorithm finds almost no positive values among the real ones.

There are cases where you have high precision and low recall: in these cases you can trust when the algorithm predicts one, because it will also be a real positive, but there will be many unforeseen positives. On the contrary, there are cases in which you have low precision and high recall: the algorithm predicts many positives that will actually be negatives, but on the number of forecasts the algorithm includes almost all the real positive values. The reliability is low but it can be said to be a particularly conservative case.

F1 Score

Since precision and recall are often in contrast, this value represents the condensation of two metrics into one number. It is often defined in various ways but the most common method is as follows:

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

Where P and R represent precision and recall values. The relation between precision and recall is usually non linear, for this reason is a good idea to use this single performance metric. Sometimes there are situatins in which the goal is to reach a good precision instead of a high recall or vice versa. So the better choice is to apply these metrics case by case.

Mean absolute error

The metric now analyzed is again a generic metric for both classification and regression problems that indicates the difference between the prediction and the real output, in practice it is a measure of the error of the

average prediction for all the examples of the training set. It is defined in the following way:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \tilde{y}_i| \qquad i = 1,2,\ldots,m$$

Mean squared error

Very similar to the previous one but it takes into consideration the square of the difference between true output and forecast, then it averages on the number of examples in the training set. The advantage compared to the previous metric is in the easy use for the search of the gradient because with this shrewdness of the square the calculation of the derivative is simplified a lot. The computational calculation is faster. In addition, the power of two enhances the larger errors making the algorithm more efficient for the main losses. The form is as follows:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \tilde{y}_i)^2 \qquad i = 1,2,\ldots,m$$

4.2 Classification Learning Algorithms

This section describes the main features of the most commonly used machine learning algorithms concerning classification problems. The description will be mostly qualitative, expressing general concepts and useful information to understand the differences between the various algorithms. Where deemed appropriate, some mathematical concepts that are on the basis of the algorithms will be expressed. In the literature exist numerous articles and manuals, please refer to these for further details.

4.2.1 Logistic Regression

It is a simple and well-established classification technique. It takes its cue from *linear regression*, a standard technique for regression problems (output to be predicted in a continuous and infinite domain). It is based on a hypothetical function to make predictions called *Sigmoid function* or *Logistic function*. It is a function with real domain between zero and one and infinite co-domain. Its form is the following:

$$g(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}$$

The Sigmoid function is defined positive, dependent on the matricial

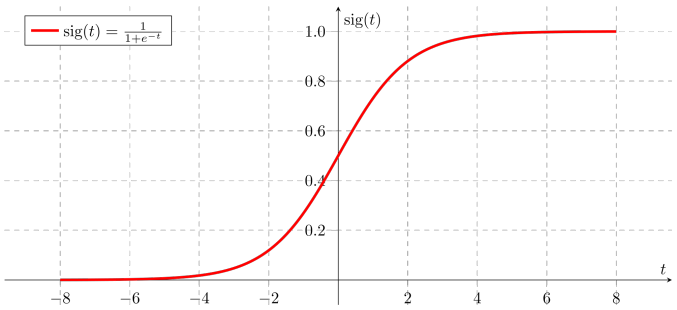

Figure 4.8: Sigmoid function: hypothesis function for logistic regression.

quantity $\theta^T X$, i.e. it is a function of the features X of the examples of the dataset once fixed the parameters θ to optimize. Returns a value between 0 and 1 not a simple binary value. However, being in a problem of classification the algorithm must choose a threshold of separation of the two classes. Usually this threshold is set to the value of 0.5 but can be modified to increase or not the confidence in the prediction obtained. In fact, a number close to 1 indicates that the prediction is more likely to be safe, while a value slightly higher than 0.5 is an indication of uncertainty of the forecast. As the complexity of the $\theta^T X$ function increases, there are more complex decision boundaries and a better fit of the training data, creating however the possible problem of overfitting and variance.

To find the parameters θ a modified cost function J is used compared to the normal minimization of the square of the error between prediction and real output. This is due to the form of the cost function created using the Sigmoid function to make predictions, in fact if you used normally the cost function would become a non-convex function and the simple application of gradient descent algorithms would not find the global minimum but would stop in a local minimum, resulting in poor performance. For these reasons the cost function becomes the following:

$$J(\vartheta) = \frac{-1}{m} \sum_{i=1}^{m} (y^{(i)} \log(h_\vartheta(x^{(i)})) + (1 - y^{(i)}) \log[1 - h_\vartheta(x^{(i)})])$$

Through the derivation of its gradient it is possible to find the global minimum and therefore the appropriate parameters. Once found, the training phase ends and you can use the Sigmoid function to quickly calculate future predictions.

Due to overfitting problems it is possible to modify the cost function to find theta parameters that are less dependent on the details of the X feautures, this technique is called *regularization*.

Logistic regression is not the most powerful and versatile technique but it is certainly well structured and there are various reliable libraries that implement it. It works well in standard machine learning problems where the purpose is well defined. It is very fast if there are no features disconnected from the output or similarities between features (independent features). It is susceptible to overfitting problems and requires well formatted and linearly separable data.

4.2.2 Naive Bayas

Unlike the previous one, this is a probabilistic method based on the Bayes theorem. First of all we calculate the probabilities of obtaining an output 0 or 1, based on the training dataset, then we individually correlate the features with the output, that is according to the value of a feature we calculate the probability that the output is 0 or 1. We repeat this operation for all the features, then we make cross correlations between the probabilities just calculated. In other words we multiply the various probabilities feature by feature obtaining a general value index of the probability of that sequence of features with respect to the output. The more features you have and the more diversity of possibilities you have for each features, the more complex the calculation and cross-referencing of data becomes. The probabilities obtained must then be normalized according to the recurrence of the features and their values.

In mathematical terms we can say that given a problem of classification it is represented by n features X (they must be independent for Bayes property). The model assigns a probabilistic outcome to each C_k for each possible K outcome. If we are in binary classification the outcomes are only two (0 or 1). In formula:

$$P(C_k|x_1, x_2, ...x_n)$$

Using Bayes' probabilistic theorem rewrites the previous formula in this way:

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)}$$

That is, in Bayesan terminology:

$$Posterior = \frac{Prior * Likehood}{Evidence}$$

Evidence is a constant that depends on features data, it is a normalization constant. *Prior * Likehood* can be rewritten (assuming indipendent

features) as:

$$P(C_k|x_1, x_2, ...x_n) = P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

The Naive Bayes classifier combines this probabilistic model with certain decision rules based on the value of probability. This creates an algorithm that provides an output based on the probability calculated with the Bayes theorem. It is therefore assigned $\tilde{y} = C_k$ for a certain k based on the following rule:

$$\tilde{y} = argmax_{K \in \{0-1\}} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

There are various versions and extensions of this classifier such as Gaussian Naive Bayes, Multinominal Naive Bayes, Bernulli Naive Bayes,...

A theoretical limitation of this algorithm concerns the independence of features. In practice, in fact, this hypothesis is difficult to achieve. However, various tests have shown that even if this hypothesis is not guaranteed, the model often behaves well. A disadvantage is the behavior of the algorithm towards new features, as these will be assigned a zero probability and you can not make a prediction. So it needs well-formatted datasets and large enough to cover all cases. It is often used in real time applications, for multiclass classification, text classification, social media analysis, recommender system.

4.2.3 Support Vector Machine

Support Vector Machine can be seen as an extension of logistic regression. Its cost function is very similar to the one previously analyzed. In fact, it is a piecewise approximation of the logarithmic values within the cost function. Its structure becomes:

$$J(\theta^T X) = C \sum_{i=1}^{m} (y_i cost_1(\theta^T X) + (1 - y_i) cost_0(\theta^T X)) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

where $cost_1$ and $cost_2$ are the piecewise approximations of the logarithms. Moreover, the constant C is the term of regularization and the part $\frac{1}{m}$ is no longer present in the minimization of the cost function. Unlike the logistic regression which has only one decision threshold to obtain the prediction 0 or 1, the SVM has limit values beyond which the prediction can only be 0 or 1:

$$\begin{cases} \theta^T X \geq 1 \rightarrow y = 1 \\ \theta^T X \leq -1 \rightarrow y = 0 \end{cases}$$

There is an additional safety factor between -1 and 1 to be sure of the result. While in logistics the only threshold was around zero.

SVM is defined large margin classifier because it always leaves the greatest margin in the choice of the decision boundary, this increases the robustness of the algorithm, it is always chosen the line with more margin with the same fitting. This property is graphically visible in the figure 4.9.

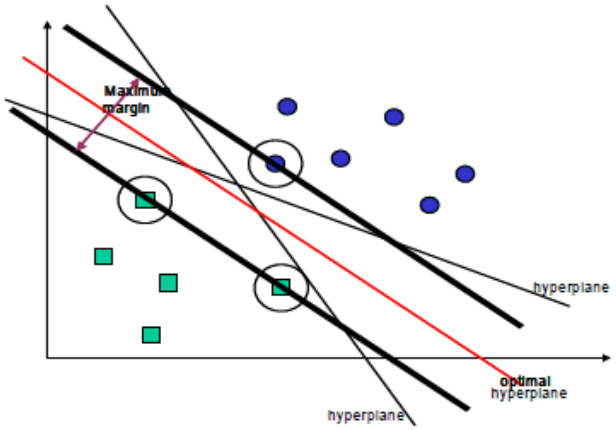

Figure 4.9: SVM: large margin classifier. The selected boundary is the farthest from the data.

SVM works by defining a similarity function that associates a value between 0 and 1 to each landmark-example pair. Each example is chosen as a landmark, so calculating the similarity function creates regions with a wide margin to identify areas where the output is 0 or 1. That is, each example becomes a nucleus from which the similarity with the other nucleus are defined. The similarity function can be more or less complex. Some examples are Linear Kernel, Gaussian Kernel, Polynomial Kernel, String Kernel and others... Due to its cost function it is possible to optimize the computational calculation making it fast and efficient. In other words, the SVM is based on the idea of finding a hyperplane that best divides a set of data. The term "support vector" identifies the points (examples) closer to the hyperplane, if they are removed or moved they modify the position of the hyperplane. The margin is therefore the distance from the "support vectors", these distances define the position of the dividing plane. The hyperplane can be linear or non-linear using the kernel idea.

The advantages of the SVM are: effectiveness in large spaces, memory efficiency, versatility in the choice of the hyperplane, accuracy.

4.2.4 Random Forest Classifier

This algorithm is based on the concept of decision tree. A decision tree can be considered as a predictive model for making classifications. It has a hierarchical structure formed by a series of ramifications that converge more or less slowly towards a result based on a series of decisions imposed during the branching of the tree. The main characteristics of a decision tree are:

Entropy : it is a measure of the unpredictability of the system, it is on the database itself.

Information Gain :when a decision is taken the database is divided in even more little parts, the entropy is reduced. The information gain is a measure of the entropy reduction after a node.

Leaf Node : it is th last node of the tree, where the decision is taken. Here the entropy is minimum and the classification estimation is performed.

Decision Node : nodes where the divisions are taken and the database is splitted into subsets.

Root Node : it is the top decision node, the database here is complete and the entropy is maximum.

Decision trees therefore lead to a result based on successive choices that tend to decrease the entropy of the dataset. They are usually used for classifications, but if carefully imported they can also be used for regressions.

A single decision tree is usually characterized by great variance, so we use a technique that increases the security of the choice called Bagging. Bagging (or **B**ootstrap **Agg**regat**ing**) is a resampling technique. The data is sampled with repetition several times, thus obtaining many different datasets. After which many decision trees are created, one for each dataset and finally the average of the results thus obtained is calculated or the most recurrent result is chosen. These measures greatly reduce the variance and make the result more reliable.

The algorithm of Random Forest is based on the idea of Bagging, but improves it because the various trees are not correlated each other. In other words, the various re-sampled datasets are scrolled between them, thus obtaining a wider safety margin and a greater reduction of the variance. The term "forest" refers precisely to the fact that various decision trees are used. The algorithm uses them to make classifications between independent resampled datasets. The chosen result is the most repeated, the one that happens most frequently in the forest. The important thing

to underline is that each dataset has a different decision-making model based on the characteristics of its own features.

The random forest algorithm has some main advantages: the use of many decision trees limits the problem of overfitting. Due to the simplicity of the model, the algorithm is fast in terms of training time. It runs efficiently on large datasets and works well even in case of datat missing. So it is an excellent algorithm when you do not have well structured data or not easily correlations because its great versatility finds the right decision boundary.

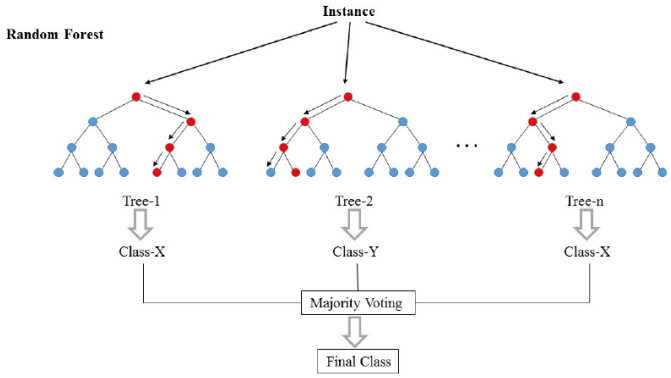

Figure 4.10: RFC: general structure of a Random Forest Classifier. First the dataset is divided in independent subset, each of them make a prediction based on their decision tree. The final outcome is a joining of all the decision tree results.

4.2.5 Nearest Neighbor

It is an algorithm that is based on the similarity of the characteristics of a certain data that you want to classify with respect to k known elements closer to it. Once fixed the k some regions are defined in the space of the features where an element for proximity to the other known elements is labeled according to the characteristics of the dominant elements of that space. Each element within a known region is immediately labeled. If the classification is binary (only 0 or 1) it is better to select an odd value of k to make sure you always have a majority in a given region of space. For the same reason k should never be multiple of the number of classes. The main disadvantage of this algorithm is the complexity of searching for the nearest elements for each new example.

The operation process of this algorithm is easily visualized graphically in the case of only two features in the database. In the figure 4.11 you can see how the regions of the classification space change as the value of k increases.

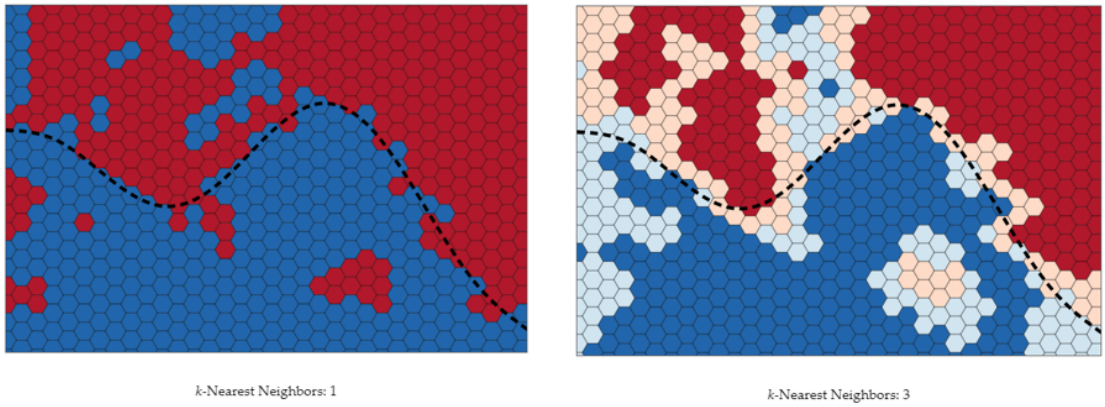

Figure 4.11: KNN: classification changes basis on different k. The lighter color indicates a lower reliability of the solution, i.e. those regions have neighbors of not unique classification.

This algorithm is very powerful when there are large non-linearities in the structure of the dataset, in fact the KNN algorithms are based on similarities and can create very complex decision boundaries, differently from any mathematical formula in other models. The result is an incredibly flexible algorithm, especially if the value of k is low. In this case, however, the fit of the data is too precise and you risk running into the problem of overfitting because the decision boundary is fragmented and too detailed. On the contrary, if k increases the decision boundary becomes more and more defined.

As we have understood, choosing the appropriate value of the parameter k has a great influence on the performance of the algorithm, in fact often entire regions change classification according to its value. To choose it correctly there are some general rules that should be followed. Usually choosing k as the square root of the number of examples in the dataset gives pretty results, but it should always be rounded to the odd number nearest the exact square to avoid possible confusion between two classes of data.

In the case of large datasets the value of the parameter k will be high, this considerably increases the complexity of the algorithm and slows it down. For this reason it works best on small or medium datasets. KNN works well on well-labeled, noise-free databases. It is also defined as *lazy learner*: doesn't learn discriminative function from the dataset.

Mathematically, the similarities on which the algorithm is based are calculated on the Euclidean distance in the multidimensional domain of the features of dimension n. A general structure of this distance is the

following:

$$d^{(i)} = \sqrt{x_1^{(i)2} + x_2^{(i)2} + \cdots + x_n^{(i)2}}$$

However, it is possible to use different distance formulas in order to give more or less importance to some features. It is evident that as the number of features n, number of examples m and value of the parameter of neighbors k increases, the complexity increases considerably in the calculation of the distance and therefore in finding the classification regions. Any algorithm that calculates measurements from features always needs to normalize the data before performing the calculation. This is necessary to give each feature the same importance.

4.2.6 Neural Networks

Neural networks are a very powerful tool for machine learning. In recent years they have become increasingly popular due to the fact that the computing power needed to manage them is now available. The neural networks are in fact a tool known for decades but really never used until a few years ago because of the scarcity of computational resources of the years of the 900'.

The neural networks are so called because their structure tries to emulate the human brain. Basically, in fact, they are made up of neurons interconnected among them by means of dentrites and axons. The dentrites receive the various inputs from the other neurons, while the axon is in charge of transmitting the processed output, which depends on the inputs received. Therefore various complex connections are formed, these connections create the network. Each input to the cell is weighed and therefore the importance changes through appropriate coefficients to be determined during the training phase of the network.

Technically, an artificial neural network is composed of three types of layers:

- Input Layer
- Hidden Layer
- Output Layer

In the simplest cases there are only three levels, but for more complex networks the number of hidden layers can be greater than one, while input and output layers are always single. If the classification to be carried out is binary, the output layer is composed of only one cell able to estimate if the output is zero or one, while for multiclass classification the output layer is composed of several cells.

The learning of a neural network involves, after having fixed the structure of the network, to find the parameters that weigh the various signals carried by the network. During the training, the examples are analyzed individually, as the machine processes the outputs, it proceeds to correct them to improve the responses by varying the weights.

The training mechanism is a continuous iteration for all examples of the binomial *forward propagation* and *back propagation*. It starts with random weights, propagates directly to the predicted output, calculates the error with respect to the actual result, and performs back propagation to calculate the error at each step of the network. In this way, the coefficients that cause the error are calibrated. Also for this algorithm we try to minimize the cost function related to the errors. This process is a complex process that can be quite slow. However, the efficiency of a well trained neural network is usually very good and the results are often more satisfactory than those of other algorithms.

The user of the neural network libraries does not have to worry about implementing propagation. Its task is to select the appropriate network structure in terms of size, number of layers and number of neurons. This is a delicate and not easy task. It is a good idea to always start from a single hidden layer and try to increase the number if the performance is not satisfactory. More hidden layers must have more or less the same number of neurons, this number must be comparable with the number of input features for each example, to not create an unbalanced network.

in figure 4.12 is shown a typical structure of a neural network at several levels for binary classification.

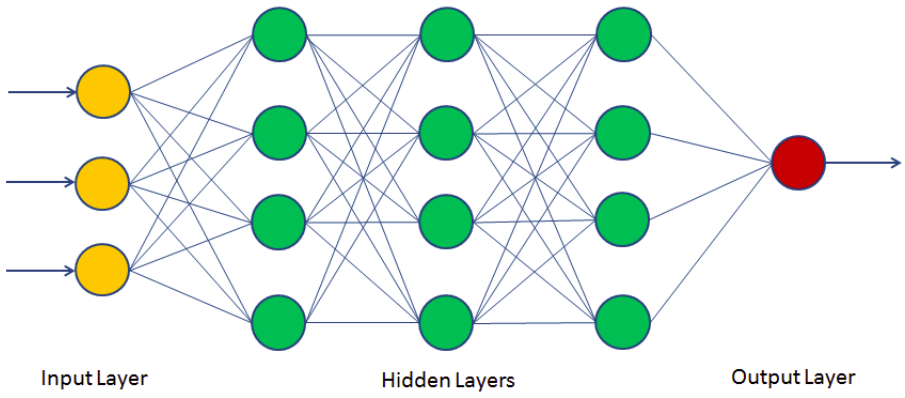

Figure 4.12: Neural Network: a typical structure: the network shown has three input features, a single binary output and three hidden layer equal is size and comparable with number of features.

The advantages of a neural network are the high parallelism thanks

to which a lot of data can be processed in a short time, the tolerance to faults, the tolerance to noise, the self-adactivity to data change. The limits, on the other hand, are linked to the fact that it is not always clear how the network arrives at the results, so the algorithm behaves like a black box. In addition, the training period is long due to the complexity of the algorithm.

4.2.7 Linear Discriminant Analysis and Principal Component Analysis

These last two algorithms are not classifiers and therefore are not used to make predictions of the results. The objective is to discard irrelevant or less relevant informations to the problem of interest.They are used in the preprocessing phase to reduce the number of features and compress them to make the classification algorithms more efficient and make the data well separable so as to reduce overfitting.

LDA and PCA are both linear transformation techniques. PCA is an unsupervised learning algorithm that identifies the direction of the *principal components* that maximizes the variance of the database (preserve pattern information to the fullest possible extent). LDA is instead a supervised learning technique that finds the direction that maximizes the separation between classes to facilitate classification (favours dimensions that discriminate against classes).

The structure of the LDA process is as follows:

1. Calculation of separability between classes.
2. Calculation of the distance between the class average and the examples for each feature.
3. Build the smallest dimensional space that maximizes the first step and minimizes the second.
4. Project the original data into the created subspace.

Some of the disadvantages of the LDA analysis are the low efficiency in small databases and the problem related to the non-linearity of the initial features. If linearity is not guaranteed, the LDA analysis cannot find the subspace to reduce features dimension. This problem can be partially solved by using the kernel, similar to SVM kernel technique. LDA analysis is often used in speech recognition and in many medical applications where features are linearly separable.

The PCA aim instead, is to reduce the number of variables describing a dataset to a smaller number, limiting the loss of information as much as

possible. This happens through a linear transformation of the variables that projects the original ones in a new Cartesian system in which the new variable with the greatest variance is projected on the x axis, second axis for second size of the variance, and so on. Unlike other linear transformations of variables practiced in the field of statistics, in this technique are the same data that determine the vectors of transformation, and that is why the PCA is defined as a technique of unsupervised learning.

In figure 4.13 a representation of the PCA and LDA application is shown for a dataset with two featuresx x_1 and x_2. The black segment that identifies the PCA solution is the hyperplane on which projecting the patterns (regardless of their class) we keep as much information as possible. The green segment that identifies the LDA solution is the hyperplane on which, by projecting patterns, we are able to distinguish between the better the two classes (red versus blue patterns).

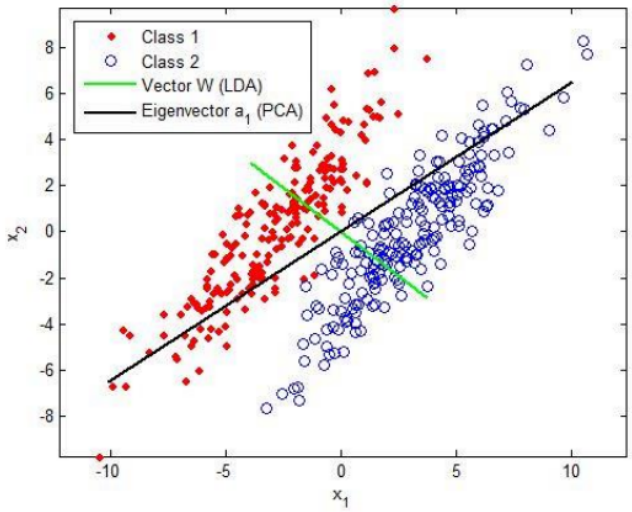

Figure 4.13: PCA and LDA representation in a two features size database: the green and the black lines are a compression from 2D to 1D for LDA and PCA. The objective on LDA is to separate the classes, the one of PCA is to retain the maximum amount of informations.

4.3 Regression Learning Algorithms

As previously done for classification problems, in this section we will mention the main algorithms of machine learning for regression problems, that is with the desired output to be searched in a continuous domain.There

are other types of regression algorithms that can be used in certain particular cases when the most common, mentioned here are not suitable for the problem to be solved. In this section we list the most important.

4.3.1 Linear Regression

Linear regression is the first and basic concept of machine learning for regression problems. As already mentioned above, it has such a hypothesis function:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 * x_1 + \vartheta_2 * x_2 + \cdots + \vartheta_n * x_n$$

As you can see is a linear prediction formula in the coefficients and in the inputs. The coefficients ϑ have to be estimate and the features x are the input informations taken from the database. The cost function to optimize to find the best ϑ to have minimum error is instead:

$$J(\vartheta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\vartheta(x^{(i)}) - y^{(i)})^2$$

As you can see the cost function tries to minimize the error between the prediction made by the hypothesis function with certain parameters and the real output desired. This means that the parameters must be changed several times before reaching a satisfactory result. The selection of the parameters therefore takes place in an iterative way according to the following formula:

$$\vartheta_j = \vartheta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\vartheta(x^{(i)}) - y^{(i)})x_j^{(i)}]$$

Here there is the introduction of a new parameter, the so-called *learning rate* α. Tuning this parameter change the velocity of the conversion and in its accuracy. A big value of α may create problems of oscillations in the convergence or even divergence, whie a too little value imply great precision but very slow convergence so the number of iterations required to converge at the minimum of the cost function will be huge. Determination of the learning rate parameter is essential for a good choice of learning parameters.

In the figure 4.14 it is visible a fitting of the data with linear regression, the figure shows a two-dimensional representation. It is possible to imagine a straight line in more dimensions to the increase of the features that makes the solution no longer graphable.

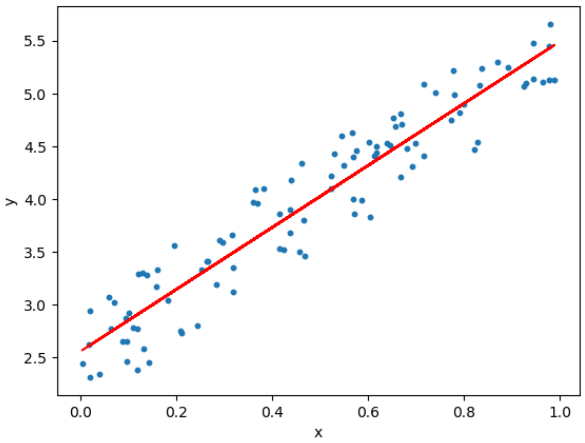

Figure 4.14: A typical example of linear regression fitting in 2D.

4.3.2 Polynomial Regression

The polynomial regression has the same cost function as the linear regression from which it starts. The objective is always to find the best coefficients that minimize the error between prediction and true output. What changes is the form of the hypothesis function. The input features are now interpolated obtaining a polynomial between degree greater than one among the features while the linearity of the coefficients remains. In fact it would be better to define this kind of algorithms as linear regression with polynomial features.

It is possible to have any degree of interpolated features, the more the degree grows, the more the number of total features will be. For example, think of a machine learning problem that involves only 2 features. The hypothesis function of a normal linear regression is:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 * x_1 + \vartheta_2 * x_2$$

while for a polynomial regression grade 2 we will have:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 * x_1 + \vartheta_2 * x_2 + \vartheta_3 * x_1^2 + \vartheta_4 * x_2^2 + \vartheta_5 * x_1 * x_2$$

and grade 3 becomes:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 * x_1 + \vartheta_2 * x_2 + \vartheta_3 * x_1^2 + \vartheta_4 * x_2^2 + \vartheta_5 * x_1 * x_2 + \vartheta_6 * x_1^3 + \vartheta_7 * x_2^3 + \vartheta_8 * x_1^2 * x_2 + \vartheta_9 * x_1 * x_2^2$$

As you can see the complexity of the hypothesis function grows a lot as soon as the degree of polynomial interpolation rises. For what has been

said so far, if the selected degree is high, the fitting will be more and more precise with a great risk of overfitting. Moreover, the number of features increases exponentially, so the computational calculation is a determining factor in the polynomial regression.

If you have only one feature you can also represent this hypothesis function in a 2D plane as shown in figure fig:regpol where the interpolation no longer generates a straight line but a curve that approximates the real values.

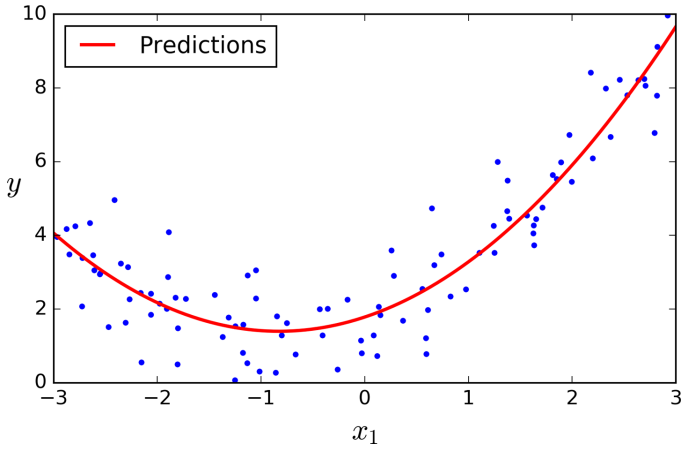

Figure 4.15: Polynomial regression, the hypothesis function is now a curve.

4.3.3 Ridge and Lasso Regression

These regression algorithms are based on linear or polynomial regression with the addition of a term in the regularization cost function. This device has been thought to avoid the problem of overfitting, therefore the term has the function to give less importance to the coefficients in the hypothesis function of certain terms, so that the fitting of the data does not lead to the above mentioned problem.

In detail, the Ridge regression has the following cost function:

$$J(\vartheta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\vartheta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m}\vartheta_i^2$$

λ (lambda) is defined as the tuning parameter, which multiplied by the sum of the squared coefficients (excluding the intercept) defines the penalty term. It is evident that having a $\lambda = 0$ means not having a penalty in the model, that is we would produce the same estimates that with the minimum squares. In another way having a λ very large means having a

high penalty effect, which will bring many coefficients to be close to zero, but will not imply their exclusion from the model. The ridge regression method never allows the exclusion of estimated coefficients similar to 0 from the model. This lack, from the point of view of the accuracy of the estimate, may not be a problem. A problem connected to this limit is on the side of the interpretability of the coefficients, given the high number of predictors.

The lasso method (least absolute shrinkage and selection operator) fills the disadvantage of the ridge regression. It allows the coefficients to be excluded from the model when they are equal to zero. It can be noticed that the formula of the ridge regression is very similar to that of the lasso, the only difference consists in the structure of the penalty, in how much it is necessary to calculate the summation of the absolute value of the coefficients:

$$J(\vartheta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\vartheta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m}|\vartheta_i|$$

As in the ridge regression the lasso method forces the estimated coefficients towards zero but, the absolute value present, forces some of them to be exactly equal to zero.

In general, it can be expected that ridge regression will do a better job when the number of predictors is high, and at the same time that the time frame will do a better job when the number of predictors is small.

4.3.4 Support Vector Regression

The support vector machine (SVM) method already seen in the regression algorithms part can also be used for part and regression while keeping the characteristic part of the massive margins intact. If used for regression its name becomes Support Vector Regression (SVR) with some differences compared toSVM: it is impossible to estimate a number with respect to an infinite range of possibilities so we use a range within which the solution is defined as ideal. The main idea is always the same, however, to minimize the error by finding the hyperplane that maximizes the margins, taking into account a certain rollerance in the error as just said.

Even the kernel function has the same function of transforming the data into a higher dimensional feature space to make it possible to perform the linear separation.

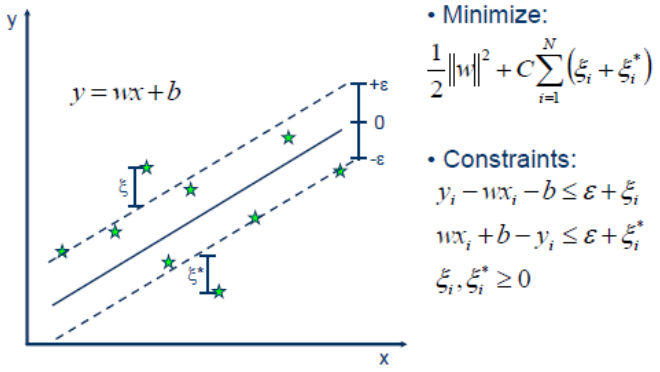

Figure 4.16: Support vector regression with the usage of tolerance range for finding the maximum margin.

4.4 Supervised Algorithm Choice

There are many types of machine learning algorithms and many libraries are able to implement them in a performing way. The choice of algorithm to use is not obvious, because each algorithm has strengths and weaknesses and there is no better model than the others but depends on each situation.

Given the large number of algorithms available, it is necessary to create a selection process through 2 phases: before and after training in order to train a limited number of algorithms based on the characteristics of the dataset.

4.4.1 Choose feasible algorithms

Before training the algorithms it is already possible to partially understand which of the various possible choices will be those that will work best for our dataset. This group of algorithms is based on the characteristics of the dataset. The parameters to select them are:

- Linearity of the database
- Missing values in the dataset
- Data repeatability
- Examples and features number
- Number of algorithm parameters
- Class imbalance

Linearity of the database

Some algorithms of machine learning use hypothesis of linearity, it is meant that the classes or the features of the database are separable from a linear function like a straight line or from a space o hgher dimensions equivalent and linear. This hypothesis is difficult to ascertain and often not applicable to real cases. It happens that the features are dependent and not separable, for this reason the linear algorithms obtain not good results. More realistically, the data needs polynomial interpolations or very complex decision boundaries. However, a high degree of polynomial interpolation is very expensive at a computational level and may therefore make a probabilistic method preferable to an exact one because of the high complexity required.

Linear algorithms are however an excellent starting point to understand how the dataset is structured and to direct the choice to more complex models, avoiding trying to find the best model with a brute force technique.

Missing values in the dataset

A very common feature in databases is the presence of missing values between the data. These are due to different phenomena such as the diversification of the examples in the dataset, data sampling errors, errors during the pre-processing phase,... It is very important to take into account the missing values because some algorithms work badly or do not work at all in the presence of missing values between the data. This problem can be solved in two ways: The pre-processing phase is extended by deleting examples or modifying data; the choice of an algorithm that is less sensitive to missing values.

In both cases the solution is not optimal because you lose time and you generally get worse solutions than you would get with a homogeneous dataset without missing values. The techniques of treatment of these values are many and if necessary they will be discussed when they occur.

Data repeatability

Any algorithm to work well requires that the outputs to be predicted depend on features very similar to those with which the algorithm was trained. However, some models are more sensitive to changes in the input data to be predicted. These make them adaptable to the training set but with a poor behavior on the test set and future predictions. This is due to the more or less important change in input data over time. It is therefore

better in these cases to select an algorithm with worse performance on the training but with greater versatility and repeatability of accuracy, this increases the reliability of the output.

Another solution if you want to keep the same high-performance algorithm in your training is to increase the training time and data to cover all possible cases. Finally, the best performance is obtained by making the algorithm self-learning over time, i.e. using the prediction data as subsequent data to perform a new training by increasing the total dataset and adding information from the new examples. With this care the algorithm adapts to the changes that can occur in time and the output remains reliable over time. The cons is that this continuous training requires high computational costs to keep the model up to date, this is often not acceptable during a normal application of the model.

Examples and features number

The size of the database is relevant not only for the training time but also for the intrinsic ability of an algorithm to properly manage a large amount of information. For some algorithms it becomes difficult to extract information on what are the relevant features causing overfitting problems that are difficult to solve and identify. In the same way there are opposite problems related to data underfitting if the size of the database is not adequate. In any case, it is necessary to act on the modification of the features, using techniques of regularization or extension of the features. Some algorithms are better then other to extract these informations without modifying the database and this can imply less time to dedicate at the preprocessing phase.

Number of algorithm parameters

The parameters of the algorithms are those coefficients that characterize the setup of the model. An algorithm with many parameters is more difficult to optimize than one with few, and requires a lot of experience to be used. In general, an algorithm with several parameters is more flexible and adapts well to many cases, but it is difficult to find the right combination of these parameters to achieve the best possible performance.

Class imbalance

For classification, an unbalanced dataset guides the user to the algorithms that will perform best. Usually having a great imbalance between the classes we tend to use more robust algorithms specifically designed for

this function. Algorithms based on research trees are very powerful in these conditions.

4.4.2 Best algorithm selection

Once you have chosen the group of algorithms that you think can best behave with the dataset you have, you can train these algorithms and evaluate which is the best of them on the basis of:

- Performance
- Training time

Performance

One of the most important features of a machine learning algorithm is that it predicts data well. This is the main purpose of any of such project. Performance evaluation is done by using a test set to simulate the algorithm's task, i.e. predict results before they occur. Thanks to the use of the test set it is possible to extract the evaluation metrics that quantify in a simple and intuitive way the goodness of an algorithm. They also make it possible to compare different models so that you can choose the one with the best performance. For further information, please refer to the chapter 4.1.5 where the main ones are discussed in detail.

Training time

It is another characteristic to take into account, especially when comparing various algorithms. If you have to do several tests and trials it is a good idea that this time is limited in order to evaluate the algorithm. Moreover, there is often a constraint imposed by the project on the time for elaboration. This time is closely related to performance but some algorithms, especially those based of probabilistic models, may take a short time to optimize the parameters and still obtaining good results. The training time factor can limit the possibilities for a decision of certain algorithms especially for very large databases because even slightly better performance can lead to long computational calculations not acceptable.

Chapter 5

Error Prediction: Data Preparation

This chapter and the following one are dedicated to the problem of error prediction. As mentioned in the chapter 1.2 the problem to be solved is to try to predict abnormal machine stops and badly finished parts. The solution chosen involves the use of machine learning algorithms to find correlations between the inputs provided to the machine. To do this, the first phase of work is to reconstruct the series of inputs that can be decisive for a possible failure of the machine. This will be discussed in this chapter, while in the next we will test various algorithms to find the one that best suits the problem.

In details we start describing the creation process of a single large table of *cuts*. Each *cut* event will be associated with all other related events in such a way as to extend the features of the data and add all the information obtainable from the raw database described in the chapter 3. From this unified table it will then be possible to easily select subsections related to the machines concerned.

In the second part we will look for information about the correlations between features, trying to identify the most similar and dependent ones. Ensuring independence between features is essential to avoid repetition, lighten computational calculation and get better results from machine learning algorithms.

Finally, a statistical analysis will be carried out between the features of each machine. This will eventually justify a separation of the application of machine learning algorithms for individual machines and will then also define the subsections for best results.

In this chapter and in the following ones we will take for granted the

knowledge of the Jupyter Notebook programming environment, as well as of the Python programming language and of the main libraries used for data management, that is Pandas and NumPy. Further information on the working environment is presented in appendix one, chapter 9.

5.1 A bug in manual mode data collection

A problem come up when plotting the number of occurrences of a badly finished cut according to the *CutLength* feature. To do this, we filtered the extended database by taking only the lines containing a *BadTerminated_MC==1* and we used the *.hist()* method of Pandas to create the histogram. In this study all data of the three machines were used.

In figure 5.1 it is visible the diagram of the occurrences of a terminated badly in function of the length of cuttings given as input. As you can see, there are 2 cutting lengths in which many errors are generated, that is, when the cut is set to $0mm$ and $3200mm$, while the rest of the errors are distributed over all the other lengths.

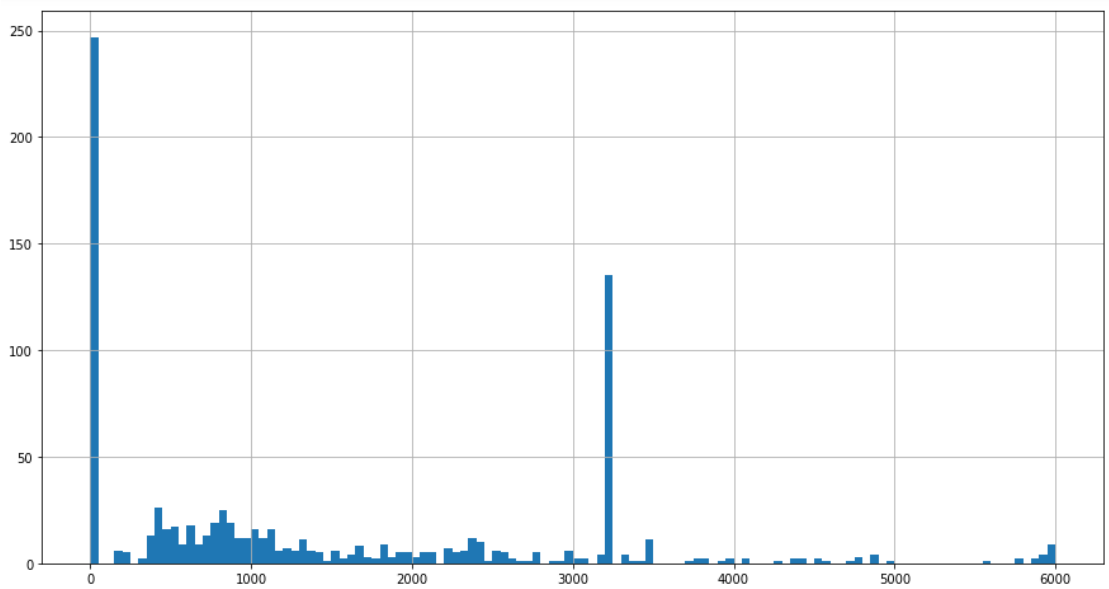

Figure 5.1: Bad terminated cuts occurrency in function of cut length: two peack are clearly visibles where the errors occurr.

Two further subgraphs have been obtained from the previous one, they are visible in the figure 5.2. By dividing the manual steps from the automatic ones, in this way it has been discovered a very important fact that changes the study conducted. In the graph of the automatic steps

the peak on the length 3200mm remains, this peak is due to the fact that the glass sheet have dimension in height of this measure, therefore it is the dimension more cut and the more recurrent, and so it is on this measure that the greater part of the errors is concentrated. The manual step graph, on the other hand, has only one value for which the errors occur, namely the value of *CutLength==0*. It was thus discovered that the software that generates the badly finished events for manual cuts contained a bug and the errors were only sent when the selected measurement was zero, without taking into account all the other potential bad situations that can occur in the reality of the problems.

The main consequence is that it is not possible to use the steps made in manual mode in the machine learning algorithm because they are buggy. From now on, therefore, the manual steps identified by the feature *StepManual_ C== 1* will be discarded. All the graphs and tables will refer from here on only to cuts made in automatic mode.

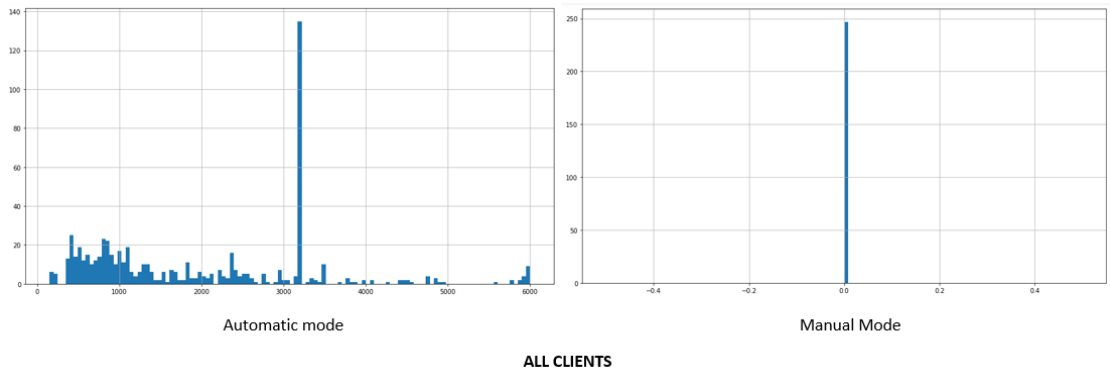

Figure 5.2: Automatic and manual occurrency of bad terminated cuts: from this graph was discovered a bug in manual mode data sending. From now on manual mode data are no more used.

A further check of the above is visible in the figure 5.3 where you see the same histogram graph of the occurrence of errors as a function of the cutting length divided by customers. This shows how the bug is contained in the software of each machine analyzed, as the badly finished occur daily at different lengths even if the step is manual.

5.2 Data merging

The term *data merging* means the union of several datasets into one, extended, containing information from all the sets that compose the

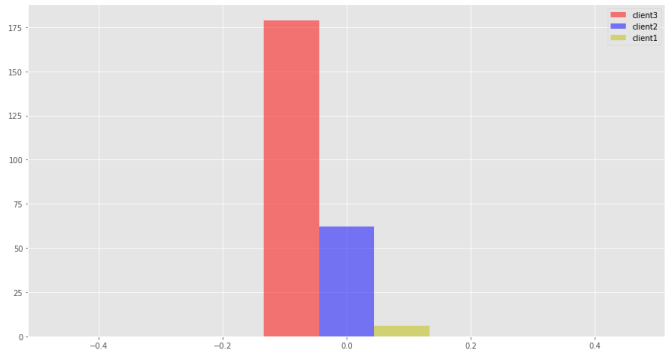

Figure 5.3: The discovered bug affects all the clients: no error registration for cut length different from zero in manual mode operations.

database. This process is necessary when you have multiple sources of information, raw data or multiple separate tables for storing data. You can divide this process into three possible groups:

Extend examples : Merging in this way extends the number of examples, i.e. the length of the table. This method of unification is also called "hanging new data". It is assumed, in this type of merging, that the tables to be merged have the same variables and that the meaning of the columns is the same for both tables. It is also good to check that the various examples (the rows, the entries of a table) are not repeated in the various tables to be merged, otherwise it is necessary to remove the duplicates after merging the data.

Extend the number of features : Extending the number of variables requires some correlation between features of the two tables to be merged. Without a proper correlation it is impossible to merge them. Normally this type of merging is done on the basis of table indexes or on the value of some feature that is repeated in both datasets. The output of this merging is an extension of the columns with respect to the starting datasets. The entries remain the same but the amount of information for each example increases. Also in this case you will eventually have to delete the related features to avoid linear correlations between them.

Associate features uing look-up : This type of merging is the most complex and is used when you have incomplete datasets to combine, i.e. with different features and different information contained. There is not an equal column for merging, only a few data in scattered order are the link between the two tables. The method of association is therefore done through some values of one or more features present in both datasets, or through an algorithm that can identify for each

example of the first set exactly a value of the first set. This implies that this merging can take place between tables of different sizes as long as there are direct or indirect correlations between some features of the tables.

An example is the association of a geographical region (based on the zipcode) to the examples of a table where each entry corresponds to the districts of a city. The table to be extended contains information about the neighborhood and the zipcode of the city. Another table instead correlates zipcode and region of origin. It is not certain that all the zipcodes of the second table also belong to the first and not even the opposite if the second table is incomplete. But surely if the zipcode of the two tables coincides you can assign the first table to the region of belonging, so as to know the location of a certain neighborhood in the corresponding region. This merging method is a kind of cross association between the two tables.

In our case the second and third merging methods will be used, as we do not need to first that extend the examples because for this work they are always all contained in the database described in the chapter 3.

5.2.1 Basic idea

Here are described the guidelines of the process that will be analyzed technically and in detail in the following sections. What you want to get is an extended table for the events of type Cut containing information of other events of Step, TypeOfGlass, Glass, Session. The method of correlation of the tables is different for each event and will be described later. The desired output is a table with m examples (corresponding to the number of examples of the cutting event) and n features. The n features are due to the sum of the single columns coming from each event. Mathematically:

$$n = n_{cut} + n_{step} + n_{ses} + n_{type} + n_{glass}$$

where each n_i is a subsection of the total number of features present for event i. If you look at the chapter 3 you will see that each example consists of a set of features, only the most relevant of them are selected to extend the table of cuts.

In table 5.1. you can have a schematic graphical display of the desired output, with the features from the events just described. It is important to note that the unified table has as its starting point the table of cuts and extends based on this. So the output can be defined as an extension of the table of cuts that collects much more information from other events.

Cut Event	Step Event	Session Event	TypeOfGlass Event	Glass Event
$1 \dots n_{cut}$	$1 \dots n_{step}$	$1 \dots n_{ses}$	$1 \dots n_{typ}$	$1 \dots n_{glass}$
		n total features		

Table 5.1: Schematic shape of table obtained after data merging.

5.2.2 Cut table

As defined in the basic idea,it is necessary first create a table of cut events. This table is created by extracting only this type of event from the database based on the *EventCode*. You then delete the unused database columns and rename the remaining ones to give them explicit meaning. Finally, extract the information from the mask in *EventData2* and insert it into the new features as boolean variables. Information about this event can be found in the paragraph 3.1.4.

The figure 5.4. shows this procedure of creating the table of cuts in a flow chart easy to understand. In a real approach I used Python with the library Pandas for managing dataframes. TEvents contains part of the entire database from a prefixed date, so the *TEvents* dataframe is structured exactly as described in the chapter 3.

5.2.3 Associate Step event

To associate step events to the newly created cut table, you must also create a suitable step table. Not all the steps, however, are related to a cut event, only the steps of type *StepVsxCut5X8* and *StepVsxDiagonal-Cut5X8* (see table **??** for further details), this is due to the fact that, as the software that manages and sends the events to the database is structured, every time a step of the type *StepVsxCut5X8* and *StepVsxDiagonalCut5X8* takes place, a cut event is also recorded. In these cases the two cut and step events are sent together at the same time. In practice the two events are recorded in the same second but with a difference of a few milliseconds. This information is the basis of the association between the two tables, in fact, based on the hypothesis that a machine can not do more than one cut per second, you can truncate the datetime of the two events in the range of seconds and use them as a comparison between the events to do the merging of data.

This procedure can lead to a slight loss of data if the cut and step events differ by several milliseconds so that their truncated DateTimes per second differ. If the number of seconds is different, the merging is

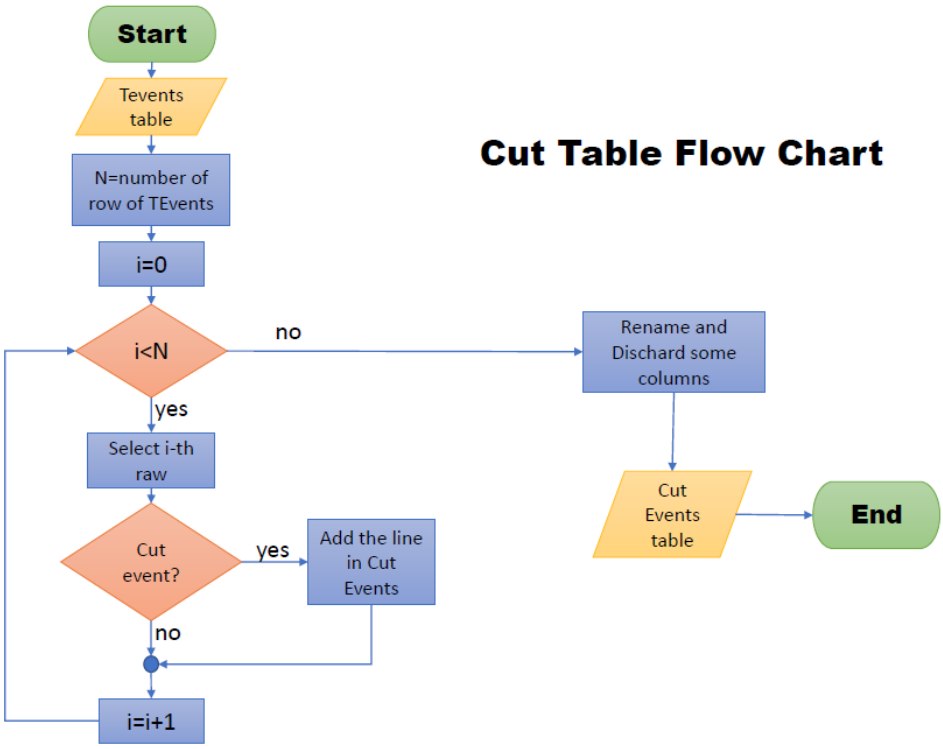

Figure 5.4: Process of cut table creation in flow chart.

unsuccessful and both parts are discarded. From the verifications carried out the loss of data is negligible.

In figure 5.5 you can see the flow chart tocreate the step table when the step type is *StepVsxCut5X8* or *StepVsxDiagonalCut5X8*, the other columns of the table are those extrapolated from TEvents and defined in paragraph 3.1.3, some of them are discharded or renamed for an easier usage. As explained before the column *DateTime* will be used to merge with the*ProcessorID*.

In the figure 5.6 there is the steps to create the extended table Step + Cut. The merging keys (i.e. the values to be compared) are the *ProcessorID* of the machine that generates the event and the *DateTime* approximated to the seconds. In the real implementation code an*inner join* is executed: only when both keys coincide a new entry in the extended table is created. This means that in the new table you will never have missing values due to the non-association of the two table entries. The negative aspect is a slight loss of data considered not to be influential.

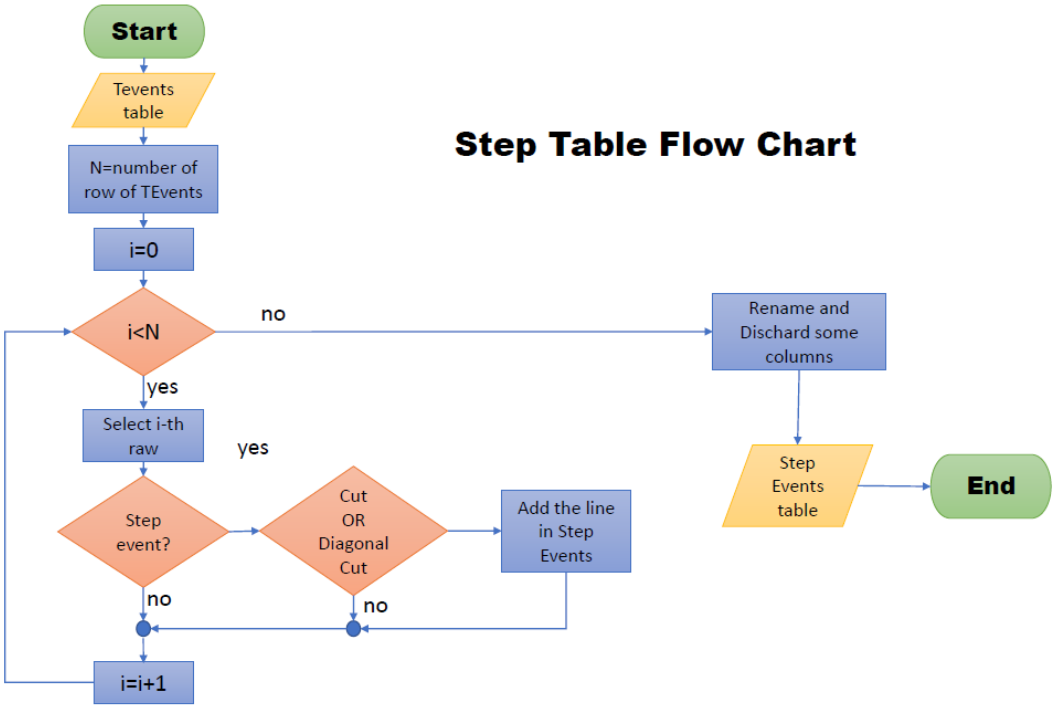

Figure 5.5: Step table creation flow chart.

5.2.4 Associate other events, the example of Session event

What will now be explained for the session event is also valid for the other TypeOfGlass and Glass events.

To insert the information about the session in the previously created cut + step table, a separate table is used, containing only information about this event code. The session event extrapolation program and the creation of the relative table is very similar to the previous and its flow chart is omitted. Remember that the meaning of the features for this event is described in the chapter on the database in paragraph 3.1.1.

To join these new informations coming from the Session it is not possible to perform an *inner join* to extend the table (like before between Step and Cut). This is due to the fact that the number of Session events is much lower than the number of cuts and there is not correlation one by one. Remember that the Session event is sent when the machine is turned on, then when the machine is started many cuts are made. This means that many cut events will be associated with the same session. To associate this session to the various cuts the algorithm searches, for each cut of a specific machine, for the last session event that took place (in

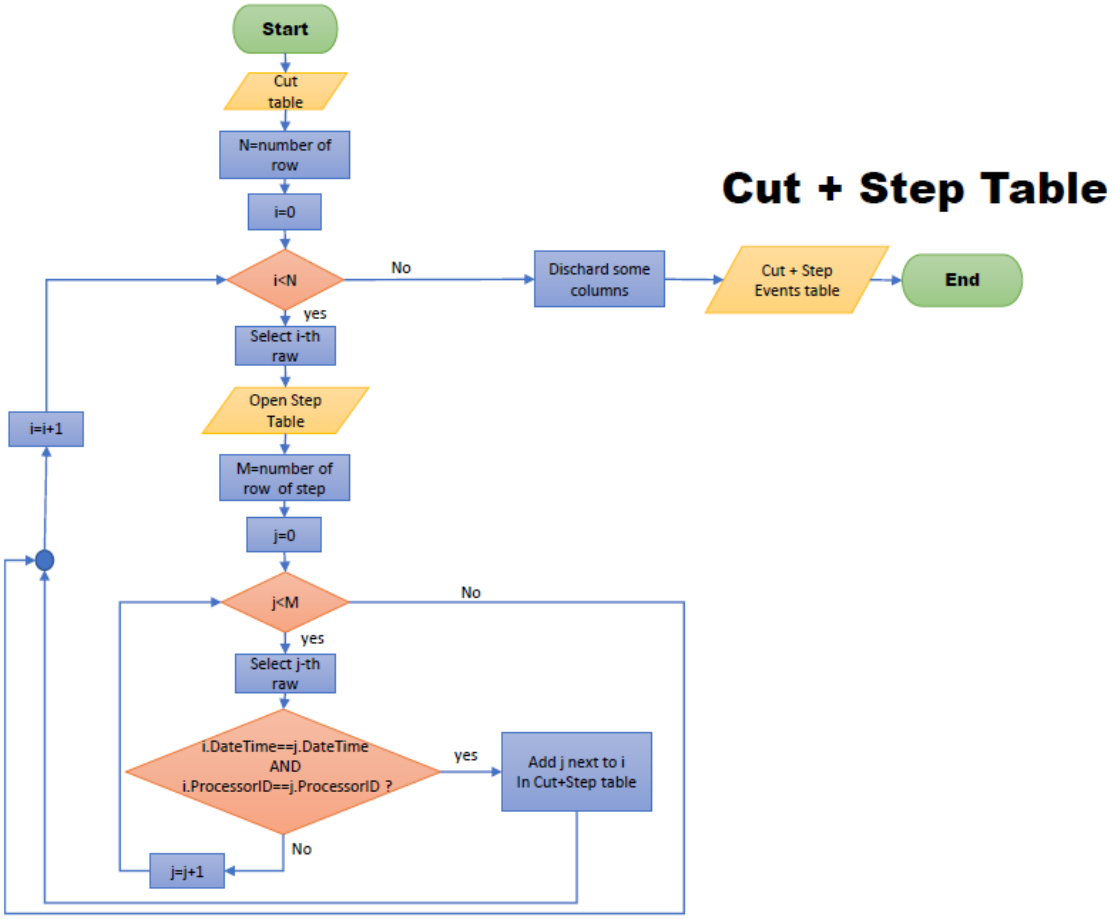

Figure 5.6: Merging of cut and step table by inner join with keys *DateTime* and *ProcessorID*

chronological order).

To do the merging the execution of a *for* cycle that runs through all the entries of the cut + step table is performed. For a specific i-th example, the *DateTime* and *ProcessorID* are extracted and the session table is cycled to find the closest and previous *DateTime* to that of the cut with the same *ProcessorID*. When you find the desired session its features are extracted and go next to the row of of step + cut table. The flow chart of this algorithm is visible in the figure 5.7.

It may happen that towards the end of the research null values appear and the table is not completely populated. This is due to the fact that the algorithm does not find precedent session events with respect to the cuts. Obviously, every cut is made when the machine is already started, but this problem occurs when the partial selection of the database starts from

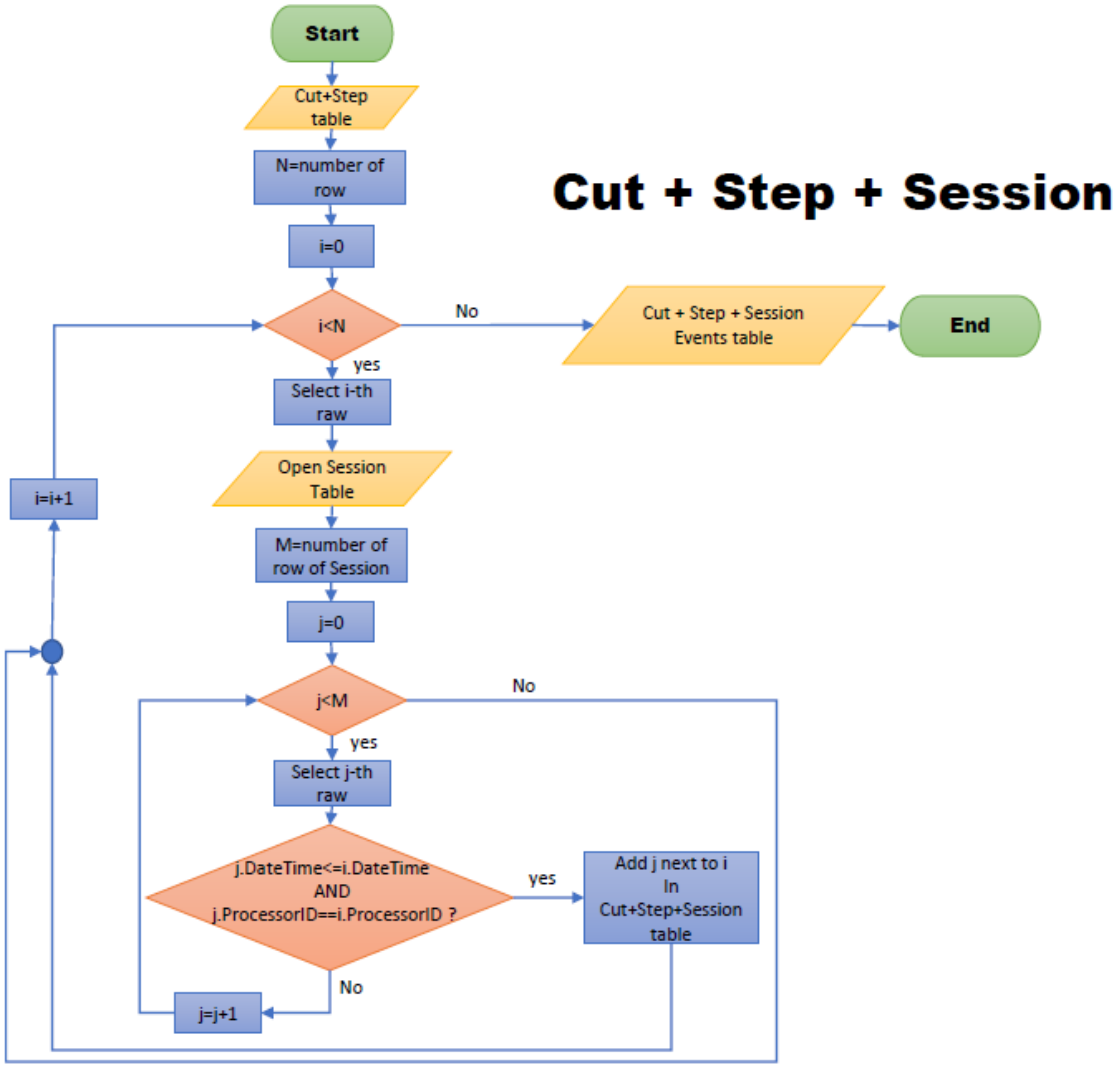

Figure 5.7: Creation of Cut+Step+Session table.

a certain sampling start date, when the machine can already be started, and therefore already switched on. For this reason it is not possible to associate data relating to a session event with some cuts very close to the sampling start date (i.e. the tail of the cut + step table).

At this point you get a table Cut + Step + Session.

The same algorithm is reused for the TypeOfGlass and Glass events and is therefore omitted from the description.

5.2.5 Final reshape

Before saving the created extended table, consisting of the merging of the 5 events, it is better to perform some actions to make the table more functional by adding and deleting columns. 3 new features are then inserted, they are related to the difference in time between the cut event and other events except the step (which has the same *DateTime* of the cuts). These columns contain a difference in time expressed in seconds. This addition is not based on any justifiable reason, but is an example of how the number of columns can be increased by mixing information from multiple features.

Finally, many columns containing repetitions or no more useful information are deleted.

Moreover, other variables are deleted:

- *DateTime_x*: event timestamp, being of type *object* cannot be used as input of machine learning algorithms.
- *ExecutionTime*: Column containing the execution times of the step. Is obtained after the step has been executed, so it is an output that should not be considered at the time of prediction because it will not yet be available in real cases.
- *HeatingTime*: another temporal measure obtained a posteriori. This is also an output of the executed step, so it cannot be used to make predictions.

The data merging ends with the saving of this extended table in a *.csv* file from which information will be extracted for each client and the techniques of correlation analysis, statistical analysis, features selection and finally machine learning will be applied.

At this time the features contained in the dataset merged are 33 containing information of cuts, steps, glass and type of glass used. The session information are not considered useful in the merged table and all the relative features regarding this event were deleted in the final reshape before saving the table. The saved features that will be used for machine learning training and prediction are therefore those in table 5.2.

5.3 Handle Categorical Features

It often happens that some features belong to certain categories, normally they are represented by a string or by one or more digits. In the latter case the numbers are not correlated by any mathematical link. For example, if there are entries belonging to three different categories (1,2,3)

Feature	Type
CutLength	int64
Grind_MC	int64
Cut_MC	int64
TSup_MC	int64
TInf_MC	int64
Heating_MC	int64
Detach_MC	int64
IsDiagonal_MC	int64
BadTerminated_MC	int64
GlassNumberInSession	int64
StepNumberInGlass	int64
GrindEnabled	int64
StepType	object
Spess1_sup	float64
Spess2_inf	float64
SpessPVB	float64
Pmax_sup	float64
Pmax_inf	float64
parWheelBreakoutInf	float64
parBreakoutInfTrim	float64
parBreakoutSupTrim	float64
NGlassInOpt	float64
NGlassInSess	float64
GlassLength	float64
GlassHeight	float64
GlassTotCut(mm)	float64
PiecesInGlass	float64
WastePiecesInGlass	float64
TimeFromLastSession	float64
TimeFromLastGlass	float64
TimeFromLastType	float64

Table 5.2: Features after data merging, the first phase of the preprocessing phase.

these categories cannot be provided as they are to the machine learning algorithm because the algorithm would tend to look for mathematical correlations between the numbers 1,2,3 (e.g. 3 is triple 1) that have no meaning between the categories that are only groups, which in fact could be identified by three letters A, B, C without changing the meaning. The problem of managing categorical features arises. The main techniques for

converting and extrapolating information from them in a mathematical form are now analysed.

5.3.1 Common methods

There are two main classes of categorical data: *nominal* and *ordinal*. In the nominal categorical data there is no concept of sorting between the various categories, i.e. no category comes before the other and it is not possible to compare them because they are substantially disconnected from each other (e.g. weather, musical genres,...).

Instead, in the ordinal categorical data there is a concept related to the sorting between the classes. An example are the sizes of clothing (S,M,L,XL). These are related to the size of the garment but it is not clear whether linearly, polynomally or otherwise.

It is therefore necessary to transform these categories into numbers in order to be able to process the data. This method of transforming categorical features is called *encodings*. Now we will analyze the methods of encoding most useful in machine learning applications, the so-called *classic* encoding. Remember that there are various types of feature encoding that will not be analyzed here.[1]

One-hot encoding

This is the most used encoder for machine learning because it is very reliable and performing. You get a total scrolling of the categories without loss of information but it is not a good solution if the number of categories per features is high. It works by transforming the m categories into m binary features, so that for each example of the dataset only one of these newly created m features will have value one, the one related to the categoryof the selected entry. The others instead will all have zero value. In this way there is a total separation of the categories with a numerically comprehensible representation of them for the machine learning algorithm.

Dummy coding scheme

Very similar to the previous one with the difference that the m categories are transformed into $m-1$ features where the m-th category is represented

[1]The encoding techniques not presented here are the so-called *contrast* encoders and the *bayesian* encoders. A good Python library in which to find implementations of these encoding types is called *category_ encoders*.

when all the $m-1$ features have null value, all the others are instead identified by a 1 in the relative feature (as for one-hot encoding).

Effect coding scheme

Practically equal to the dummy coding scheme with the only difference that the category previously indicated by the zeros in all $m-1$ features is now represented with all values equal to -1 in the $m-1$ features.

Bit-counting scheme

The previous encoding methodologies work very well when the number of categories is small but begin to become problematic when the number of categories rises because too many new binary features are created. The problems that can emerge if the number of categories is high and one of the previous methods is used are related to storage, computational training time and dimensionality of the dataset. In fact, if the number of examples becomes comparable to the number of features, overfitting problems are created in the machine learning model (this possibility is called in jargon *curse of dimensionality problem*).

If you have many categories the bin-counting scheme represents a good encoding to avoid the problems mentioned above. It is based on the assignment of a probability to each category based on historical occurrence. It is clear that for this type of encoding we need datasets containing all the categories in order to calculate true probabilities.

Binary encoding

Immediately each element of the categories is transformed into a number if it was not yet, then this decimal number is transformed into binary and the various bits that make it up are separated one by one going to create each a new feature, so as to create a series of unique columns for a certain bit. This encoding creates a limited number of new columns and is therefore also indicated if the number of categories is high. It is more efficient than one-hot encoding both at the computational and storage levels, but the performance obtained is lower as there is no complete separation between categories. In fact, equal bits do not correspond to the same number but there is the risk that the algorithm of machine learning seeks erroneous associations between these elements. It therefore represents a compromise between efficiency and performance.

Hashing encoders

This last encoding algorithm is similar to one-hot but with fewer columns created and some information lost. It is based on the *hashing trick* [2] concept. The new number of features created is established a priori and therefore you can check the growth of the dataset before applying the encoding algorithm.

5.3.2 Application on the case study

After creating the extended table of cuts as explained in the section 5.2 it is needed to manage the categorical features. The only column represented by categories is *StepType*. The *StepType* represents the type of cut made that, remember, can be *StepVsxTaglio5X8* or *StepVsxTaglio-Diagonale5X8*, these two strings represent the categories of the feature *StepType*.

There are only 3 machine that send data in the format described in the chapter on the database (chapter 3), the *ProcessorID* that represent the machine can seem a categorical feature but, as we will see, studios will be kept separate for each machine because of their diversity. This means that in every algorithm training the *ProcessorID* will be unique for each machine and will be eliminated from the treatment, so it is not necessary to treat it as a categorical feature.

Since the categories of *StepType* are few we will use the classic one-hot method that works well when you do not have too many categories to divide. It can be implemented directly with the *SKLearn* machine learning library but it is easier and more intuitive to use a library dedicated to encoding called *category_encoder*. The use is simple, you create the object capable of encoding thanks to the class *OneHotEncoder* passing the names of features to be managed. Then the *fit_trasform* method is applied to the dataframe to be modified. The method returns another dataframe with the addition of the newly created columns.

In table 5.3 you can see how the features are now increased after handling categorical features. Moreover the type of the categorical features was *object* and could represent a problem for many machine lerning algorithm. Now this is no more a problem thanks to the transformation in numerical features.

[2]For a clear and satisfactory explanation of the hashing trick see "Don't be tricked by the hashing trick" by Lucas Bernardi.

Feature	Type
CutLength	int64
Grind_MC	int64
Cut_MC	int64
TSup_MC	int64
TInf_MC	int64
Heating_MC	int64
Detach_MC	int64
IsDiagonal_MC	int64
BadTerminated_MC	int64
GlassNumberInSession	int64
StepNumberInGlass	int64
GrindEnabled	int64
StepType_StepVsxTaglio5X8	int64
StepType_StepVsxTaglioDiagonale5X8	int64
Spess1_sup	float64
Spess2_inf	float64
SpessPVB	float64
Pmax_sup	float64
Pmax_inf	float64
parWheelBreakoutInf	float64
parBreakoutInfTrim	float64
parBreakoutSupTrim	float64
NGlassInOpt	float64
NGlassInSess	float64
GlassLength	float64
GlassHeight	float64
GlassTotCut(mm)	float64
PiecesInGlass	float64
WastePiecesInGlass	float64
TimeFromLastSession	float64
TimeFromLastGlass	float64
TimeFromLastType	float64

Table 5.3: Features after handling categorical features, now all features are of int or float type, only the datetime is of object type.

5.4 Correlation analysis

Once you have combined the various events you need to have an idea of the relationships that may exist between the various features, this to

build a more reliable, lightweight and performing model. Formally, the degree of correlation is a statistical measure that indicates whether there are associations between two inputs.

5.4.1 In theory

In this section we discuss the mathematical process that leads to the calculation of this indicator's value. There are various methods of correlation, we will discuss three of the most widely used:

Pearson Correlation Coefficient : it is the most widely used technique to obtain the degree of correlation between two features. The value of the degree of correlation indicator varies between -1 and +1. Domain extremes represent high negative and positive correlation. The value 0 instead indicates that there is no correlation between the two parts which are therefore independent. Pearson's coefficient is a measure of the linear degree of association between variables that is supposed to be continuous:

$$\rho_{x,y} = \frac{Covariance(x,y)}{\sigma_x \sigma_y}$$

where $\rho_{x,y}$ is the Pearson Correlation Coefficient, $Covariance(x,y)$ as the name suggest is the covariance between the two features, σ_x and σ_y are represent the standard deviations of the two features x and y. **Important**: If the values of x or y are always constant, their standard deviation is null. This imply that the Pearson Coefficient is not computable as a zero appears at the denominator of the formula. The Pearson Coefficient will assume a NaN value.

For series of data that contain m examples the previous formula can be rewritten as:

$$\rho_{x,y} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{m}(y_i - \bar{y})^2}}$$

where x_i and y_i are the entries of the dataset.

Geometrically this correlation coefficient can be obtained from the graphs called *scatter plots*, which are nothing more than a graph of one feature as a function of the other. This coefficient represents the linearity of the data curve. To calculate it graphically we first need to draw a fitting line of the data in order to minimize the square of the distance between the fitting line itself and the data. Then calculate the scatter with respect to an axis and the fitting line. The ratio of

these spillages represents the desired coefficient. In other words, the variance of the data with respect to the goodness of the linear fitting designed is being verified. What has just been described is visible in the figure 5.8 where three examples of scatter plots with the best fitting line and the relative value of the Pearson coefficient are shown.

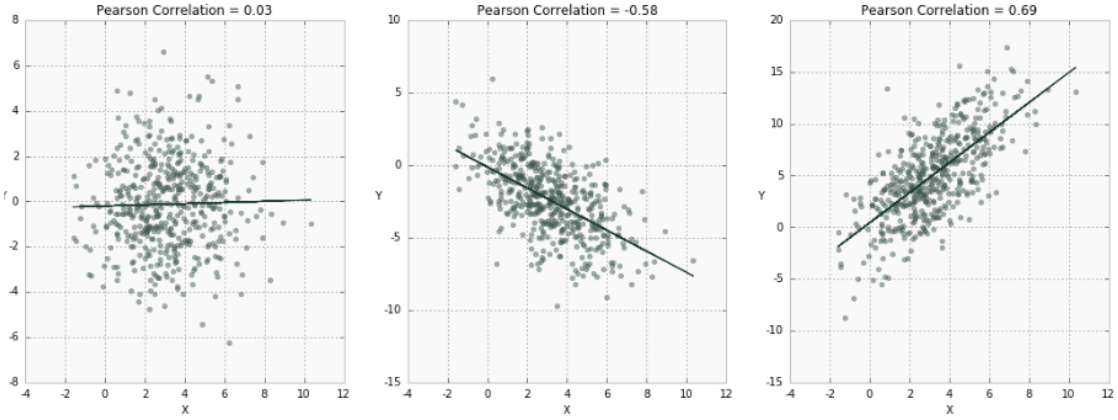

Figure 5.8: Three example of scatter plots and relative Pearson coefficient.

Spearman's Correlation : this index is a special case of the previous Pearson coefficient, applied to ordered variables. The relations between features can also be non-linear, but the measure obtained from this calculation is always referred to a monotonic association. This means that this correlation is more versatile than the previous one that was linear, but it has however big limitations as it is not able to calculate complex relations between data (like almost all polynomial relations).

The formula is the same as Pearson's but we have to input the ranks instead of the raw data.

There is a simplified version of the formula in case of unique ranks:

$$\rho_s = 1 - \frac{6 \sum_{i=1}^{m} d_i^2}{m(m^2 - 1)}$$

with $d_i = rankx_i - ranky_i$ and m is the number of example in the dataset.

Sometimes it happens that the degree of correlation changes depending on the correlation algorithm used. The message that must be leaked is that the degree of correlation may not be an exhaustive method of the relationship between two features, i.e. if you get a

zero correlation value there may still be association between features, especially if this association is not linear.

Kendall's Tau : it is a measure based on the concept of matching and discordant pairs in a dataset, where a pair is composed by two entries of features. In practice a set of m examples can be divided into $m\frac{(m-1)}{2}$ possible combinations of pairs, each of which can be identified as matching or discordant pairs. Kendall tau is based on this definition:

$$\tau_k = \frac{ConcordantPairs - DiscordantPairs}{m\frac{m-1}{2}}$$

that is, we look at the similarity between pairs of the dataset and we normalize this value with the total number of combinations, so as to have in output a value between -1 and +1 as for the other indicators.

5.4.2 In practice

The degree of correlation is easily obtained in Phyton using Pandas dataframe thanks to the method *.corr()* which automatically calculates for each pair of features the degree of correlation according to the method given in input to the function. This method returns a dataframe of size nxn (where n is the number of features) containing the degrees of correlation. Using this structure it results that the correlation matrix is symmetrical, with maximum values equal to 1 on the diagonal. This is due to the fact that on the diagonal the algorithm is looking for correlations between a feature and itself, and obviously this leads to a perfect correlation.

To get a clearer idea of the situation you can place scatter plots next to the correlation matrix, so as to see a graphical representation of the relationship between the features, in fact the correlations are based on the distribution of data of the two features.

In figures 5.9, **??** and **??** it is possible to see the correlation matrix of the unified database, the three graphs are relative to the three methods described above to estimate the degree of correlation: Pearson, Kendall and Spearman. As we can see the matrices are very similar to each other, this is synonymous of a good reliability of the correlation. Because of the high similarity between the matrices obtained, from now on only Pearson's will be proposed, since it is the most used method for these analyses.

In the correlation matrices shown the bright red color indicates a value very close to +1, while the deep blue a value very close to -1, the grey

instead indicates that there is no correlation between the features. As we can see, there is a maximum correlation between some features, which can therefore be considered dependent. This conclusion allows to reduce the number of features to be given in input to the algorithms of machine learning since dependent features are not useful to perform predictions and train the data.

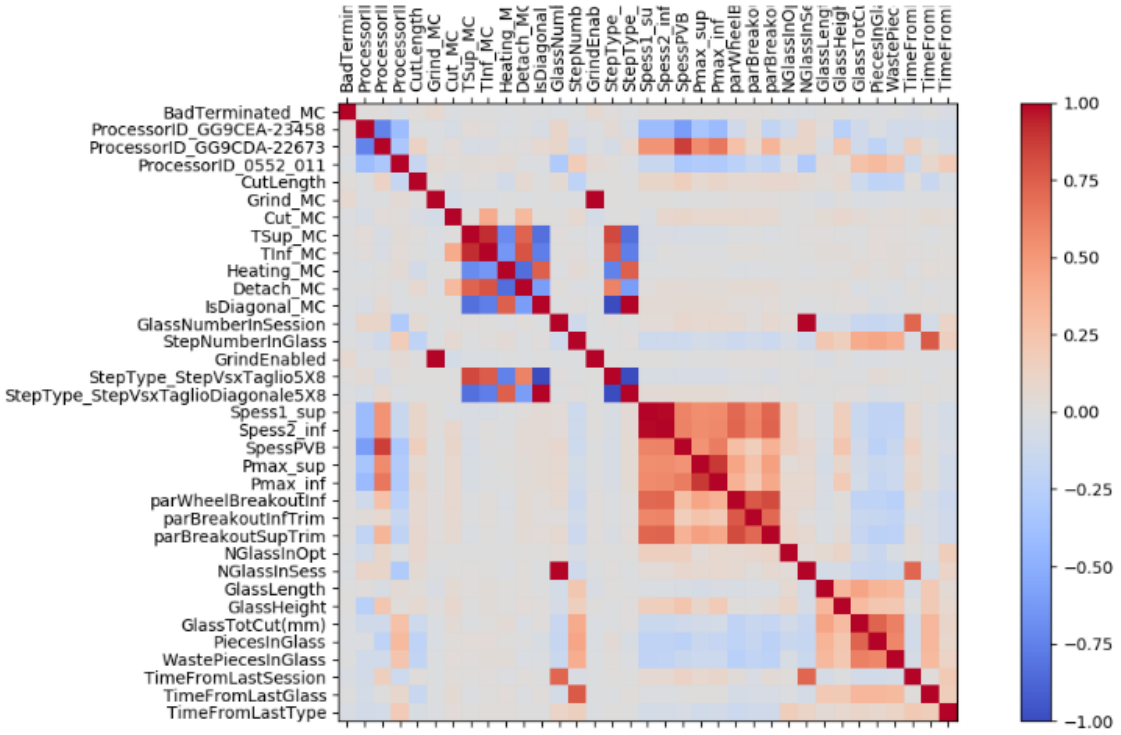

Figure 5.9: Pearson correlation coefficients matrix: there are no big differences among the other two correlation methods. For this reason from now on only Pearson will be used.

A very important detail to note is that for the output of the machine to be predicted (that is *BadTerminatedMC* in the table), there do not seem to be strong correlations with any of the input features. This is not good news and may define a *data mismatch* problem (see paragraph 4.1.4). A possible cause of this phenomenon is when the input dataset is not representative of the output you want to predict. This leads to poor results once machine learning techniques are applied, as there are no significant associations between input and output to predict.

On the other hand, there is some correlation between the various features and the *ProcessorIDNumeric*, which is an identification number of

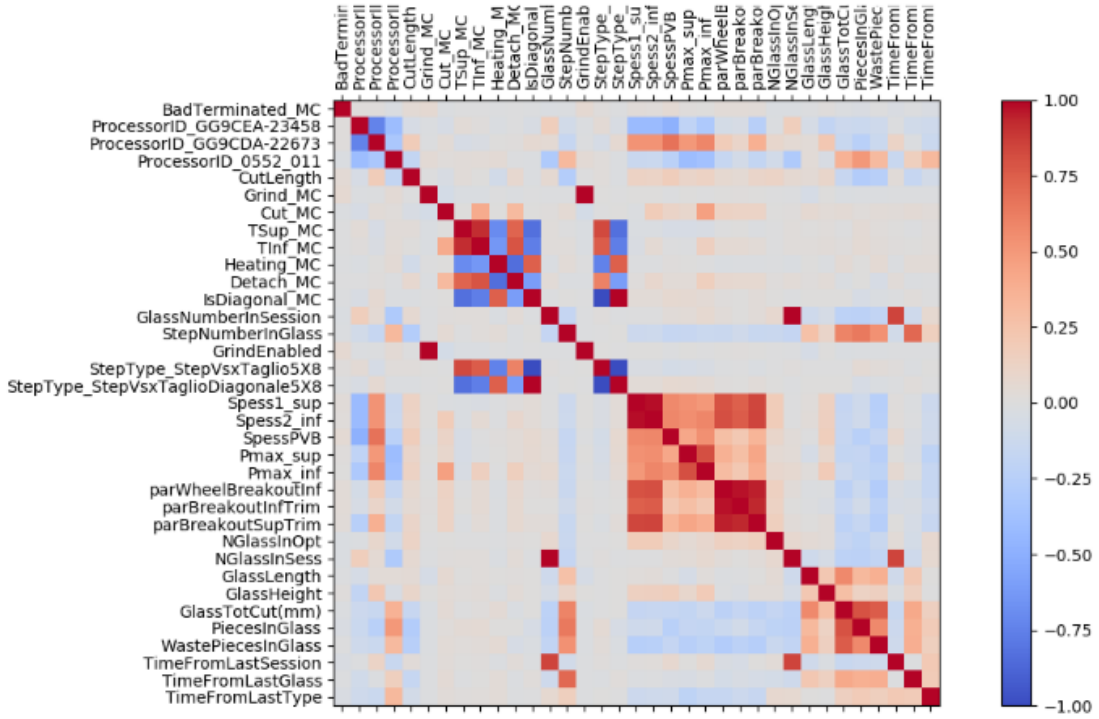

Figure 5.10: Kendall correlation coefficients matrix.

the *548 Lam* machines installed by the various customers. It is therefore useful to study in depth the correlations of each individual *548 Lam* machine, because as mentioned in the chapter describing the machinery, the *548 Lam* sold by Bottero are very customizable and the setup parameters are reconfigured on each of them.

NOTE: The format of the database described in this thesis and the sending of the data as described in the chapter 3 is currently valid (May 2019) for only 3 machines even if other machines are still connected to the database but send partial information and not updated due to an obsolete software version. The three customers in question will henceforth be named as *client1*, *client2* and *client3* to maintain confidentiality. The algorithm performance will be discussed only on their data.

Client1

The client1 has the machine on which you have the most interesting results with regard to correlation analysis. In figure 5.12 it is possible to see the correlation matrix of this client. You can see strong correlations between some features resulting from both cutting events and TypeOfGlass

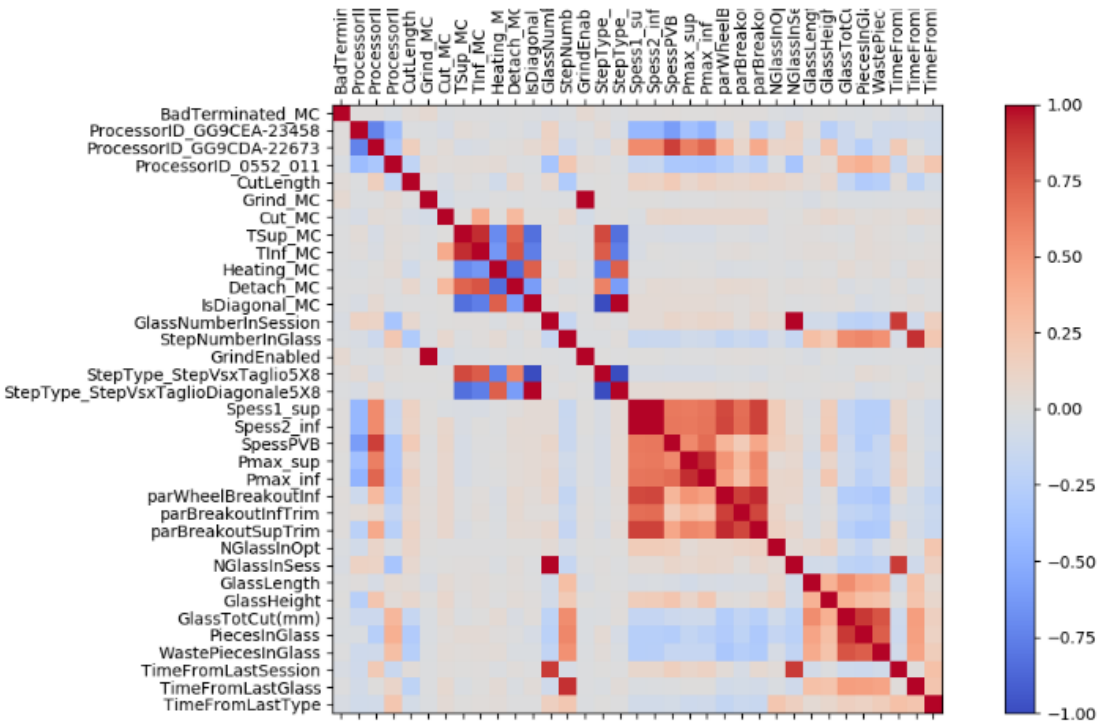

Figure 5.11: Spearman correlation coefficients matrix

events. In addition, the output *BadTerminatedMC* has slight correlations with other features, this may be an indication of a more accurate prediction. It is possible that sometimes some features are without correlation coefficient, the main reason is because those features have always the same value, so the variance is zero and it is not possible to calculate the correlation with other features. Basically, these columns with no variance are useless because they do not give information that can be used by automatic learning algorithms.

Client2

The correlation matrices for the second customer (figure 5.13 is also shown. From this you can see how the correlations between features are lower with respect to the first client, even the output to be predicted seems less dependent from the features.

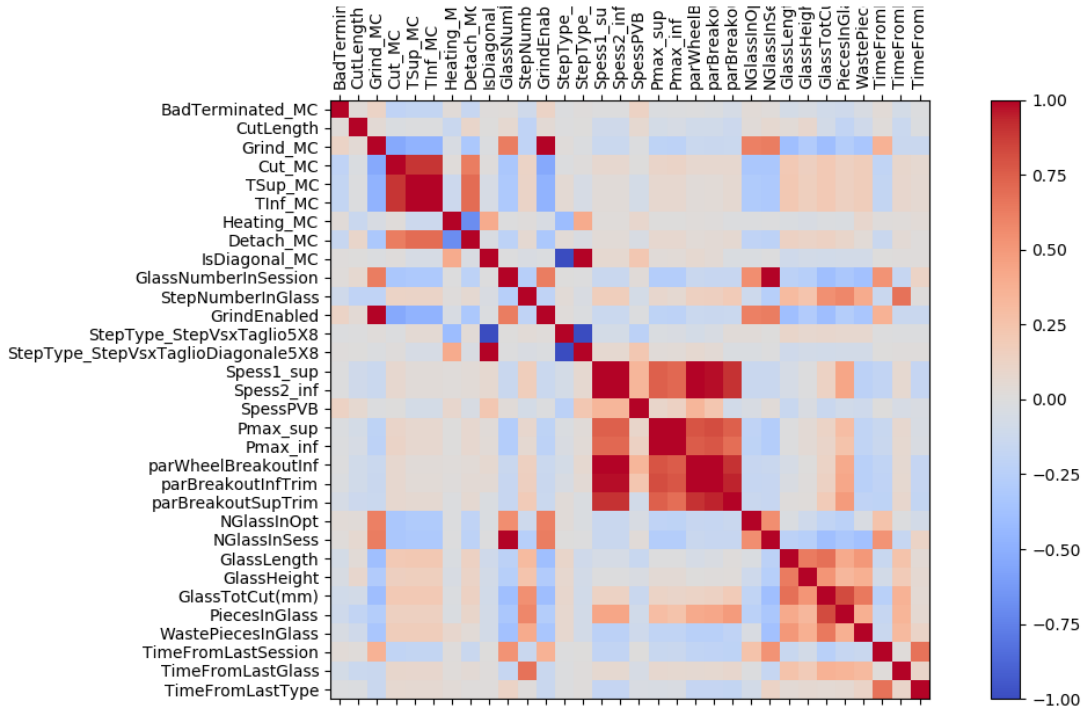

Figure 5.12: Correlation matrix for the first client .

Client3

The same matrice is also displayed for customer 3. The results shown in figure 5.14 are similar to the previous ones.

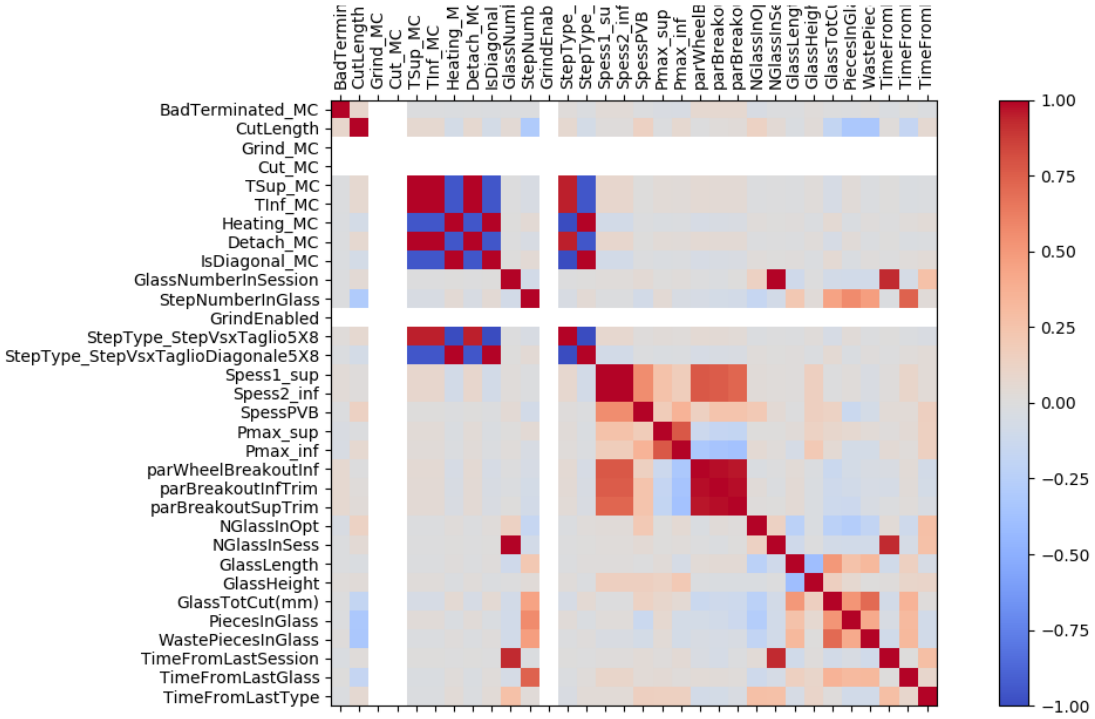

Figure 5.13: Correlation matrix for the second client

5.5 Statistical analysis

In this part there are presented some statistics. Usually I will refer to three separate clients of Bottero in order to show the difference among the various machines that send data in the database. This statistical analysis can demonstrate the possible need of a machine learning algorithm for each machine rather than a single generic predictor. The *548 Lam* as already said it is not a standard and uniform machine, so its parameters are customized and different.

As mentioned in the beginning of this chapter only automatic mode is selected because manual mode have a bug in the collection of data,described in paragraph 5.1.

5.5.1 Unbalanced dataset

First of all, it is necessary to quantify the number of errors that occur during the cutting phases. In the period considered (March-May 2019) the cuts recorded in automatic mode by the 3 machines under study were 15701, this number is also the number of entries (rows) in the extended

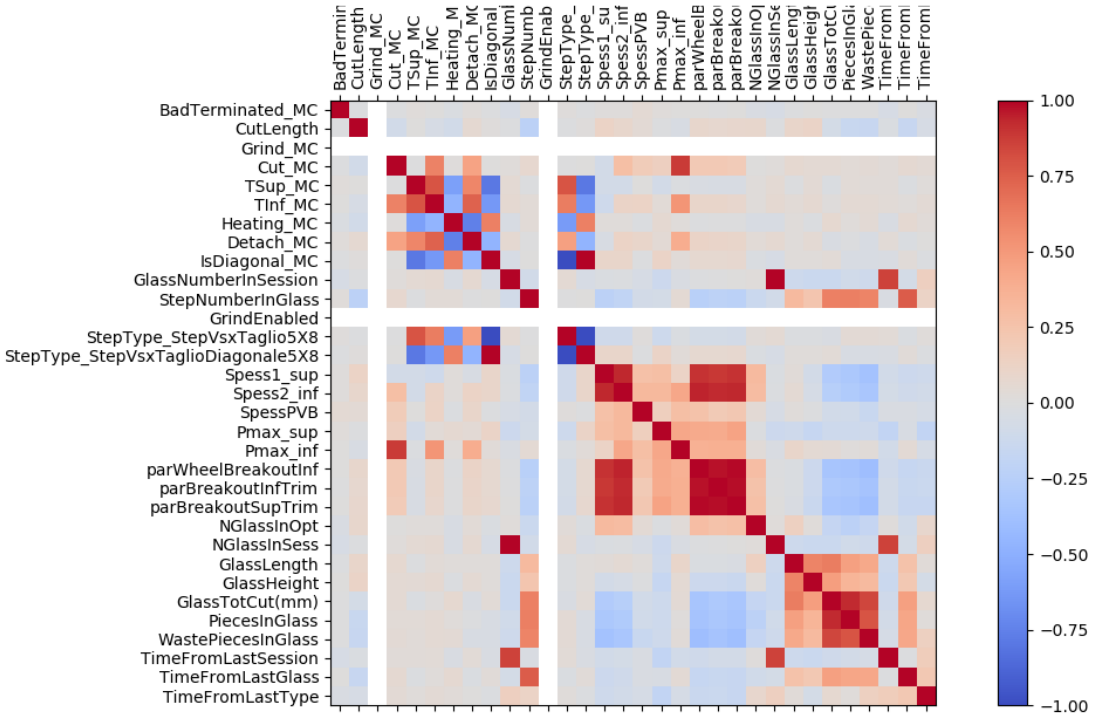

Figure 5.14: Correlation matrix for the third client

table of cuts previously created. Of these, 641 were badly finished, i.e. 4.1%.

It is possible to divide this data for the three customers in question so as to understand which machine has the most problems. The table 5.4 collects the number of outputs divided by customer and by cutting mode (that is by *StepType*).

Output		Only Automatic Cut	Diagonal Cut
Client 1	0	3142	14
	1	112	1
Client 2	0	5489	104
	1	251	2
Client 3	0	7459	46
	1	338	0

Table 5.4: Historic review of the output to predict divided by client.

In general, the number of badly finished cuts is always much lower

than the number of successful cuts. This fact indicates that the dataset under study is an *unbalanced* dataset. As explained in the paragraph 4.1.5 the best metrics to evaluate datasets of this type are f1 score, recall and precision.

5.5.2 Differences among customers

To evaluate the differences between the various customers the merged dataset of the cuts is divided into various subsets one for each machine and basic statistics are calculated for each of them. The average and the standard deviation of each dataset feature is given in the table 5.5. These two measures are useful to evaluate mathematically how much a machine is different from the others and how much a feature can be useful in the algorithm of machine learning.

Another way to understand visually and more easily the differences between customers is to plot a variable based on the number of occurrences. These graphs show the preferences and needs of one customer over another. An example is the length of the cuts shown in the figure 5.15. From this graph we can see that customer 2 is the only one who makes cuts up to about $6000mm$ long while the others reach up to$3200mm$. These values are not random but depend on the standardized measurements of the glass sheets.

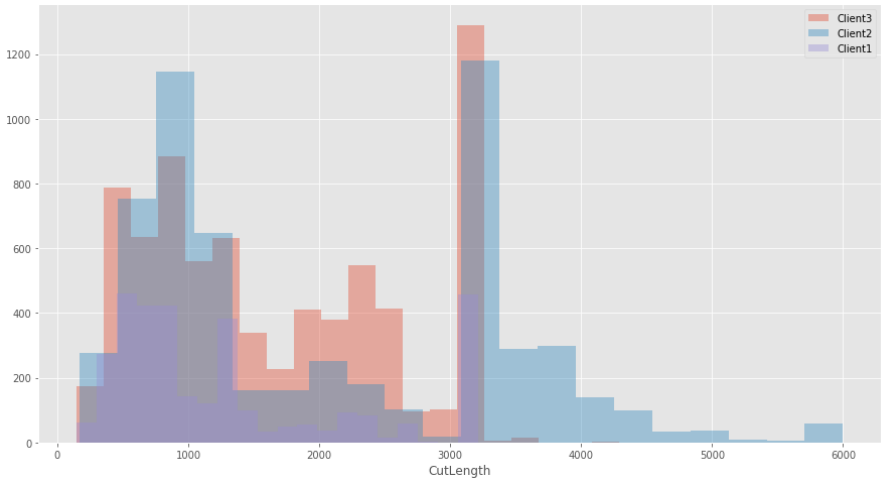

Figure 5.15: Cut length divided by clients.

A similar graph is the comparison between the different thicknesses of the cut glass. From the figure 5.16 it can be seen that customer 3 tends to use glass that is thinner than customer 2.

	Client 1		Client 2		Client 3	
	Mean	Std	Mean	Std	Mean	Std
CutLength	1316.42	935.56	2022.25	1330.95	1682.45	971.53
GrindMC	0.05	0.23	0.00	0.00	0.00	0.00
CutMC	0.98	0.13	1.00	0.00	1.00	0.07
TSupMC	0.98	0.14	0.98	0.14	0.99	0.09
TInfMC	0.98	0.14	0.98	0.14	0.99	0.12
HeatingMC	0.03	0.16	0.02	0.13	0.01	0.12
DetachMC	0.96	0.20	0.98	0.14	0.98	0.15
IsDiagonalMC	0.00	0.07	0.02	0.13	0.01	0.07
BadTerminatedMC	0.03	0.18	0.04	0.20	0.04	0.20
GlassNumberInSession	3.60	3.34	8.06	5.31	8.79	7.31
StepNumberInGlass	64.20	53.15	32.84	28.43	37.53	32.61
GrindEnabled	0.05	0.23	0.00	0.00	0.00	0.00
StepType_StepVsxTaglio5X8	1.00	0.07	0.98	0.13	0.99	0.07
StepType_StepVsxTaglioDiagonale5X8	0.00	0.07	0.02	0.13	0.01	0.07
Spess1sup	3.46	0.81	4.58	0.84	3.47	0.81
Spess2inf	3.46	0.81	4.58	0.84	3.45	0.85
SpessPVB	39.53	25.22	72.96	10.3	40.92	15.60
Pmaxsup	119.19	12.65	136.75	6.16	127.69	8.07
Pmaxinf	118.10	14.49	140.66	7.45	126.06	10.10
parWheelBreakoutInf	1.49	0.54	1.98	0.77	1.77	0.59
parBreakoutInfTrim	1.80	0.58	2.12	0.97	1.97	0.66
parBreakoutSupTrim	1.40	0.50	2.10	0.73	1.58	0.58
NGlassInOpt	2.22	1.60	2.54	1.98	2.27	2.16
NGlassInSess	3.60	3.34	8.06	5.31	8.79	7.31
GlassLength	5268.09	1308.59	5316.70	960.48	5187.61	1386.13
GlassHeight	3106.01	406.69	3312.86	567.64	3078.44	389.84
GlassTotCut(mm)	44836.92	18412.41	31714.61	9019.60	32655.17	15250.60
PiecesInGlass	26.65	16.55	12.07	8.53	14.08	7.88
WastePiecesInGlass	11.00	5.77	7.14	3.88	7.40	4.30
TimeFromLastSession	14420.29	10938.77	21798.93	14559.96	18841.33	21013.40
TimeFromLastGlass	1426.18	1541.67	1111.35	1008.40	1036.56	1014.77
TimeFromLastType	9081.26	9637.52	3570.15	3893.92	3869.39	4521.33

Table 5.5: Mean and Standard Deviation divided by clients for automatic cuts. The row represent the features.

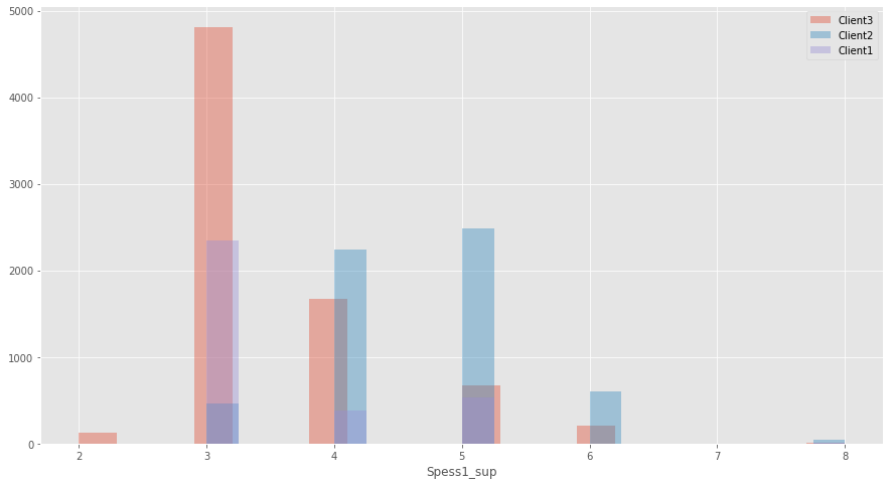

Figure 5.16: Thickness of cut glass divided by clients.

As far as the cutting parameters are concerned, the values of the maximum cutting pressures (in bar) for the three customers are shown in the figure 5.17. This is important for the machine as it can create a bad terminated situation if the step is unsafe because the glass can break or not truncate if the chosen value is incorrect. The differences are considerable but are somehow related to the thickness of the glass type chosen (see previous graph).

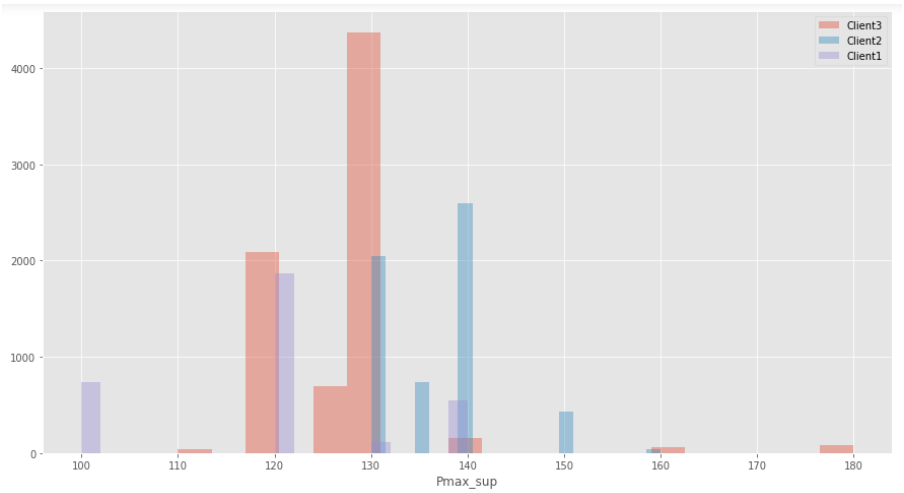

Figure 5.17: Pression used for upper truncation divided by clients.

5.6 Feature selection

The last step of the preprocessing phase is the selection of features. The output of this phase is a modified table that takes into account only the most important features for prediction. It is essential to provide machine learning algorithms with the right features in the correct format, this has some advantages:

- Better performance during training and testing, as the algorithm does not assimilate irrelevant information.
- Correlations between cause and effect are clearer, so it will be easier to understand how the algorithm works and on which parameters the prediction depends.
- Less training time and prediction because usually after this phase the total size of the dataset decreases.

There are several ways to classify feature selection processes. The most common is to divide them according to the process used:

Filter methods : they select features by sorting them based on their direct importance to the target or on the diversity of the inputs. They are based on statistical or correlation analysis.

Wrapper methods : generate subset of features that are trained with some machine learning algorithms. The best subset is selected based on the performance achieved.

Embedded methods : are a combination of the two previous methods.

A different categorization to features selection is defined by the degree of automatism of the process and the amount of work required to obtain the result. It is possible to distinguish between:

Manual Features Selection [3]: requires great knowledge of the dataset, this method consists of several distinct phases based on data analysis. it is very versatile but slow and complex. It is suitable for expert data scientists or for those who have a clear view of the problem.

Automatic Features Selection : it is faster than the previous one but in some cases it can give unexpected or worse results. It is especially suitable for beginners and those who do not have a clear view of the data to be processed or even when you have a very large number of features that would be difficult to manage manually.

[3]The methods cited here are a part of 12 methods identified by Vishal Patel in his speech: "*A Practical Guide to Dimensionality Reduction Techniques*"

Some algorithms of features selection divided according to the degree of automation required are now described, as they represent a substantial difference in practical implementation.

5.6.1 Manual features selection

This category includes all those processes that have to be imitated on their dataset in a different way according to their needs. To select the features with the methods listed below you need good experience of both the dataset and the work of data scientist. The following are generic concepts to apply to your case study with the algorithm that you prefer and that must usually be written ad hoc.

Percent of missing values

This is a study based on the amount of missing values that often represent a problem in data science, as they reduce the performance of many prediction algorithms. If certain features have high missing values percentages, they must be removed from the final dataset. In the case of smaller percentages, the importance of the feature must be evaluated on a case-by-case basis to understand whether it can be removed from the discussion or not. If it is not possible to convert the NaN value into a mathematically acceptable value that does not influence the parameterization of the algorithm.

Amount of variation

A statistical analysis of the variance and standard deviation from the average value of each feature can identify which data are almost always present in a constant manner. These data are usually of little use and indicate that a certain feature does not contain characterizing information to parameterize the machine learning algorithm. Before eliminating low-variance features without any problems, however, it must be ensured that there are no obvious correlations with the target.

Pairwise correlation

Based on the concept of correlation between features, this analysis compares the features in pairs. The pairs that obtain a great value (positive or negative) of correlation contain very similar information and therefore constitute a sort of redundancy that slows down the process of training and prediction by lowering performance. We must surely eliminate the pairs with maximum correlation value and evaluate all the others.

Correlation with the target

As the previous one is based on a correlation analysis but now the aim is to eliminate the features with low correlation with the target, because they are not very useful to correctly identify the output to be predicted.

Forward/Backward/Stepwise selection

Forward : It is based on the concept of the subsequent selection of the most promising features on the basis of a certain evaluation criterion decided a priori. The end of research method can be the achievement of a certain number of features or the maintenance of a certain standard.

Backward :Unlike the previous one, we start from a model with all the features and discard a feature by iteration, the least promising. Continue until the search constraint is met.

Pairwise :Alternate Forward and Backward selection by adding and removing a feature at each step until you get a stable subset of variables.

5.6.2 Automatic features selection

There are many methods to automate the process of selecting features, many are based on an optimal criteria to be met, this includes measures of evaluation metrics penalized by the number of features selected. In this way we try to reduce the number of features selected to maximize the value obtained. Other methods are based on the degree of correlation between features or between features and target. These automatic methods can lead to bad performances if not correctly used as they are based only on mathematical calculations made without considering the context.

In the automatic selection of features it is therefore a risk to lose useful information and thus decrease the performance of the system only to improve its efficiency.

The automatic selection of features becomes very useful when the number of variables is very high and it would be difficult to manage them manually.

There are countless algorithms of automatic features selection, here are briefly described those present in the library Python *SKLearn* used for this work. They are the most commonly used.

Univariate features selection

Select the best K features based on univariate statistics between each feature and the target output. Input features with high statistical relationship are selected.

Recursive features elimination

IT is a specific method of the library *SKLearn*, it is based on the training phase and the evaluation of its goodness thanks to an algorithm provided at the entrance to the method. At each iteration the algorithm does the training phase and eliminates the features less useful to the calculation of the output. This iterative process stops when you reach the number or percentage of features you want. The importance of features is calculated based on the coefficient that is associated with it during the training phase. Low coefficients indicate that the features are not indicative of the problem to be predicted.

PCA: Principal component analysis

This technique of reducing the dimensionality of the dataset has already been briefly discussed in the paragraph 4.2.7.

The PCA is not a technique that selects the most promising features but creates new ones by combining the characteristics of the original ones. It is therefore more correct to define it as a technique for reducing the dimensionality of the dataset. Apparently, the output of the PCA algorithm cannot be associated with the old features and the data obtained no longer has any physical meaning. The problem of the PCA analysis is that it is based only on the relationships between the features because it is a technique of unsupervised learning and therefore does not take into account the information coming from the output.

LASSO: Least Absolute Shrinkage and Selection Operator

This algorithm performs two main tasks: a feature selection and a regularization task. This means that in addition to selecting the most promising features so as to make the model clear and fast, it also helps to reduce the possible problem of overfitting.

It works by selecting constraints on the maximum value that the sum of the model parameters can reach. If this sum exceeds the preset threshold, LASSO applies a correction through a lambda regularization variable that decreases the value of the model parameters, so as to decrease the

sum below the threshold value. Only some parameters are kept, these will indicate the features to be selected.

Select From Model

It is another method specific to the *SKLearn* library. It works in a similar way to *recursive features elimination* so it can work with many estimators. *Select from model* is a little less robust as it only removes features based on a threshold without using any iterative algorithms.

5.6.3 Some algorithms implementation in the real dataset

To reflect what has been explained in the previous pages, both automatic and manual methods of fetures selection have been implemented. The first one that has been implemented is the process of manual features selection that requires the use of correlational and statistical analysis explained in the previous paragraphs. We will then show some results of the process of automatic features selection, not customizable and the output a bit cryptic and not always performing.

It should be added that in our case the feature selection process is not done so much to decrease the number of features and therefore the dimensionality of the dataset because the number of variables is relatively small. The feature selection aims to increase the quality of the data provided in input to the machine learning algorithm, so you should not expect a big increase in performance as much as an improvement in time and understanding of the algorithm.

Manual features selection

If only one customer is selected through the relative feature, the various ProcessorIDs, deriving from the separation due to the categorical features, can be eliminated as they are always constant for a certain customer, so that the columns shrink a little bit. If, on the other hand, only one table is considered for all customers, there are 3 more features, one identifying each customer. In each of the two cases the process of manual features selection followed is inclusive of 4 steps:

1. **Missing values**:With Pandas it is easy to see how many missing values there are in the dataset, just enter the code df.isnull().sum() and the list of features with its number of null values will be printed on the screen. In our case the null values are always zero: the tables used are without null values.

2. **Pairwise correlation**:As seen in the paragraph 5.4 you can get correlation values between features with the command *.corr()*. By analyzing the coefficients obtained and sorting them in descending order by degree of correlation, it is easy to identify the redundant features. The code written permitt to show the correlation between features in a decreasing order. A value of 1indicate they are perfectly redundant. It was immediately decided to eliminate this redundancy by deleting the features *Grind_MC, NGlassInSess, Spess1_sup, IsDiagonal_MC* that have perfect correlation with other features (as shown also in paraghraph 5.4.2). Other pairs have correlation value close to 1 but being our dataset unbalanced it is better to avoid drawing hasty conclusions and it is better to wait to evaluate the correlation with the target.

3. **Correlation with the target** : This type of analysis is also based on the correlation, not between features this time, but between the target (BadTeminated_MC) and the various features. Obviously you have to eliminate the features not very related to the target as they give little information on the desired output. You have to choose therefore a minimum threshold that has been fixed to 0.005 under which the features are eliminated. The threshold has been chosen very small because we have an unbalanced dataset and therefore a small correlation can still be important to find the when the target assumes value 1. Another reason for this choice is due to the fact that we have few features so in the choice of features is not necessary to remove much information as the database remains small. So you try to delete only those features completely useless, keeping those with little information.

4. **Amount of variation** :To be able to make a study on variance you need to do a data scaling. After that we use on the normalized dataset the method *.describe()* of Pandas that calculates statistics divided by features and returns them in a small dataframe object. It is now possible to sort the object obtained in descending order of standard deviation. As you can see in figure 5.18 the trend of the decrease of standard deviation as a function of the features is linear and has zero value only for two features. this means that they always have the same values. They are StepManual_C (the manual steps in fact are no longer considered in this treatment as described in the paragraph 5.1) and the ProcessorID is a certain customer if you consider only that machine in the table.

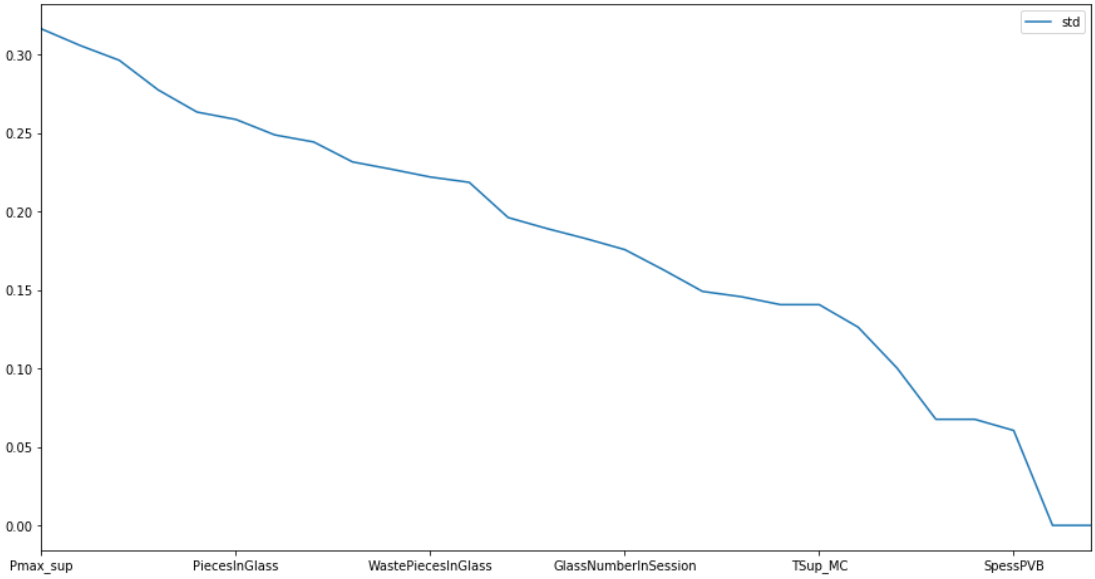

Figure 5.18: Normalized standard deviation reduction plot in function of related feature.

Automatic features selection

Of the various algorithms presented in the theoretical paragraph, 5 have been chosen for implementation, namely:

1. **Univariate Statistical Test (Chi-squared for classification)**
2. **Recursive Features Elimination**
3. **Principal Component Analysis**
4. **Features Importance**
5. **Select from model**

Their implementation is very simple and the code for the univariate statistical test is given as an example (figure 5.19). For the others the code is similar, just a few changes on the chosen feature selection algorithm are enough. For the PCA it is not possible to identify the chosen features because it creates new ones and you lose the physical meaning of the numbers so that the various columns will no longer have an eloquent index.

For these automatic feature selection methods, the number of maximum features to be selected can be inserted as parameter, for this thesis purpose a number of 30 can be optimal, thanks also to the considerations made during the automatic features selection. The 5 methods of automatic feature selection create five tables for each client and their effectiveness will be tested in the chapter on the evaluation of the best

algorithm.

```
#separo input/output
X = ml_table.drop(columns=['BadTerminated_MC'], axis=1)
y =ml_table['BadTerminated_MC']

X_new1 = SelectKBest(chi2, k=30).fit_transform(X, y)
X_chi=pd.DataFrame(data=X_new1)
```

Figure 5.19: Code for implement the univariate statistical test features selection.

Chapter 6

Error Prediction: Algorithm evaluation

In this chapter we continue with the error prediction, in detail after having prepared the data during the preprocessing phase in this chapter we evaluate the performance of the algorithms. A good result is desirable given the importance of the agrande problem to be solved, so as to avoid abnormal stops of the machine or pieces terminated badly. For further indication on the problem see chapter 1.2. The methods used for machine learning are described in the machine learning theoric chapter, section 4.2. The algorithms used are both classic and probabilistic.

We will describe, starting from the tables created specifically thanks to the procedure described in the chapter 5 concerning the preprocessing phase, how to divide the examples for the test and train phases, how to evaluate the algorithms concretely thanks to the evaluation metrics described in the paragraph 4.1.5 and finally we will discuss about the selection of the best solution, trying to understand, thanks to the final result, why the dataset at our disposal behaves better with one solution rather than another.

6.1 Training and Test sets and Normalization

The first step to apply machine learning after data preparation is the division of the dataframe into 2 sections:

- X contains all the features of the dataframe except the desired target to predict

- y is the column containing the target to predict

For example if our table after data preparation contained 30 columns, this step will reduce to 29 the inputs X and 1 column will be the output to be predicted y.

After this step it is possible to apply a normalization of the X train and test inputs through the *StandardScaler()* class. The standardization performs the following transformation on each feature:

$$z = \frac{(x - u)}{s}$$

with u the average of the feature and s its standard deviation.

Then data is divided between training and testing as explained in the chapter 4.1.1. To do this there is a simple function of the SKLearn library called *train_test_split()* that takes as arguments *X, y* and the test size i.e. the percentage of the database to be used for the test, default at 0.25. The method returns 4 tables X and y for the train and X and y for the test.

The problem of the train-test division is that of the randomness of the division between the two datasets, this involves different scores of the algorithm each time the dataset is divided because of the distribution between train and test set. Therefore, making with this classical division various tests if you often get results even very different from each other means that the prediction depends very much on the division of the initial dataset. This problem occurs especially in small and medium sized datasets.

Another more effective method is dividing the training dataset thanks to a process called cross-validation. This method divides the dataset into k parts and uses *k-1* for the train and the last one for the validation. Repeat this operation k times until each combination has been performed. This way the validation set will always be different and you will surely analyze all possible cases. For each training and prediction you will get different results from which you then calculate the average to get the final result. In SKLearn there is a function that performs just this operation called *cross_val_score()* to which you pass the parameter k and the algorithm to train.

In figure 6.1 it is shown the general process of training and testing by means of cross-validation.

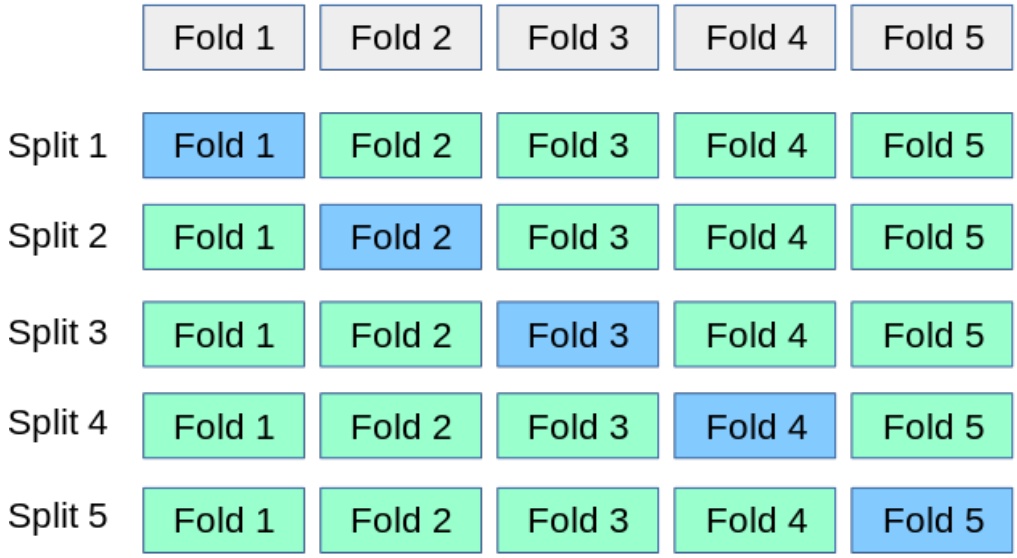

Figure 6.1: Process of training and testing with cross-validation

6.2 Evaluation Metric Choice

It remains to choose the method of evaluation and comparison of the various algorithms that will be tested. Since the prediction of errors is a problem of classification with unbalanced datasets, the best choice is certainly to use the recall and precision. Not having particular constraints or preferences between one and the other, it was decided to combine the two evaluation metrics in a single numerical datum that is the F1 score defined in the section 4.1.5. The F1 score is a number that varies from 0 to 1, where 1 is the best grade that the algorithm can get, in fact if F1 is equal to 1 it means that the algorithm finds all the machine errors without false negatives or false positives.

In practice this value is calculated on SKLearn using the *classification_report()* method which takes two inputs: y real output to predict and compares it with h the output predicted by the machine learning algorithm. The method calculates many values starting from these two columns as accuracy, recall, precision, F1 score, micro average, macro average. Alternatively, the F1 score can be the output of the cross-validation procedure that perform this operation automatically.

What has been done in this work is to compare the F1 score, obtained in one of these way, of various algorithms for each client to find the best algorithm for error prediction.

6.3 Hyperparameters tuning

In this section we select the machine learning algorithms among those available in the Python SKlearn library that will be compared in the next section. The set of algorithms chosen before training are:

- Random Forest Classifier
- Support Vector Machine
- Logistic Regression
- Neural Network
- KNN
- Naive Bayes Gaussian
- Gaussian Process Classifier
- Decision Tree Classifier
- Gradient Boosting Classifier

This group is based on the characteristics of the dataset in our possession where it was mainly taken into account to have an unbalanced dataset. An unbalanced dataset implies that the probabilistic models are the most suitable. Moreover, the correlation between the features and with the target are very slight, this is another factor that directs the choice towards algorithms based on subsequent choices and not on mathematical models staandard. In any case it has been decided to insert also simple deterministic models (SVM, Logistic Regression) and more complex (Neural Network) for completeness and comparison.

Hyperparameters tuning is the choice of the best and most efficient parameters for each algorithm, so that the performance in terms of prediction is maximum. Each algorithm has many parameters to modify, often the right combination is difficult to find manually, so there are automatic methods for choosing the best parameters.

There are mainly 3 methods for the automatic selection of hyperparameters:

Grid Search : perform a brute force search by trying all combinations and selecting the best one. If the parameters have an infinite range, select the search range. It can be very slow but always finds the best parameters.

Random Selection : random search of parameters up to a set limit of iterations. Definitely faster but less effective.

Bayesian based optimization : Efficient method that manages to improve the search for parameters at each iteration by understanding the search direction and minimizing time. Often it finds the optimal combination especially if you leave a random component that avoids the fall in local optima.

.

6.3.1 Grid Search

Since the datasets to be worked on are small, it was decided to use a deterministic approach to find the best parameters for each algorithm. To do this SKLearn provides a method called *GridSearchCV()* which executes all possible combinations of the parameters passed as input. By performing this search for each algorithm you will find the optimized parameters in the table 6.1. All the parameters not mentioned in the table are set as default.

Algorithm	Modifies parameters
Random Forest Classifier	max_features='log2', n_estimators= 80
Support Vector Machine	C=3, kernel='poly',degree=3
Logistic Regression	C=5, penalty='l1', tol=0.001, class_weight='balanced'
Neural Network	activation='relu', solver= 'adam', hidden_layer_sizes= 500
KNN	n_neighbors= 7, weights= 'distance', algorithm= 'ball_tree', p= 1
Naive Bayes Gaussian	var_smoothing= 1
Gaussian Process Classifier	warm_start=True, max_iter_predict= 100, multi_class: one_vs_rest
Decision Tree Classifier	presort= True, splitter= 'random', criterion='entropy', class_weight='balanced'
Gradient Boosting Classifier	warm_start=False, loss='exponential', learning_rate=0.5, n_estimators=100,subsample=0.5

Table 6.1: Hyperparameters tuned after grid search.

6.4 Results for each client

Once the best parameters for each algorithm have been chosen as explained in the previous section, the performance can be evaluated for individual customers. As explained in the 5.6.3 section, both manual and automatic features selection were performed, so the performance of both feature reduction methods was evaluated. For automatic features selection, only the table of the selection that obtained the best results is shown.

The results were obtained thanks to the cross-validation method as explained at the beginning of the chapter. In practice, k=4 folds were used to obtain 4 sets of predictions from which average F1 and standard F1 deviation values were obtained. This evaluation method is more well structured than a single train-test division.

Client 1

As will be seen below, customer 1 is the one who gets the best score.

Now we will analyze in detail the scores obtained with the various algorithms in situations of different feature selection.

The first results discussed are related to Client 1 with manual selection of features (see paragraph 5.6.3). The values of the manually filtered dataset are visible in the figure 6.2.

Client 1 Manual features Selection					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,52	0,058	0,34	0,068	0,12	0,015
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,45	0,101	0,49	0,043	0,21	0,047
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,32	0,081	0,42	0,049	0,41	0,11

Figure 6.2: Client 1: in manual feature selection the Random Forest classifier is the one that reach the best results, while Logistic Regression is the worse. The standard deviation is quite low indicating stability of the results.

The second comparison carried out concerns the evaluation of the dataset where an automatic feature selection was carried out. In order not to report the results of each of the 5 automatic feature selection described in the paragraph 5.6.3 it was decided to report only the best results that in the case of Client 1 were obtained from the use of Principal Component Analysis and Chi-squared for classification obtaining similar results reported in fugura 6.3.

Overall, the best results are obtained with probabilistic algorithms based on the tree classifier.The result is a conseguence of unbalanced database property.

You can still achieve a 50% of F1 score, which is about half of the maximum score.

Client 1 Chi - PCA					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,48	0,094	0,35	0,038	0,11	0,012
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,43	0,088	0,41	0,075	0,17	0,007
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,34	0,056	0,38	0,045	0,36	0,053

Figure 6.3: Client 1in automatic feature selection. Also in this case Random Forest and Logistic Regression are the best and worse algorithhm. After automatic features selection the performances degrade a little bit and the solutioni is more instable.

Client 2

The same division based on the features selection was made for the second client, generally obtaining lower scores. This fact had already been anticipated in some way by the correlation analysis (paragraph 5.4) which identified lower correlation values with the target for the second client compared to the first one. In practice this minor correlation may mean the need to have more data or a mismach dataset.

Also in this case customer 2 is evaluated after the manual feature selection process, results in figure 6.4.

Then, the best scores for the automatic feature selection are shown in the figure 6.5. The features selection that got the best results in this case are Features Importance, Chi-squared and Recursive Features Elimination all with a score around the one shown in the figure.

As anticipated and as shown in the summary tables the results for customer 2 have worsened quite a bit. Probabilistic algorithms are always the ones that get the best results, but this time, in addition to those based on trees, it increases the score of K Nearest Neighboard, an algorithm based on the percentages of similarity between outputs.

Client 3

Customer 3 obtains very similar results to customer 2 and therefore quite distant from customer 1. Again, there is a dependence between these poor results and the correlation analysis of this customer, which was similar to

Client 2 Manual features Selection					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,32	0,026	0,11	0,041	0,12	0,003
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,26	0,014	0,31	0,033	0,04	0,016
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,21	0,035	0,31	0,061	0,23	0,062

Figure 6.4: For Client 2 the manual feature selection scores bring to the same best algorithm as before but the performance degrade a lot. This time Naive Bayes is the worse algorithm.

Client 2 Chi-Rfe-Fi					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,27	0,037	0,15	0,011	0,12	0,015
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,19	0,047	0,26	0,023	0,04	0,019
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,19	0,049	0,23	0,022	0,21	0,042

Figure 6.5: Results of client 2 with automatic feature selection are very similar at those in manual feature selection.

that of the second.

As for the other customers, we proceeded to analyze the data related to the manual features selection, the results of which are visible in the figure 6.6.

Finally, the best processes of automatic features selection have been selected and give life to the results of figure 6.7. These processes are Chi-squared, Features Importance and Select From Model based on Random Forest variable selection.

Client 3 Manual features Selection					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,21	0,006	0	0	0,09	0,005
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,04	0,025	0,23	0,033	0,06	0,036
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,01	0,01	0,19	0,046	0,12	0,038

Figure 6.6: Client 3 in manual feature selection reach very low performances with KNN as best algorithm and SVM the worse that does not predict anything, there is here a clear model mismatch.

Client 3 Chi -Fi - Rf					
Random Forest		SVM		Logistic Regression	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,23	0,047	0	0	0,1	0,005
Neural Network		KNN		Naive Bayes	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0,01	0,01	0,23	0,022	0,05	0,014
Gaussian Process		Decision Tree		Gradient Boosting	
F1 mean	F1 std	F1 mean	F1 std	F1 mean	F1 std
0	0	0,19	0,031	0,11	0,055

Figure 6.7: Finally client 3 is evaluated with automatic features selection. Results are quite poor KNN and Random Forest reach the best performances, KNN with lowewr standard deviation, so it is a more reliable solution. SVM and Gaussian Process do not center the goal at all.

Probabilistic algorithms once again prove to be the most effective with this type of data even for the third customer. Only in one case does the KNN obtain slightly better results than the other algorithms, which are however very close in score. In this case other algorithms such as Logistic Regression, SVM, Gaussian Process, Naive Bayes are not at all effective and they do not succeed at all in centering the problem to be solved.

6.5 Notes on performances

The results shown in the previous section do not achieve high performance in absolute terms. However, since there is no deterministic method capable of analyzing the database and finding any error produced by the machine, I think that all the true positives found by the algorithm are still a satisfactory result because we are working in the right direction to effectively predict the errors of the 548 Lam machine. On the other hand you would expect better results from a complex, complete and long work like this.

A general speech should be made about the variance and the average of the results expressed. The value indicated in the previous tables is the result of an average carried out on various training tests and predictions. Several tests are necessary because of the scarcity of data available. When dividing the initial database into train and test datasets it is very important to distribute the examples in order to effectively predict the behavior of the machine. Having a few examples it happens that the variance between the different results obtained is high. It was decided to take as a result the average of the various tests, however, it is neded to take into account a normal variance of about +- 5%, with peaks of F1 score in the most extreme cases can reach peaks of +-10% (means that if 0.55 is the score of an algorithm for a given customer this can vary by making a new test in a maximum range of about 0.4-0.7 depending on the distribution of the examples of the dataset).

A further comment should be made on the comparison of the results obtained with manual and automatic feature selection. A first very important note is that the feature selection does not significantly change the final score in any of the cases examined. This means that there are certainly features that are not very useful to the prediction and that the useful features have always been kept within the database used to train the algorithms. Another consequence of the stagnation of the results is due to the fact that you have few features available. The feature selection is very useful when the columns of the tables reach a large size that affect the computational performance and therefore also the scores achievable. Having in the dataset about 30 columns at most the feature selection is often irrelevant or even worse if the deleted information contained useful data. This is connected with the fact that only the scores of the best automatic feature selection were reported, the others not reported probably eliminated useful information so as to obtain worse results.

Finally we can discuss why some algorithms often get useless results by not finding any correlation with the target (value 0 of F1 score). It can

happen that due to the fact of having a small and unbalanced database, some train or test datasets do not contain examples of machine errors inside them. If this happens in the train set the algorithm is not able to find correlations and will never predict 1. If, on the other hand, there are no negative examples in the test set, the algorithm will never predict true positive, obtaining null scores in this case too.

6.6 Algorithm Choice

From the results shown above it is clear that the algorithm that best predicts the errors of the 548 Lam machine is currently the **Random Forest Classifier**. As already mentioned, however, in general the probabilistic algorithms are the ones that perform best. This is typical of unbalanced datasets.

Another feature of our database is that it has few obvious correlations with the target, so a deterministic mathematical model cannot find linear links between the data to approximate the correct result, so this is the second reason why probabilistic algorithms work better.

You can do further studies now that you have the algorithm selected as best available. For example, you can search for the causes of the poor performance in the training set score, or you can better analyze the prediction by going back to the two evaluetion metrics that make up the F1 score that are recall and precision. These, unlike the F1 score, have a more concrete meaning and it is therefore easier to understand which are physically and practically the results obtained.

6.6.1 Random forest learning curve

You can study in more detail the behaviour of the Random Forest Classifier algorithm through the learning curves (see paragraph 4.1.4). As you can see, the cross-validation score curve is increasing as the number of examples in the dataset increases, while the training score curve is almost constant around the maximum value. This last fact probably indicates an overfitting as the value of the learning curve should decrease slightly. Another possible cause of the gap between the two curves is the lack of data. As you can see the validation curve is constantly increasing with the increase in the number of examples, you can therefore expect a further growth as the amount of data increases.

In figures 6.9 and 6.10 you can see the learning curves for customers 2 and 3. The comments are similar to those made for customer 1 with

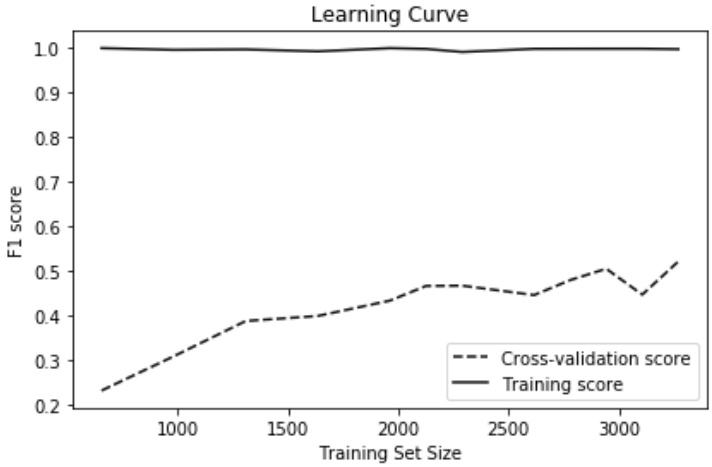

Figure 6.8: Random forest learning curve for client 1. In x axis the size of the dataset, in y axis the F1 score.

the difference that the scores are lower. Here too there may be a problem of overfitting or lack of data.

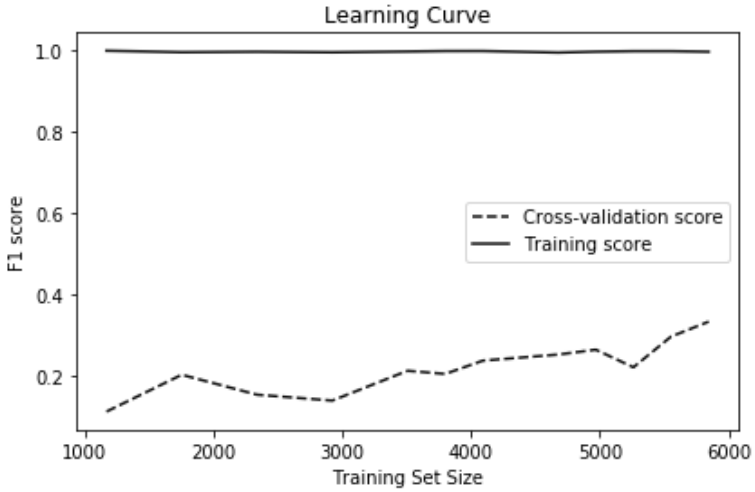

Figure 6.9: Learning curve for client 2

A further possibility is due to the uniqueness of the error situation, so you can train an algorithm to predict only the errors for which it was trained, but it is difficult to find correlations with the test set. Surely in reality there are causes that lead to an error in the machine. When the available data (features) are not representative of the problem to be solved we speak of Dataset Mismatch already discussed in the paragraph 4.1.4. If the problem that causes a scarcity of results is due to a dataset

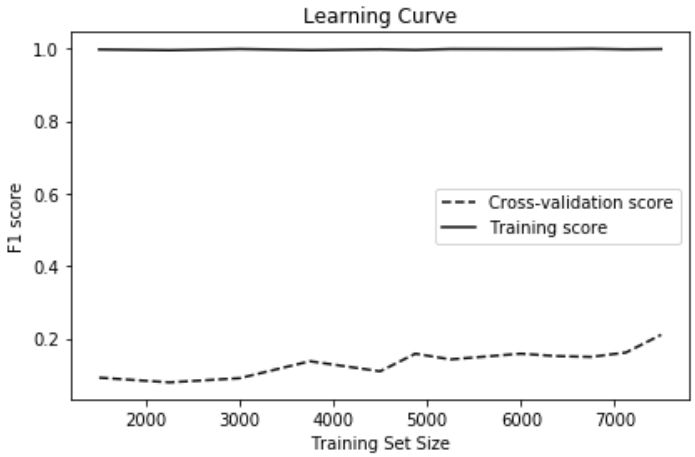

Figure 6.10: Learning curve for client 3

mismatch the algorithms of machine learning and their tuning can not overcome this problem. The best solution to get good results for the predictioin error would be in this case a new project, collecting new features and new data.

Chapter 7

Time Prevision Solution

This chapter discusses the second problem cited in the introduction, chapter 1.3, i.e. the prediction of plate processing times. The forecasting of processing times is a problem of great importance for Bottero, as customers often expressly request a time for processing. Since the machine has many possibilities of operation, indicating a time is not at all easy. Before this work they tried to create for some applications a deterministic approach that often turned out to be incorrect and that above all referred only to certain processes. In this chapter we will try to explain how it is possible to generalize the process obtaining satisfactory results.

This second part of the thesis has therefore a more practical approach and focuses not only on optimizing the results, but on the contrary its purpose is to create an algorithm that can be recreated directly on the machine so that before the processing of the glass plate begins customers know what time will be spent.

The time prediction is based on regression machine learning algorithms. The idea of using machine learning in this second work also stems from the need to generalize the solution quickly and reliably. Differently from the error prediction part we will not discuss so much the process that leads to the result (i.e. mathematical formulation, data merging, ...) but this second part is more focused on the structure of the practical work that must be performed for transport and ricreate directly the prediction platform to the *548 Lam* machine.

7.1 Requirement

Bottero is very interested in forecasting the cycle times of a glass plate. This in a perspective of a more complete work can lead to two main benefits:

- Prediction of working time on existing 548 Lam machines installed at various customers in order to make the time management of glass works more efficient.
- Giving a good estimate to possible new customers of the company who will use the values of the predicted processing times as a comparison between companies so that they can rely on this work to choose whether or not to buy a 548 Lam. The estimate of the time on non-existent machines can be carried out for similarity with already existing machines or it will be necessary to create in a second step a forecast of the times based directly on the features of the machine and not only on the inputs and outputs produced by the already existing machine. Remember that every 548 Lam is customizable and therefore making the time forecast on a single machine that does not yet exist is a very difficult task.

In this chapter, I will focus on the first of the two works just mentioned. The requirements to be met are:

- Improve the probabilistic approach already implemented for some machining operations.
- Extend the forecast to every possible processing (many steps, every glass thickness, every possible combination)
- Keeping the error within 10% without take too much in account the algorithm or the efficiency.
- Create an easily transportable and reproducible algorithm directly on the 548 Lam machine.

From these requirements it is clear that the objective is directed more to practicality than to optimization, which will then be an aspect to be improved once a functioning apparatus has been built.

7.2 Starting Data

The data available on the AWS Database are used to make the prediction:

- view_StepDetail (VIEW)
- TLamiWinDataToRtx548 (TABLE)

These two tables have already been mentioned in the paragraph 3.3. As discussed in this second part of the thesis the preprocessing phase is minimal because the starting database are these two tables already created by Bottero in function of this work.

The only part to do in data preparation is to associate the two types of information thanks to the step ID so you create a single DataFrame

containing a lot of information about the machine and the step whose execution time you want to predict. The result is a table called *StepAllInformation* that contains information from the two tables.

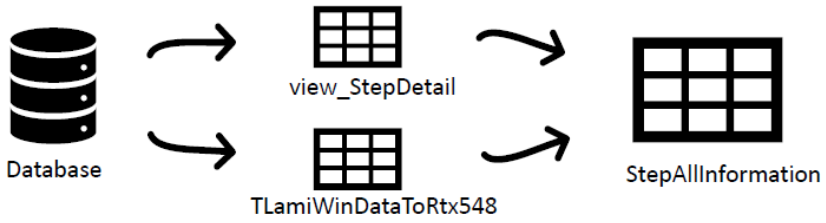

Figure 7.1: Data preparation phase for time prevision.

7.3 Idea

The idea to solve the problem described by satisfying the requirements is to use a Machine Learning algorithm that uses numerical input data to obtain the desired time. The inputs are all columns of the extended table *SteppAllInformation* except the output column to be predicted *ExecutinTime*.

The problem clearly falls within the part of machine learning called regression. There are various types of regression as seen in the paragraph 4.3 including:

- Linear Regression
- Redge Regression
- Lasso Regression
- Polynomial Regression

The common characteristic of these algorithms is that of finding suitable coefficients to multiply the inputs, thus creating the function of prediction:

$$h_c(x) = c_0 + c_1 * x_1 + c_2 * x_2 + \cdots + c_n * x_n$$

This function is very simple and that is why we will use one of these algorithms to recreate the prediction on the 548 Lam. For a first step you can also exclude Polynomial Regression again as it extends the number of features by creating interpolation between columns and making the prediction more complex to recreate.

The difference between the remaining algorithms consists in how the convergence to the values of the coefficients happens, in practice it changes the way in which the coefficients are chosen. In general, the coefficients are found thanks to the data available on the database: having a lot of data it is possible to associate the input x to the cycle times recorded during the various steps and saved on the database, this correlation is manifested through the coefficients.

Once the coefficients have been found, they will be saved in a **JSON**[1] file and it will therefore be possible to reconstruct the prediction function in the 548 Lam machine interface software when an optimization of a glass plate is selected.

As shown graphically in figure 7.2 the structure of this project is based on three main actors:

Database Used for collecting, formatting and organizing data and for creating the two tables described above for forecasting times.

PC with ML Performs the following tasks:

- Get the data
- Pre-processing and training
- Obtaining coefficients
- Validation of results
- Exporting information (JSON)

548 Lam Here the prediction function is reconstructed, the cycle time estimates are made thanks to the reconstructed function and from the machine there is a continuous sending of data to the database that will serve to create more precise algorithms.

7.4 Implementation

For the practical realization we used also in this case the Python programming language and the Jupyter Notebook environment for the development and debugging of the various parts to then create a final script that when launched executes a series of operations that will be explained

[1]Json (Javascript Object notation), is a type of format widely used for data exchange, based on JavaScript but its development is specific for data exchange and is independent of the development of the scripting language from which it is born and with which it is perfectly integrated and easy to use. Json has made his way through the various protocols and data exchange formats for ease of implementation.

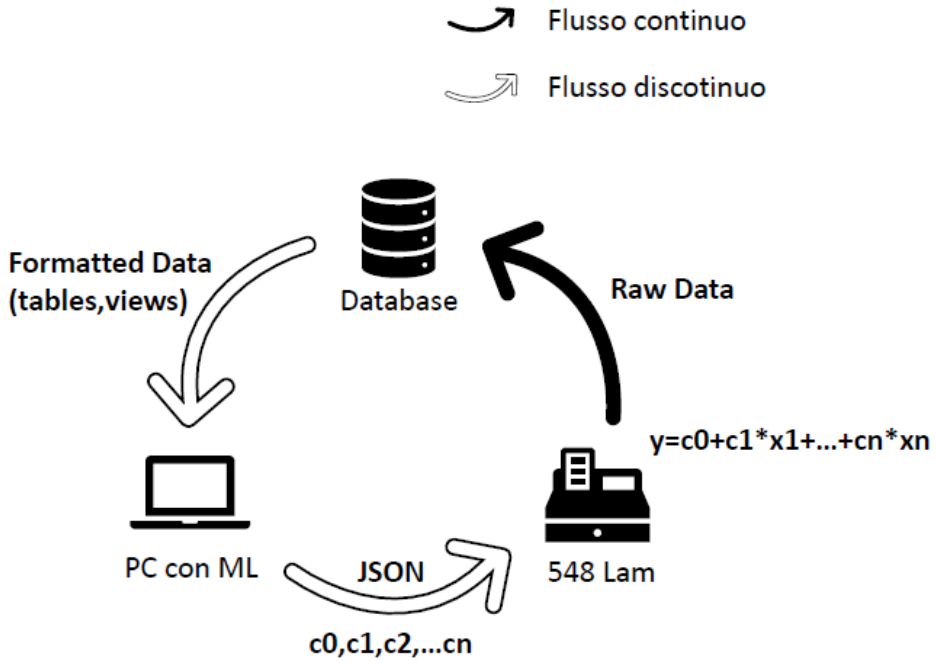

Figure 7.2: Data flow of time prevision: A PC selects data from the database and trains a machine learning algorithm to find the coefficients necessary for prediction. These are saved in a JSON file sent to the 548 Lam where the function is reconstructed and the forecasts take place. The 548 Lam continues to send data to the database from which the algorithm can be refined.

shortly generating at the end a file in JSON format containing the coefficients to reproduce the functions of prediction for each step.

The Python script does the following:

- reads as input a configuration file containing the names of the customers for which to perform the training of the algorithm and the subsequent generation of JSON files. Each JSON file is referred to only one customer.
- connects to the local SQL server where there is a copy of the AWS database and imports the formatted tables referring to the steps (view_StepDetail and TLamiWinDataToRtx548)
- Many columns not suitable for prediction are eliminated (usually all those with literal and/or redundant information)
- The badly finished steps are eliminated
- The two tables are merged

- Unified table is filtered to leave only selected customer data in the configuration file
- For each ProcessorID in the configuration file, the algorithm is trained as follow:
  - Search the various step to train (cut, transport,rotation,...)
  - For each of the step find before train a regression algorithm to make prediction for each step (so finding coefficients for each type of step)
  - Save coefficients: append in a dictionary the parameters
  - Repeat these actions for each step of the 548 Lam machine
- The results are saved in a JSON file and explanatory graphs are printed on the screen, such that it is possible to validate the results.

The structure of the JSON file is shown in figure 7.3. The structure is fixed: first an external structure contains 4 information (Client, MeanValueDataInputForClients, Datetime, Data) defined for the client to predict:

Client : ID of the client to predict.

MeanValueDataInputForClients : debug purpose.

Datetime : date and time of the JSON creation.

Data : recursive information are here stored. Contain sub-structures (one for each type of step) composed as follow:

MeanValueDataInputForStep :debug purpose.

Coeff : list of the coefficients $(c_0, c_1, \ldots, c_n)$

StepName : Name of the step

DataInput : List of ordered data input (features) to multiply the coefficients.

MeanExecutionTime : mean value of the historical data of the execution time for this step for that client. The scope of this list is to compare later the prediction done thanks to the machine learning process with the sum of these mean value to have an approximation of the working time of the machine.

Intercept : c_0 coefficient, is the first parameter who does not multiply any feature.

7.4.1 548 Lam implementation

When the coefficients have been obtained for each step and saved in a JSON file you can proceed with the implementation in the 548 Lam. The

```
{
    "Client": "03176_011",
    "MeanValueDataInputForClient": [325173.0162132621, 31.152173913043477, 0.2544935632742288
    "Datetime": "2019-07-17 14:04:42"
    "data": [{
        "MeanValueDataInputForStep": [324804.9069637883, 30.550974930362116, 0.28579387186629
        "Coeff": [0.0041236390919758255, 2.7217606245894026, 2605.1509759919736, 4.7931819517
        "StepName": "StepVsxTaglio5X8",
        "DataInput": ["GlassID", "NumberStep", "GrindEnabled", "GlassSessionID", "M.[ID]", "M
        "MeanExecutionTime": 34204.09400459578,
        "Intercept": -19407.609784361193
    }, {
        "MeanValueDataInputForStep": [324829.66758349707, 29.30884086444008, 0.31669941060903
        "Coeff": [-0.042073045244887246, -1.6243938094299222, 891.2895890415907, 1798.1842531
```


Figure 7.3: JSON file example:it is clear the data format inside the file

interface takes the data from the JSON file to reconstruct the prediction function. Once the optimization of the glass plate has been chosen (number and sequence of steps to be carried out), the various execution times of each step are predicted. Finally, these times of the individual steps are added together to obtain the forecast of the plate time. The *548 Lam* implementation is shown in figure 7.4.

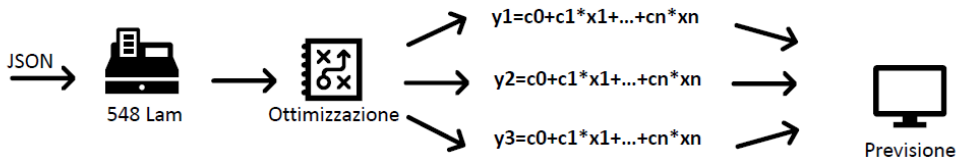

Figure 7.4: Implementation of time prevision inside 548 Lam: it read the JSON file and predict independently each step. Then the step time forecast are summed to obtain the final glass sheet optimization prediction.

7.4.2 Validation of the performance

Only part of the available data is used to find the coefficients. The remaining part is used to make forecasts for validation purpose. The validation is done comparing the result obtained with the forecast and the real one, which was hidden from the algorithm in the training phase. In this way you have the equivalent of new steps within a plate to predict. Finally,

the cycle times of each step belonging to the same glass slab are added together, thus obtaining the total slab time. In figure 7.5 is summarized this process.

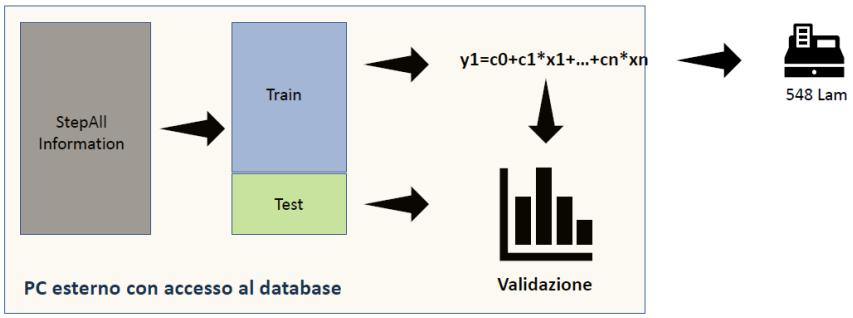

Figure 7.5: Validation process: before use the coefficients found in a real application inside 548 Lam the results are tested.

7.5 Results

This section shows the results of two customers following the proposed process. The results are related to the validation phase as explained above and not to the actual implementation on the 548 Lam that will be tested by Bottero.

For simplicity and confidentiality the two customers will be called Customer A and Customer B.

The graphs 7.6 and 7.7 contain the results of the entire process of processing a glass plate. The results reported in this section are those of the entire glass plate processing, composed by various step. Remember that the algorithm predict the steps and sum the individual results for obtaining the total glass processing time, that is the one we want to predict from requirements. The following results must therefore be read as the sum of various predictions made by algorithms with different coefficients for each step performed, predictions are then added together to obtain the final result.

Moreover, in the histograms are also reported the real measured values to which we tend. Those results are registered after the end of the glass processing phase and so they are the target we want to achieve.

Also the results obtained with another algorithm[2] that scrutinizes the average of the times of the steps (also reported in the JSON files for each step) is reported in the third column of the figures.

The axes of the graphs contain in the abscissa the *GlassID* (identification of the glass plate being processed whose times are to be predicted) and in the ordinate the predicted, average and true processing times measured in milliseconds.

Customer A achieves good results with an average error of 4.8% compared to the true value. We also report the average value of the error found thanks to the forecast made with the average time of each step, which in this case is about 12%. Results shown in figure 7.6.

Customer B also gets good results as you can see from the figure 7.7 reaching an average error on the forecast of 8.8% compared to about 10% obtained using the average values of the times of each step and then adding them up.

7.6 Further Applications

As said, this work of time prediction focuses on predicting the individual steps of each optimization and then adding them up to obtain the total time predicted for the entire processing of the glass plate.

This work has not been focused on maximizing performance but on simplicity of implementation, also to give a way to obtain some results that will lead to further studies and work. In this way it has been possible to create all the necessary apparatus for the prediction of time, with practical purposes and with real consequences for the company. Surely now that this first project has brought innovation thanks to the introduction of machine learning inside the 548 Lam machine, there will be further studies to improve the system presented here.

A limitation is that the process explained here is only suitable for existing machines, so it can be customized for any customer who wants to add this functionality to their machine. The heart of the matter is that

[2]The above algorithm is entered for comparison with the time prediction obtained with the machine learning algorithm. Instead of predicting each machining step, we look at the historical machining times of the steps and replace the prediction with the average of the times calculated from the data in the dataset. Then these averages for each single step are added up to obtain a sort of "prediction" based on the averages of the entire glass plate. This approach is evidently heuristic, it is reported to compare the goodness of a machine learning approach compared to this other approach more random and simpler that gives worse results.

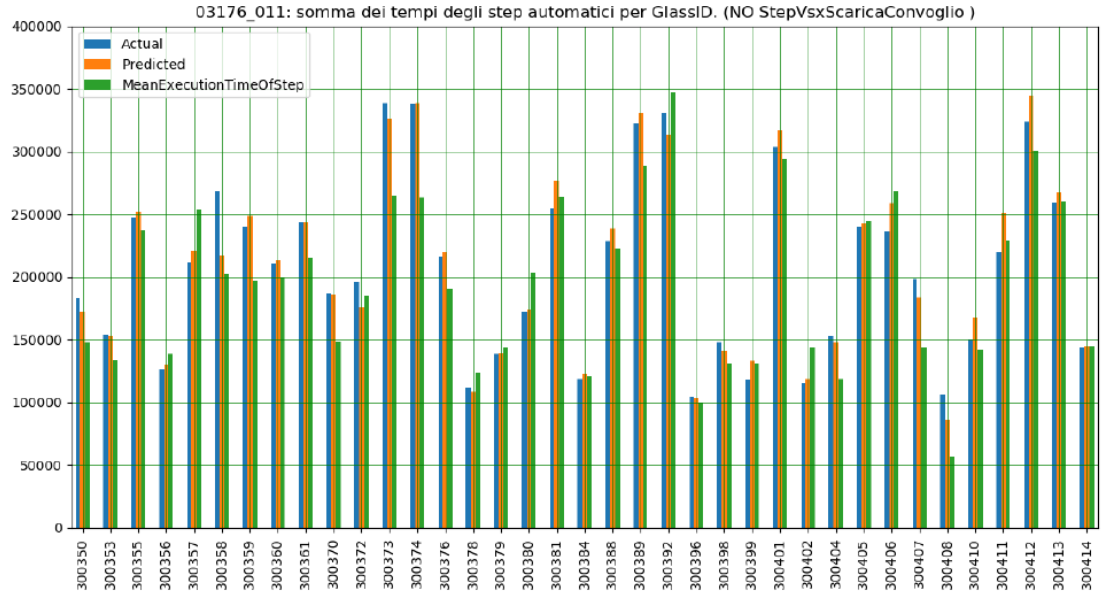

Figure 7.6: Client A time prediction results from validation process: in x-axis the GlassID of the glass sheet to predict, in y-axis the time in *ms*. The graph compare three columns for each glass plate worked. The three columns indicate actual values, the predicted one with the machine learning implementation and a comparison with the time obtained using the mean values of the step. the mean value is calculated on the basis of the historical data contained in the database. The prediction perform better than the mean value and reach an error of 4.8% wrt the actual values.

this process is based on the data that the machine has already produced and recorded, that is on a historical data, the database precisely.

What further studies and works would like to achieve instead is the prediction of the processing time of the glass plate for machinery not yet existing. The 548 Lam machine is a very customizable machine and can produce very different processes. The hope is to be able to correlate the prediction of working time with some features of the machine. Let's suppose that the machine can be composed of several parts A B C D E what we want to do is to correlate the parts with the speed of execution of the machine so that if a customer wants only parts B and C installed on the machine the working time can be predicted from this information. Surely we will rely on existing machines and the like, but the hope is a generalisation of the dependence between machine features and time prediction.

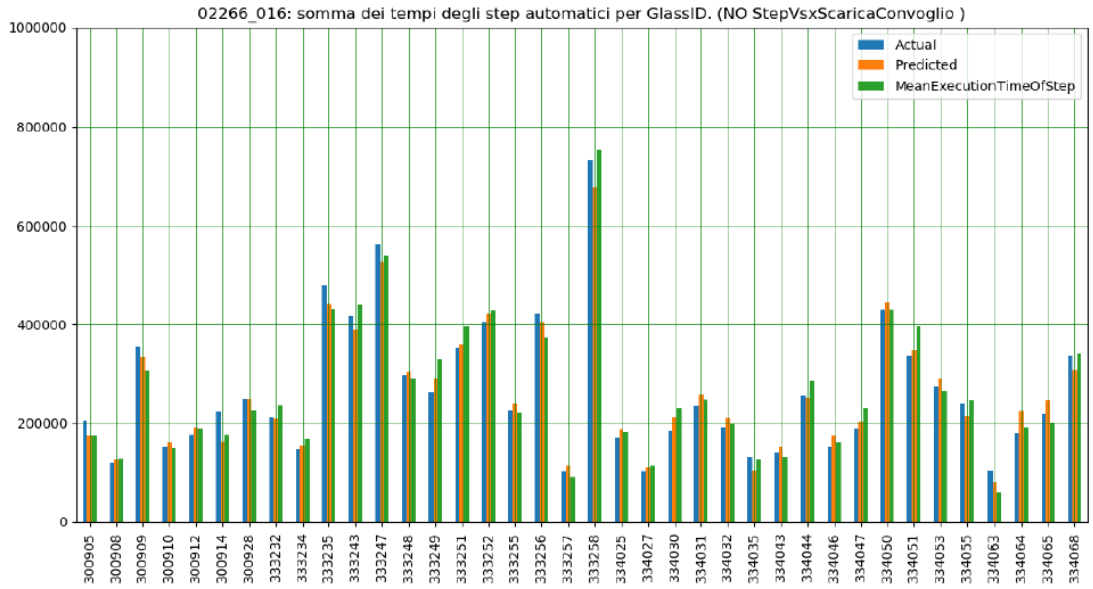

Figure 7.7: Client B results from validation process: in x-axis the GlassID , in y-axis the time in *ms*. Also in this case the three columns compare actual, predicted with ML and estimated with the use of the historical mean values. Also in this case the ML prediction perform better than the mean and reach the requirement staying behind 10% of error wrt the actual values.

Chapter 8

Final Observations and Comments

The proposed thesis focused on two distinct parts of work, the Error Prediction and the Time Forecast. The two parts as discussed in the previous chapters are very distinct, one more didactic and study, the other more practical and concrete. However, both are based on supervised machine learning, the first deals with a problem of classification, the other with a problem of regression. Inserting these two parts not only shows how to manage and deal with a machine learning problem but also shows how to implement it and have practical implications.

The first one has a more theoretical and more meticulous imprint on the problem, and has had as its pro the fact of being very accurate and of having tried to maximize the result. However, it is less concrete and more detached from the reality of use and from a possible implementation because of the greater didacticity. The main aim of this first part was not so much to achieve the result that would be difficult to implement, but to study and develop a complex and complete project of machine learning. The main output of this first phase is therefore to be found in the scrupulousness of the work and in the knowledge applied and studied during the months of the thesis. What I have learned and hope to have expressed in a clear and exhaustive way in this work, so that even a reader can learn is how to carry out a project of classification of machine learning, so in the thesis there are various ideas and quotations to parts related to the work but not strictly necessary. However, they have been included by virtue of a broader and more general view of the world of machine learning.

The second part, unlike the first, is almost entirely focused on the actual implementation of the idea carried out. The aim here is not to

achieve maximum results but to use this idea to obtain practical feedback from the company as well. The practicality obtained is highly correlated with the simplicity of the theoretical part of this second phase. The results, unlike the previous part, are vital because the study was done only by virtue of the application that must therefore be working. In this second part we can also see how machine learning can fit into and coexist in a historically deterministic environment such as that of a time forecast that has always been based on formulas and statistics. The result obtained shows how machine learning is suitable in cases of multitude of possibilities, where hundreds of formulas would be needed, one for each specific case, a quantity not manageable by hand because of the increasing industrial complexity.

Chapter 9

Appendix 1

In this appendix there are practical references to the work done in relation to the working environment, the programming language and the development method used.

9.1 Programming Environment

9.1.1 Jupyter Notebook

Jupyter Notebook is a free and open-source web application that allows you to create and share documents that contain code, equations, text and images incorporated and are offered educational insights that illustrate the potential of its use. With Jupyter it is, in fact, possible to combine the phases of data collection, code writing, visualization of graphs and tables as well as the possibility of making the code shareable on the GitHub platform (the most popular hosting service for software projects). In particular, the code is, from time to time, modifiable and executable in real time. What has been done can then be automatically exported in HTML, PDF and LaTeX format. In concrete, through Jupyter it is possible to unify moments normally separated from the work of a scientist until now.

The Notebook was used together for the first part of the thesis and for debugging in the second part because, when satisfied with the work, a normal executable was created.

Versions of the different parts of Jupyter used for the thesis:

jupyter 1.0.0
notebook 5.7.8
jupyter-client 5.2.4

jupyter-console 5.2.0
jupyter-core 4.4.0

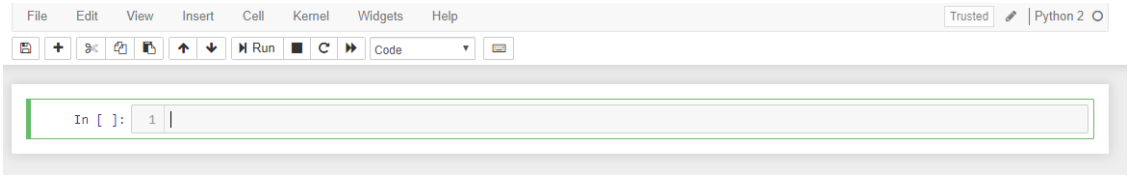

Figure 9.1: A typical interface of Jupyter Notebook

9.1.2 Python

Python is a high-level programming language, supports several programming paradigms, such as object-oriented (with support for multiple inheritance), imperative and functional, and offers a strong dynamic typing. It comes with an extremely rich built-in library, which together with automatic memory management and robust exception handling constructs makes Python one of the richest and most convenient languages to use.

Python version used for this thesis:

Python 2.7.15rc1

9.1.3 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations to manipulate numerical tables and time series. This tool allows the creation of data sets that can be manipulated by executing filters by column, row, order, join as well as SQL syntax.

The used version is:

Pandas 0.24.2

9.1.4 SKLearn, NumPy, SciPy

Scikit-learn is an open source library of automatic learning for the Python programming language. It contains classification, regression and clustering algorithms and support vector machines, logistic regression, Bayesian classifier, k-mean and many others. It is designed to work with the NumPy and SciPy libraries that are two libraries to manage complex numerical problem in Python.

The software version as follow:

SKLearn 0.20.3
NumPy 1.16.2
SciPy 1.2.1

9.1.5 Other libraries

MatplotLib

Matplotlib è una libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy.

Seaborn

Seaborn is a Matplotlib wrapper that simplifies the creation of common statistical graphs. The list of supported charts includes univariate and bivariate distribution charts, regression charts, and a number of methods for plotting categorical variables. Creating graphs in Seaborn is as simple as calling the appropriate graphics function. The style of the graph can also be controlled using a declarative syntax.

Category-Encoders

It is a library to automatically manage categorical features. It leans and is perfectly compatible with Pandas. There are various methods of categorizing more or less complex features.

Pyodbc

It is a library for establishing connections with SQL server. You can connect to the local or remote server. From Python with simple commands you can get all or part of one or more tables. It is also possible to insert an authentication or modify directly the data in the database without using the SQL language but programming in Python.

Jsonschema

Necessary to create, manipulate and validate JSON format.

The relative versions are expressed below.

matplotlib 2.2.4
seaborn 0.9.0
category-encoders 2.0.0

pyodbc 4.0.26
jsonschema 3.0.1

A typical import of the occurrent libraries and classes for machine learning

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier


from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score

from sklearn.utils import shuffle

%matplotlib inline
```

9.2 Structure of the work

This section briefly discusses how the work has been structured.

9.2.1 Error Prediction: Scripts Structure and Division

For the first part of error prediction 10 files were used in jupyter notebooks, the work was carried out in an orderly manner and follows the common thread presented in the chapters refpar:preproc and refpar:evaluation. The scripts are as follows:

- 1_Cuts_Extended.ipynb
- 1a_HandleCategoricalFeatures.ipynb
- 2_DivisionByClients.ipynb
- 3_CorrelationAnalysis.ipynb
- 4_StatisticalAnalysis.ipynb
- 5a_ManualFeaturesSelection.ipynb
- 5b_AutomaticFeaturesSelection.ipynb
- 6_HyperParameter_Tuning.ipynb
- 7_ML_Algorithms_Comparison.ipynb
- 8_MyLearningCurve.ipynb

The scripts ar enot reported here because of the big space requested to include all.

9.2.2 Time Prevision Script

For the second part of the weather forecast Jupyter Notebook was used at an early stage to then create a final script presented here.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from time import gmtime, strftime
import category_encoders as ce
import json
import pyodbc
from pandas.api.types import is_numeric_dtype

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import LinearRegression
#from sklearn.linear_model import Lars
#from sklearn.linear_model import BayesianRidge
#from sklearn.kernel_ridge import KernelRidge
#from sklearn.svm import SVR
#from sklearn.linear_model import SGDRegressor
#from sklearn.neighbors import KNeighborsRegressor
#from sklearn.gaussian_process import GaussianProcessRegressor
#from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

file=open('MyClientsToPredict.json', 'r')
my_clients=json.load(file)['clients']
file.close()


########################################################################
#funzioni utili dopo..
def bitwise_step(n):
    auto = 1
    manual = 2
    bad = 4

    mask=[0,0,0]

    if n & auto == auto:
        mask[0]=1
    if n & manual == manual:
        mask[1]=1
    if n & bad == bad:
        mask[2]=1
    return mask

def get_element(mylist,position):
    return mylist[position]


#......................................................................

########################################################################
#IMPORTO DA SQL LOCALE
try:
    print 'Importing StepDetail and TLamiWinDataToRtx548 from local SQL...'
    conn = pyodbc.connect('Driver={SQL Server};'
                          'Server=DESKTOP-JKHKFQL\SQLEXPRESS;'
                          'CustomersProductionStatistics;'
                          'Trusted_Connection=yes;')
    SQL_Query1 = pd.read_sql_query('''select * from
    CustomersProductionStatistics.dbo.viewStepDetails''', conn)
    SQL_Query2 = pd.read_sql_query('''select * from
    CustomersProductionStatistics.dbo.TLamiWinDataToRtx548''', conn)
    StepDetail1 = pd.DataFrame(SQL_Query1)
```

```
71          DataToRtx=pd.DataFrame(SQL_Query2)
72          print 'Tables imported.'
73      except:
74          print 'Impossibile connettersi al database SQL locale. Esco dal programma'
75          exit(1)
76
77
78
79
80
81      #faccio Merging dei due df
82      StepAll=StepDetail1.merge(DataToRtx, left_on='ID', right_on='StepID', how='inner')
83      #qualche operazione sulla tabella, estraggo info da maschera e elimino le colonne
        con troppi dati nulli
84      StepAll.rename(columns={'StepAuto':'StepMaskDec'}, inplace=True)
85      #extract information from the step mask (StepMaskDec), it create a new feature
        (StepMaskBit) containing a bit list
86      StepAll['StepMaskBit']=StepAll.StepMaskDec.apply(bitwise_step)
87      #create the features assigning the element of the just created list
88      StepAll['StepAuto_MS']=StepAll.StepMaskBit.apply(get_element,position=0)
89      StepAll['StepSemiAuto_MS']=StepAll.StepMaskBit.apply(get_element,position=1)
90      StepAll['BadTerminated_MS']=StepAll.StepMaskBit.apply(get_element,position=2)
91      #elimino le maschere perchè ho le appena creato creato nuove colonne con le stesse
        info estese
92      StepAll.drop(columns=['StepMaskBit','StepMaskDec'], inplace=True)
93      StepAll.drop(columns=['G.[Duration]','G.[TimeWorking]','G.[TimeWaiting]','parStacco.pr
        eriscaldamentoInit','parStacco.preriscaldamento','AllParam_NoDependences'],inplace=Tru
        e)
94      StepAll.drop(columns=['MachineID_x','M.[Name]','M.[TypeMachineID]','M.[NameExtended]',
        'G.[DateTime]','G.[DateTimeEnd]','G.[OptName]','G.[StringBottero]'],inplace=True)
95      StepAll.drop(columns=['T.[DateTime]','T.[Name]','DateTime_y','MachineID_y','Step','Pre
        vStep','Laminato','NextStep'],inplace=True)
96      StepAll.drop(columns=['ID_x','ID_y','StepID'],inplace=True)
97      StepAll['G.[Grind]']=StepAll['G.[Grind]'].apply(int)
98
99      ############### 0_Prendo dati dal 10 aprile
100     print 'step 0...'
101     t1=pd.to_datetime('04/10/2019')
102     StepAll['DateTime_x']=pd.to_datetime(StepAll.DateTime_x)
103     StepAll=StepAll[(StepAll.DateTime_x>t1)]
104     StepAll.drop(columns=['DateTime_x'],inplace=True)
105     StepAll.reset_index(drop=True,inplace=True)
106
107     ############### 1_Elimino Bad terminated
108     print 'step 1...'
109     StepAll=StepAll[StepAll.BadTerminated_MS==0]
110     StepAll.drop(columns=['BadTerminated_MS'], inplace=True)
111
112     ############### 2_ Elimino Step manuali
113     print 'step 2...'
114     StepAll=StepAll[StepAll.StepAuto_MS==1]
115     StepAll.drop(columns=['StepAuto_MS'], inplace=True)
116
117     ############### 3_Elimino step SemiAuto
118     print 'step 3...'
119     StepAll=StepAll[StepAll.StepSemiAuto_MS==0]
120     StepAll.drop(columns=['StepSemiAuto_MS'], inplace=True)
121
122     ############### 4_Gestisco valori nulli
123     print 'step 4...'
124     #print StepAll.isnull().sum()
125     #print 'e necessario gestire i valori nulli in questa parte'
126
127
128     ############### 5_Seleziono Clients
129     print 'step 5...'
130     print 'somma di tutti gli step divisi per macchina:'
131     print StepAll['M.[ProcessorID]'].value_counts()
132     flag=0
133     my_clients_restricted=[]
134     for ii in my_clients:
135
```

```
136         if ii in list(StepAll['M.[ProcessorID]']):
137             if flag==0:
138                 StepAllClients=StepAll[(StepAll['M.[ProcessorID]']==ii)].copy()
139             else:
140                 StepAllClients.append(StepAll[(StepAll['M.[ProcessorID]']==ii)])
141                 flag=1
142             my_clients_restricted.append(ii)
143 
144         else:
145             print 'non è possibile trovare il cliente tramite il processor id '#%s
                immesso. ' %ii
146             print 'Il processor ID puo essere sbagliato o non presente nella tabella di
                ingresso StepAll. Verificare.'
147             print '-------------------------------------------------------------'
148             print 'Il processo e stato arrestato, eliminare le fonti di errore e
                riprovare'
149             exit(1)
150 
151     #categorizzo i clienti selezionati per la tabella generale valida per tutti i clienti
152     ce_one_hot = ce.OneHotEncoder(cols = ['M.[ProcessorID]'],use_cat_names=True)
153     StepAllClients=ce_one_hot.fit_transform(StepAllClients)
154 
155     my_clients_restricted.append('AllClientsSelected')
156 
            ##############################################################################
            ##############################################à
157 
158 
            ##############################################################################
            ##################################################
159 
160 
            ##############################################################################
            ##########################################################
161         #parte 2
162 
163     for tt in my_clients_restricted:
164         #read the table
165         print '\n\n\n\n-----------------------------Working on client: %s
            ------------------------------\n' %tt
166         if tt!='AllClientsSelected':
167             my_table=StepAll[StepAll['M.[ProcessorID]']==tt].copy()
168             my_table.drop(columns=['M.[ProcessorID]'],inplace=True)
169         else:
170             my_table=StepAllClients.copy()
171     ##########GESTIONE VALORI NULLI DI DEFAULT
172         #controlo valori nulli. se ce ne sono esco dal programma. Bisogna gestire questi
            problemi nella parte di pre-processing: file A1_StepDetail_Preprocessing
173         if my_table.isnull().sum().sum()!=0:
174             print 'Rilevati valori nulli per il cliente: %s' %tt
175             print 'Si desidera convertirli in 0 (zero)? '
176             print '(inserire y e premere invio per la conversione, altrimenti inserire
                altro comando per uscire dal programma e gestirli a mano)'
177             scelta=raw_input('>')
178             if scelta=='y':
179                 my_table.fillna(0, inplace = True)
180             else:
181                 print 'Exiting from the program. The table of the client %s contains
                    Null (NaN) values. Manage this problem manually then re-try.' %tt
182                 exit(1)
183 
184     ##########GESTIONE DATI NON NUMERICI (int64 o float64)
185         #controlo di avere solo dati numerici int64 o float64
186         non_numerics = [x for x in my_table.columns if not (my_table[x].dtype ==
            np.float64 or my_table[x].dtype == np.int64)]
187         for a in list(non_numerics):
188             if a!='Name':
189                 print 'DataTypes di una o piu colonne diverso da int64 o float64,
                    controllare: %s' %a
190                 exit(1)
191 
192 
```

```
193
194         #Creao lista con step eseguiti da cliente e per cui si creano algoritmi di
            predizione
195         my_list=list(my_table.Name.value_counts().index)
196         print 'lista di step eseguiti dal cliente %s:  ' %tt ,my_list
197
198         #Creo dataframe vuoto con n colonne. Questo dataframe sarà utile per debug per
            risalire a predizione, step,..
199         Names=list(my_table.columns)
200         Names.append('NameStep')
201         Names.append('Actual')
202         Names.append('Predicted')
203         Names.append('MeanExecutionTimeOfStep')
204         Pred=pd.DataFrame(columns=Names)
205         Pred.drop(columns=['Name','ExecutionTime'],inplace=True)
206
207
208         #creo struttura dei vari file che verranno scritti in formato JSON
209         my_dic={'Client': tt, 'Datetime': strftime("%Y-%m-%d %H:%M:%S", gmtime()),
            'data':[], 'MeanValueDataInputForClient': None}
210
211         #immetto media input per cliente (non in base agli step ma è media globale degli
            input cliente) sono per alcune colonne (quelle di input) di Pred
212         col_temp=list(my_table.columns)
213         col_temp.remove('ExecutionTime')
214         col_temp.remove('Name')
215         temp=my_table[col_temp].copy()
216         my_dic['MeanValueDataInputForClient']=list(temp.mean(axis=0).values)
217         #per debug
218         canc1=list(temp.mean(axis=0).index)
219
220
221
222
223         #popolo il dizionario my_dic aggiungendo strutture dentro data. Ogni struttura
            tra parentesi graffe sarà relativa ad una tipologia di step
224         for i in my_list:
225             #struttura interna di data, che chiamo al singolare dato (relativo ad una
                singola tipologia di step)
226             dato={'StepName':i,'Coeff':None, 'Intercept': None,'DataInput':None,
                'MeanExecutionTime': None, 'MeanValueDataInputForStep': None} #Json di ogni
                step
227
228             #filtro la tabella selezionata nel ciclo principale in base allo step
                i-esimo (secondo ciclo)
229             df=my_table[my_table.Name==i]
230
231             #separo input/output
232             X = df.drop(columns=['ExecutionTime','Name'], axis=1)       #machine
                learning input
233             y =df['ExecutionTime']                                      #machine leaning
                output
234             #train dataset
235             X_train, X_test, y_train, y_test= train_test_split(X,y,test_size=0.25)
236
237             try:
238                 #select the ML algorithm
239                 regressor =Ridge()   #Lasso(max_iter=2000, alpha=2.0) # Ridge()
                    #LinearRegression()
240                 regressor.fit(X_train, y_train) #training the algorithm ==> può dar
                    problemi se abbiamo troppi pochi esempi
241
242                 #Riempio il file esterno con i dati (coefficienti e termine noto ) della
                    fase di learning. Metto anche le colonne da dare per effettuare la
                    predizione
243                 #avrò:     y=c0+c1*x1+c2*x2+....+cn*xn
244                 #To retrieve the intercept:
245                 dato['Intercept']=regressor.intercept_
246                 #For retrieving the slope:
247                 dato['Coeff']=list(regressor.coef_)
248                 #To retrive data input
249                 dato['DataInput']=list(X_test.columns)
```

```
250                 #others
251                 dato['MeanExecutionTime']=y.mean()
252                 dato['MeanValueDataInputForStep']=list(X_train.mean(axis=0).values)
253                 #per debug
254                 canc2=list(X_train.mean(axis=0).index)
255
256
257
258                 #implemento l'algoritmo e stimo y_test, poi metto y_test e y_pred in un
                    dataframe
259                 y_pred = regressor.predict(X_test)
260                 y_df=pd.DataFrame({'Actual': y_test, 'Predicted': y_pred, 'NameStep': i,
                    'MeanExecutionTimeOfStep': y_train.mean() })
261
262
263                 #concateno i due dataframe X_test e y_df e li concateno in un grande
                    dataframe prima creato chiamato Pred
264                 Pred=pd.concat([Pred,pd.concat([X_test, y_df], axis=1,
                    ignore_index=True)],sort=False)
265
266                 print 'Working on %s ...' %i
267
268
269                 #plotto i risultati ottenuti
270                 df1 = y_df[['Actual','Predicted']].head(50)
271                 df1.plot(kind='bar',figsize=(16,10))
272                 plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
273                 plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
274                 plt.title(tt+ ' ' +i)
275                 plt.show()
276
277                 #time.sleep(2)
278
279             except:
280                 print
                    '\n\n\n----------------------------------------------------------------------
                    ----------------------'
281                 print 'non possibile stimare lo step:' , i , '. Non verrà inserito
                    questo step per il cliente '
282                 print
                    '----------------------------------------------------------------------------
                    ---------------\n\n\n'
283             #infine aggiorno my_dic con l'output dell'appendimento del machine learning
                    in modo da poter utilizzare questi dati esternamente per effettuare predizioni
284             my_dic['data'].append(dato)
285
286             if canc1 != canc2 or
                    len(my_dic['MeanValueDataInputForClient'])!=len(dato['MeanValueDataInputForSte
                    p']) or dato['DataInput']!=canc1:
287                 print ("DataInput for mean and for prediction differ, check the problem.")
288                 exit(1)
289
290
291         #salvo file Json in un file esterno
292         json_name= tt + '_JsonForPrediction.json'
293         file1 = open(json_name,"w")
294         file1.write(json.dumps(my_dic))
295         file1.close()
296
297         print '\n\n\n'
298         print 'File Json savato per cliente %s' %tt
299         print '\n\n\n'
300
301
302
303         #creo e poi plotto il risultato totale ottenuto in un grafico, unendo tutti gli
                step con lo stesso GlassID (pedizione tempo lastra)
304
            GlassActual=Pred[Pred.NameStep!='StepVsxScaricaConvoglio'].groupby('GlassID').Actu
            al.sum()
305
            GlassPredicted=Pred[Pred.NameStep!='StepVsxScaricaConvoglio'].groupby('GlassID').P
```

```
        redicted.sum()
306
        GlassMean=Pred[Pred.NameStep!='StepVsxScaricaConvoglio'].groupby('GlassID').MeanEx
        ecutionTimeOfStep.sum()
307     Glass=pd.merge(GlassActual,GlassPredicted, on='GlassID', how='outer')
308     Glass=pd.merge(Glass,GlassMean, on='GlassID', how='outer')
309     Glass['DifferenceInSec']=(Glass.Actual-Glass.Predicted)/1000
310
        Glass['PercentageErrorPrediction']=(abs(Glass[Glass.Actual>100000].Actual-Glass[Gl
        ass.Actual>100000].Predicted)/Glass[Glass.Actual>100000].Actual)*100
311
        Glass['PercentageErrorMean']=(abs(Glass[Glass.Actual>100000].Actual-Glass[Glass.Ac
        tual>100000].MeanExecutionTimeOfStep)/Glass[Glass.Actual>100000].Actual)*100
312
313     df1 =
        Glass[['Actual','Predicted','MeanExecutionTimeOfStep']][Glass.Actual>100000].iloc[
        0:50]
314     print 'media errore cliente %s usando la predizione:  '
        %tt,Glass.PercentageErrorPrediction.mean()
315     print 'media errore cliente %s usando la media:       ' %tt,
        Glass.PercentageErrorMean.mean()
316
317     #plot
318     try:
319         df1.plot(kind='bar',figsize=(16,10))
320         plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
321         plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
322         plt.ylim(-500, 1000000)
323         #plt.xlim(0,10)
324         plt.title('%s: somma dei tempi degli step automatici per GlassID. (NO
            StepVsxScaricaConvoglio )' %tt)
325         plt.show()
326     except:
327         print 'impossibile plottare grafico totale tempi predetti/veri per cliente:
            %s' %tt
328
329
330
331 print
    '\n\n\n----------------------------------------------------------------------------------
    ---------------'
332 print
    '----------------------------------------------------------------------------------
    ---------'
333 print 'Tutti gli apprendimenti ML sono andati a buon fine. Tutti i file JSON con i
    coefficienti sono savati.'
334 print '\n\n\n'
335
336
```

Bibliography

[1] Bottero Glass Technology,*Laminated Cutting*

[2] Bottero Glass Technology,*548 Lam*

[3] Bottero Glass Technology (2019),*548 Lam: Linea automatica ad alte prestazioni per vetro laminato*

[4] Bottero Glass Technology,*Manuale Software: macchina di taglio automatica 5448 Lam*, sezione H *Uso del software*

[5] Alan Beaulieu (2009),*Learning SQL* ,Second Edition, O'Reilly

[6] Andrew Ng (2018),*Machine Learning Yearning*

[7] Murphy K.P. (2012),*Machine Learning: A Probabilistic Perspective*, The MIT Press

[8] Hastie T, Tibshirani R., Friedman J. (2017), *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*,Second Edition, Springer

[9] S. B. Kotsiantis, D. Kanellopoulos and P. E. Pintelas (2006),*Data Preprocessing for Supervised Leaning*

[10] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015),*Machine Learning for Predictive Maintenance: A Multiple Classifiers Approach*, Queen's University Belfast

[11] H. M. Hashemian, Senior Member, IEEE, and Wendell C. Bean, Senior Member, IEEE (2011),*State-of-the-Art Predictive Maintenance Techniques*

[12] Gregory Carey (2003),*Coding Categorical Variables*

[13] Joseph Lee Rodgers; W. Alan Nicewander (1988),*Thirteen Ways to Look at the Correlation Coefficient*,The American Statistician, Vol. 42, No. 1.

[14] Sunil Kumar and Ilyoung Chong (2018),*Correlation Analysis to Identify the Effective Data in Machine Learning: Prediction of Depressive Disorder and Emotion States*, International Journal of Environmental Research and Public Healt.

[15] Valeria Fonti (2017),*Feature Selection using LASSO*, VU Amsterdam

[16] S. B. Kotsiantis (2007),*Supervised Machine Learning: A Review of*

Classification Techniques, Department of Computer Science and Technology, University of Peloponnese, Greece

[17] Pedro Domingos,*A Uni
ed Bias-Variance Decomposition*, Department of Computer Science and Engineering, University of Washington

[18] Raul Rojas (2015),*The Bias-Variance Dilemma*

[19] Alaa Tharwat,Tarek Gaber, Abdelhameed Ibrahim and Aboul Ella Hassanien (2000),*Linear discriminant analysis: A detailed tutorial*,Department of Computer Science and Engineering, Frankfurt University of Applied Sciences, Frankfurt am Main, Germany

[20] S. B. Kotsiantis ,I. D. Zaharakis, P. E. Pintelas (2007),*Machine learning: a review of classification and combining techniques*

[21] Freek Stulp, Olivier Sigaud (2016),*Many regression algorithms, one unified model - A review*

[22] Matilde Ugolini (2015),*METODOLOGIE DI APPRENDIMENTO AUTOMATICO APPLICATE ALLA GENERAZIONE DI DATI 3D*

[23] Lars Buitinck & C. (2013),*API design for machine learning software:experiences from the scikit-learn project*

Sites

[24] Bottero,`https://www.bottero.com/it`

[25] *548 Lam*,`https://www.bottero.com/taglio/548%20lam/58450`

[26] Jupyter Notebook, `https://jupyter.org/`

[27] Pandas, `https://pandas.pydata.org/`

[28] SKLearn, `https://scikit-learn.org/stable/`

[29] Wikipedia, *Vetro Stratificato*, `https://it.wikipedia.org/wiki/Vetro_stratificato`

[30] SKLearn, *User Guide*, `https://scikit-learn.org/stable/user_guide.html`

[31] Pandas, *Dcumentation*, `https://pandas.pydata.org/pandas-docs/stable/`

[32] Coursera, *Machine Learning*, `https://www.coursera.org/learn/machine-learning`

[33] Wikipedia, *Supervised learning*, `https://en.wikipedia.org/wiki/Supervised_learning#Algorithm_choice`

[34] Gaurav Gahukar, *Classification Algorithms in Machine Learning...*, `https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4`

[35] Ritchie Ng , *Evaluate linear regression*, `https://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/`

[36] Scott Fortmann-Roe, *Bias Variance Tradeoff*, `http://scott.fortmann-roe.com/docs/BiasVariance.html`

[37] Sunil Ray, *7 Regression Techniques you should know!*, https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[38] Aditya Mishra, *Metrics to Evaluate your Machine Learning Algorithm*, https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

[39] Jason Brownlee, *How to use Learning Curves to Diagnose Machine Learning Model Performance*, https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[40] *Correlation In Python*, http://benalexkeen.com/correlation-in-python/

[41] Hugo Ferraira, *Dealing with categorical features in machine learning*, https://medium.com/hugo-ferreiras-blog/dealing-with-categorical-features-in-machine-learning-1bb70f07262d

[42] Jason Brownlee, *Feature selection for machine learning in Python*, https://machinelearningmastery.com/feature-selection-machine-learning-python/

[43] Jason Brownlee, *Statistics for Machine Learning (7-Day Mini-Course)*, https://machinelearningmastery.com/statistics-for-machine-learning-mini-course/