POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Quality Data Analysis - Project Work [FULL PROJECT]

Project Report of Quality Data Analysis - Mathematical Engineering

Team number: 1
Components: Giulia Mezzadri, Federico Angelo Mor, Abylaikhan Orynbassar, Federica Rena
Academic year: 2023-2024
Professor: Panagiotis Tsiamyrtzis
Project page: `https://github.com/abylai11/qda-project`

Contents

Phase 1

1.1. Introduction

Ensuring product quality and operational efficiency is critical in modern manufacturing processes as the ability to detect defects in real-time during production, known as in-line detection, is essential for maintaining high standards and minimizing waste and costs. In-line defect detection allows in fact for immediate identification and correction of production errors, preventing defective products from progressing through the manufacturing process. This real-time monitoring enhances operational efficiency by reducing idle time and the need for rework. Furthermore, it helps in maintaining a high level of product quality, which is essential for customer satisfaction and competitiveness in the market.

The purpose of the project was to create a robust statistical process monitoring (SPC) method, specifically for the in-line detection of defects on Voronoi filters printed with an HP Jet Fusion 580 Color 3D printer. As our samples, we had filters organized on ten trays, each containing four filters, and their top-view gray-scale images were captured at the MADE Competence Center.

Our project was structured into two phases. In the first one we were given objects without any defects to develop our statistical process control method, while the second phase involved testing our proposed method to identify objects with various defects. We used the Shewhart univariate control chart and multivariate control chart to design our SPC methods, as they provide a visual and statistical method for monitoring process stability and detecting anomalous variations, helping the organization to maintain consistent quality and identify opportunities for process improvement.

In the following sections, we will detail the project's methodology, results, and implications, offering insights into how our SPC method could be adopted to drive quality improvements and operational capabilities.

1.2. Assumptions and preliminary data analysis

Data manipulation

Once completed the image acquisition we ran the python script provided on webeep to generate the images' statistics, obtaining the first dataset, which we will refer to as `df_old`.

At that initial stage the unique "key" which identified each of the four objects inside each image was the combination of the `Image Name` and `Position` variables, so we opted to combine them into a single identifier, `Part_ID`, using a simple function. This way we could more easily refer to single objects in the subsequent analysis.

The python script also included a segmentation part, which automatically isolated the objects by cropping a region around them. However, the crop was quite large and the objects inside were tilted, so we decided to implement our own code to do the segmentation. We deemed this step necessary since we thought that to compare more fairly the statistics of the images we needed a "common ground" (that is, perfect framing) on which they should be computed.

To perform this correction we binarized the images using Otsu treshold, detected the borders using a Canny filter, selected the main outer edge, and from that we derived the parameters for the proper rotation matrix and crop region (see Figure 1). After this refinement we re-computed their statistics obtaining the dataset `df_new`.

Before continuing with the analysis, we also deemed appropriate to split both old and new

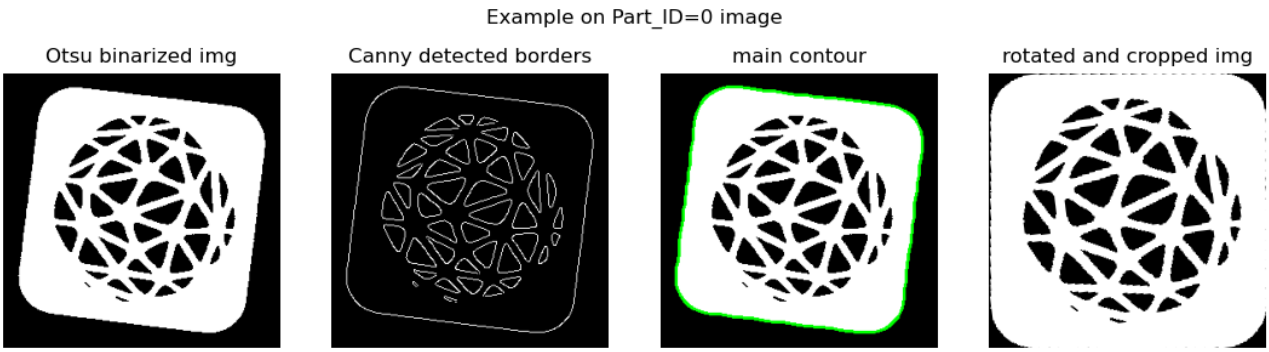


Figure 1: Visualization of the steps of our segmentation algorithm.

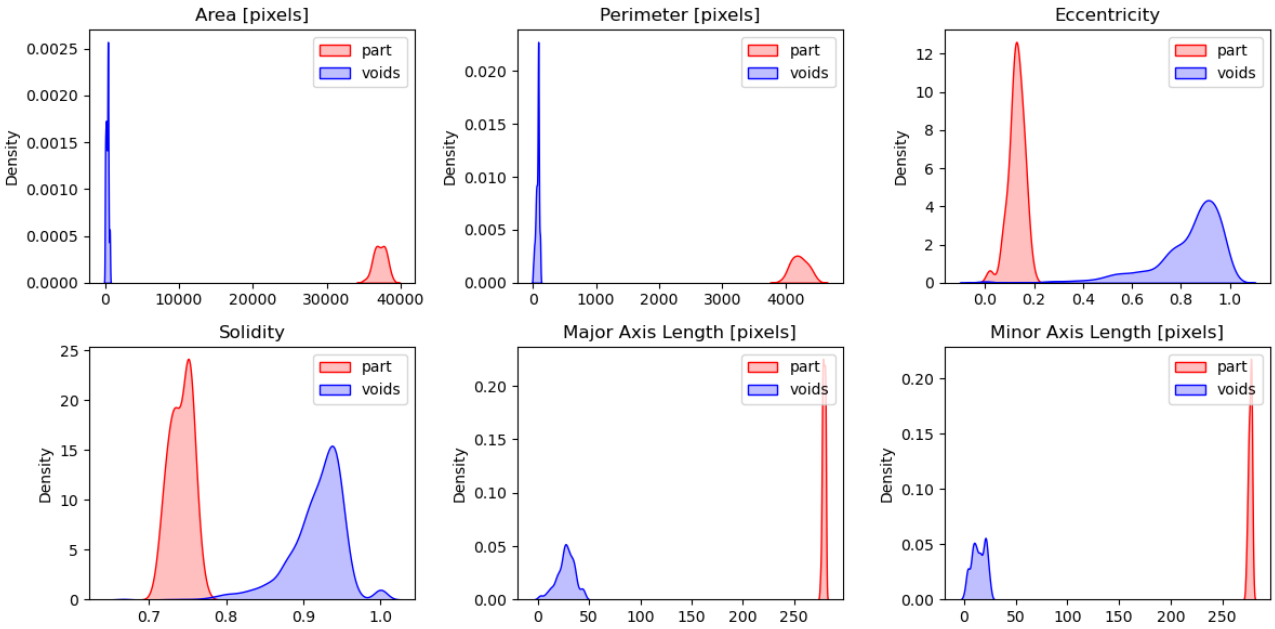


Figure 2: Visual comparison of the approximated distributions between parts and voids, for some variables of the `df_new` dataset. Similar results were also present for the `df_old` one.

datasets into a *part* subset and a *void* subset, since they are constitutionally different features (the full object versus the small holes inside it), and we also saw a clear separation in the distribution of the variables, as depicted in Figure 2 and confirmed by ANOVA tests about the difference of the variables in the void and part groups, of which we report the pvalues of the tests in Table 1.

Variable Name	Pvalue of ANOVA test
Area	0.0
Perimeter	0.0
Eccentricity	$4.3781482 \cdot 10^{-146}$
Orientation	$4.6979078 \cdot 10^{-8}$
Solidity	$2.8502201 \cdot 10^{-157}$
Extent	$2.4072606 \cdot 10^{-23}$
Major Axis Length	0.0
Minor Axis Length	0.0
Equivalent Diameter	0.0

Table 1: Pvalues of the ANOVA test, on each variable generated from the original python script, using the `df_new` dataset. The null hypothesis is that, for a certain variable i, the two groups have the same population mean.

Dataset comparison

The two datasets, the old and the new one, seemed to provide different values, so we investigated this difference through nonparametric intervals (Boostrap t-intervals) and the respective tests. In this preliminary phase of our analysis we did not assume normality: that's why we chose to use a nonparametric approach to perform the comparison. We studied the parts separately from the voids.

About the parts we tested, for each variable i,

$$H_0 : \mathbb{E}(\texttt{df\_old\_parts}[i]) = \mathbb{E}(\texttt{df\_new\_parts}[i]) \quad \text{vs} \quad H_1 : H_0^c$$

and we obtained the following confidence intervals (where `Voids number` is a new variable that we quickly created to store how many voids each object had):

Variable name	Lower CI	Point Estimate	Upper CI	$0 \in$ CI?
Area	-1539.53	-1452.3	-1362.32	no
Perimeter	87.112	96.9639	105.541	no
Eccentricity	-0.0022843	0.0025	0.00738961	yes
Orientation	-0.456515	-0.057225	0.0647833	yes
Solidity	-0.0227009	-0.02125	-0.0197547	no
Extent	0.00990142	0.017275	0.029109	no
Major Axis Length	1.02045	1.1686	1.3052	no
Minor Axis Length	0.914555	1.03865	1.15556	no
Equivalent Diameter	-4.4616	-4.20975	-3.94027	no
Voids number	-0.550939	-0.2	-0.0646082	no

Table 2: Confidence intervals for `df_old_parts` against `df_new_parts` variables.

As the majority of intervals do not include zero, we can confidently reject the null hypothesis in most cases with a confidence level of 95%, which means there is a substantial difference between `df_old_parts` and `df_new_parts`.

Regarding the voids the same test was conducted, albeit with a variation: in this case, we considered 40 distinct datasets, each corresponding to a different image (since originally we had ten images of trays, each of which with four objects) in order to compare the different variables related to the same image. So for each variable i referred to the image j we tested

$$H_0 : \mathbb{E}(\texttt{df\_old\_voids}^{(j)}[i]) = \mathbb{E}(\texttt{df\_new\_voids}^{(j)}[i]) \quad \text{vs} \quad H_1 : H_0^c$$

and we obtained the following intervals (for the sake of example we show only the ones related to the first image, i.e. `Part_ID` $= 0$):

Variable name	Lower CI	Point Estimate	Upper CI
Area	-27.3227	18.8049	63.3162
Perimeter	-3.93578	3.15383	9.72439
Eccentricity	-0.0363188	-0.00497561	0.0276549
Orientation	-0.462142	-0.0247073	0.382791
Solidity	-0.0273247	-0.0168537	-0.00623376
Extent	-0.0241511	0.00826829	0.0376097
Major Axis Length	-1.79098	0.688024	2.91621
Minor Axis Length	-1.0699	0.578268	2.20956
Equivalent Diameter	-1.0602	0.667683	2.34608

Table 3: Confidence intervals for the difference of variables between `df_old_voids` and `df_new_voids`; reported for `Part_ID` $= 0$.

Almost all intervals seem to include zero and so we cannot reject the null hypothesis in the majority of the cases.

These tests show how there is a substantial difference between `df_old` and `df_new` (or regarding what we will focus on hereafter, `df_old_parts` and `df_new_parts`). However, this difference should be read as an improvement step, since the new dataset proved to have better regularity and properties with respect to the old one, for example granting gaussianity to some variables that didn't have it in the old dataset, not even after a box-cox transformation.

Variables Analysis

In order to reduce the dimensionality and select only the necessary variables, we started from the ones respecting the assumptions of gaussianity (through a Shapiro-Wilk test) and excluding `Voids number` which being discrete couldn't be easily implemented in control charts.

As we will better describe in Section 1.3, we studied correlation both graphically, through a scatter plot, and quantitatively, through the correlation matrix (of which Figure 3 gives a visualization). From that we noticed how we could exclude some highly correlated variables and filter just a few "representatives" among them, in order to have a single variable carrying the information shared by multiple ones (e.g. `Area`, `Extent` and `Solidity` all relate to the amount of white pixels in the image). Indeed some variables conveyed the same information as others, a perfectly explainable result since the rotation and cropping had "standardized" the images, thus making some measurements redundant.

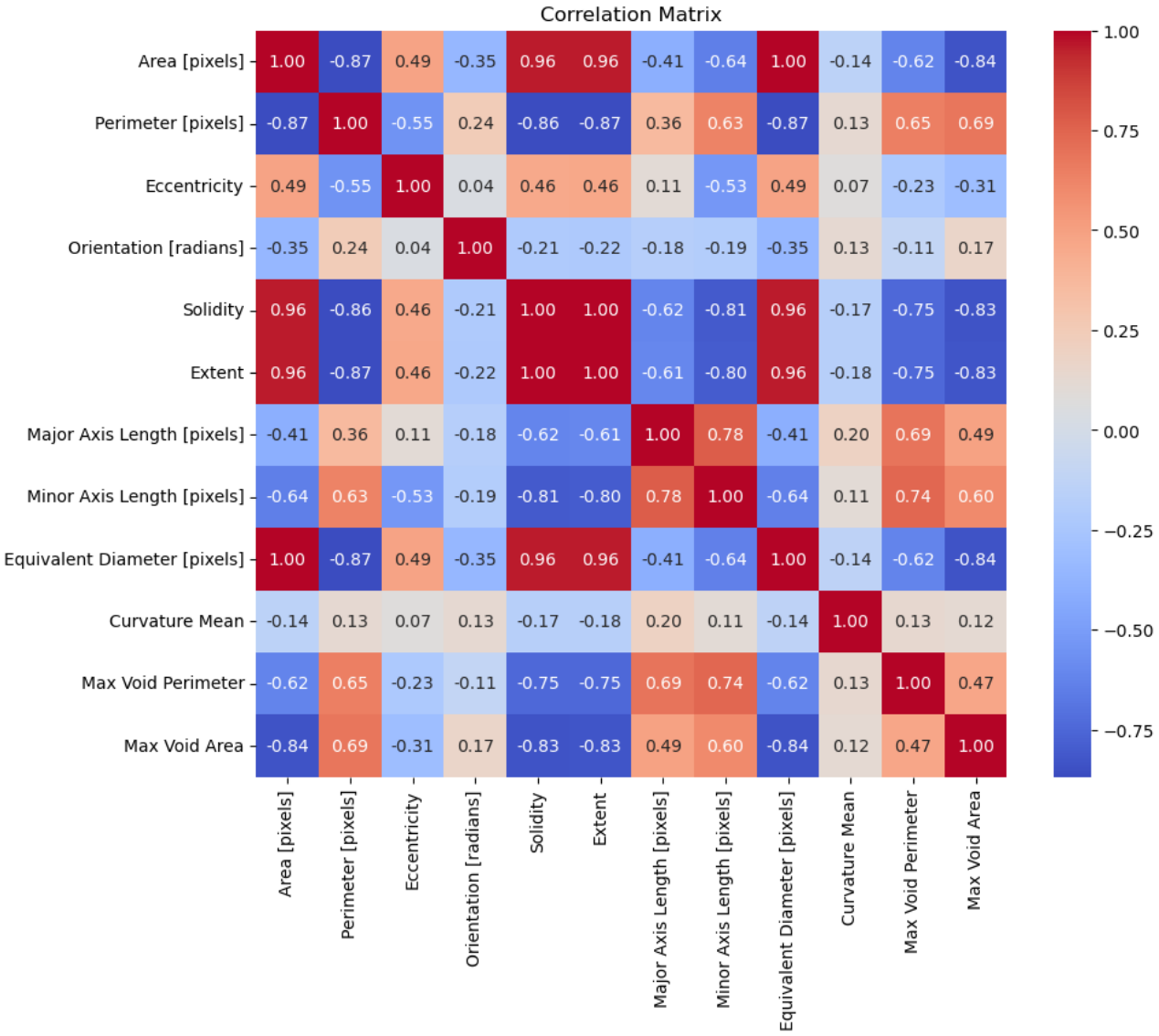


Figure 3: Correlation matrix of the variables of the dataset `df_new_parts`. The newly introduced variables `Curvature Mean`, `Max Void Perimeter` and `Max Void Area` will be defined in Section 1.3.

Another attempt at dimensionality reduction was brought through Principal Component Analysis. We applied PCA on the standardization of the dataset consisting of the previously selected uncorrelated columns, but the remaining dimensionality was still pretty high, and the loadings didn't allow a simple interpretation (and were not gaussian).

A PCA had also been made on the complete dataset, but even if it highly reduced the number of dimensions needed, the problems with normality of the selected PCs and interpretability made the attempt unsuccessful.

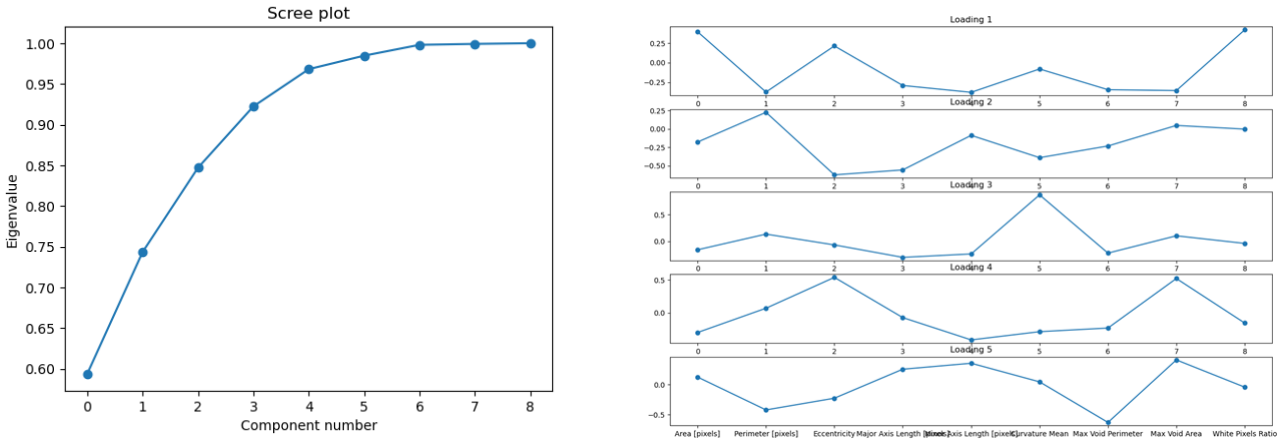


Figure 4: PCA results on the complete dataset.

1.3. Proposed metodology

Before describing in details the steps we performed, we would like to highlight a first important modeling choice. The structure of the collected data led us thinking about two possible approaches:

1. using the original images, in which there were four objects in each tray;
2. using the objects-isolating images, obtained trough the segmentation code.

Having in total 40 objects, case 1 would have implied to use $m = 10$ samples with $n = 4$ as sample size; while in case 2 we would have set $m = 40$ and $n = 1$, i.e. considering individual observations.

Since the printer prints objects individually, we though the second approach to be more useful in a real-life scenario, since it would imply a more accurate monitoring on each object, rather than waiting for a set of them to be printed and then control that set.

This clarification will be important only in Idea 1 and 2 sections; while now we illustrate the work we performed on the dataset variables.
Note that the second approach will be discussed in Section 1.4 as it will be one of the methods we will use in the second phase.

Variable selection and definition

As mentioned before, towards the building of control charts we decided to consider only the necessary variables, looking for the ones respecting the assumptions of gaussianity (through a Shapiro-Wilk test) and randomness (through a Runs test), and that seemed worthy and meaningful to be included.

Therefore, our main idea was to select a minimal set of variables (i.e., avoiding redundancies), which could be useful to detect objects' defects, taking into account both mathematical assumptions and qualitative reasoning.

In this direction, we chose to remove the following variables:

- `Orientation`, since the new dataset cropping removed any difference in orientation between images.
- `Solidity` and `Extent`, as the information was already captured by `Area` after the re-centering correction.
- `Eccentricity`, as we were not sure about how the computation behind it worked, nor if

it could be useful.

- `Minor Axis Length`, since we decided to keep `Major Axis Length` instead, which together with the minor (through a ratio) were highly correlated to `Eccentricity`; therefore including also the minor axis would have morally meant to include the eccentricity information, which instead we had discarded.

So we were left having saved `Area`, `Perimeter` and `Major Axis Length`.

To support our argument about `Eccentricity` ineffectiveness, we created an image with a visible deformation, but the obtained value seemed to be (wrongly) close to the correct ones. So in the end we did not valued this variable as interesting.

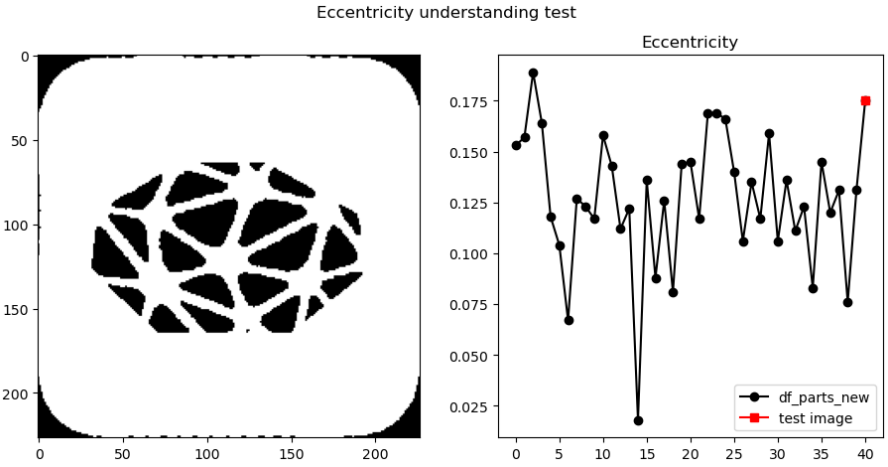


Figure 5: Plot of the test that we performed to understand the eccentricity variable.

Related to the `df_new_voids`, we decided not to keep any of the proposed variables because it made sense for us to include a variable if and only if it had a normal distribution in all images. However, we did not find any variable that satisfied this constraint. Moreover, we couldn't think the variables in that dataset as i.i.d. since the voids were different in terms of location and dimension, and problematic to group and compare. So we did not use this dataset as it was, but processed it to generate two new variables illustrated below.

After experimenting a bit trying to extract some more information from the images, we generated some new variables, to improve the descriptive capabilities of our model, and here we report the ones that made it to the control charts:

1. `Curvature Mean`. From our own segmentation code, we took the section about the main contour extraction and from there we computed the curvature for each set of three consecutive points. This new variable stores the mean of all those values.
2. `Max Void Area`. The voids of the objects were not i.i.d., due to different sizes and positions, so we could not use them altogether. Instead, we decided to consider, for each object, only the void with the biggest area (and not the one of smallest area because it wouldn't have been much informative as the segmentation of smaller voids was heavily affected by pixels noise).
3. `Max Void Perimeter`. With a reasoning similar to the previous variable, we decided to include the information about the perimeter of the biggest void inside each object. In fact, the coefficient of correlation with `Max Void Area` was just 0.47, therefore we thought this variable as also worthy to be included.

Regarding voids but being related to each object individually, the last two variables were computed from `df_new_voids` but stored as new variable in `df_new_parts`, to have a single dataset to work on. Another new variable not included but still worth mentioning is `White`

`pixels ratio`, created as an attempt to obtain a more robust equivalent of `Area`, being the new one a percentage of white pixels rather than a simple count.

The following table summarizes how these new variables, together with those already present, could detect defected pieces:

Variable	Kind of defect it could help to detect
Area	too many holes; broken pieces
Perimeter	wrong dimensions; presence of bulges
Major Axis Length	flattening deformations
Curvature Mean	cracks along perimeter
Max Void Area	too big void in the inside; wrongly shaped voids
Max Void Perimeter	voids collapsed together; weirdly shaped voids

Table 4: Interpretation about which kind of defects these selected variables could help to detect, through their control charts.

All the variables satisfied the gaussianity assumption but we had some problems regarding the independence assumption. We proposed two approaches to deal with it, based on the different weight and role given to time.

Variable	Shapiro Test pvalues	Dasaset
Area	0.7310	`df_new_parts`
Perimeter	0.8690	`df_new_parts`
Major Axis Length	0.0967	`df_new_parts`
Curvature Mean	0.5501	`df_new_parts`
Max Void Area	0.5666	`df_new_parts`
Max Void Perimeter	0.9064	`df_new_parts`

Table 5: Pvalues of the Shapiro test to check normality on the variables we selected for the control charts.

Idea 1: FVC and SCC

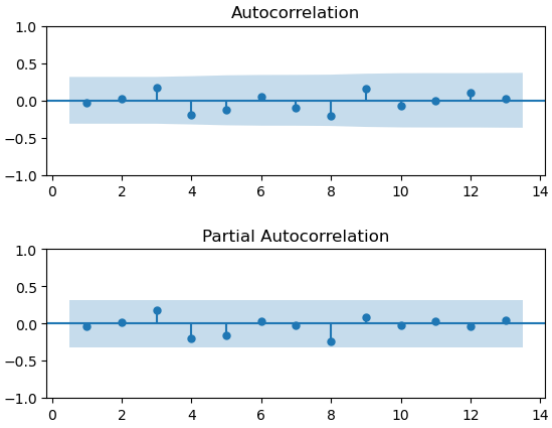
The first approach we tried out consisted in treating the data as time series, assuming that issues with independence were caused by underlying models and patterns.

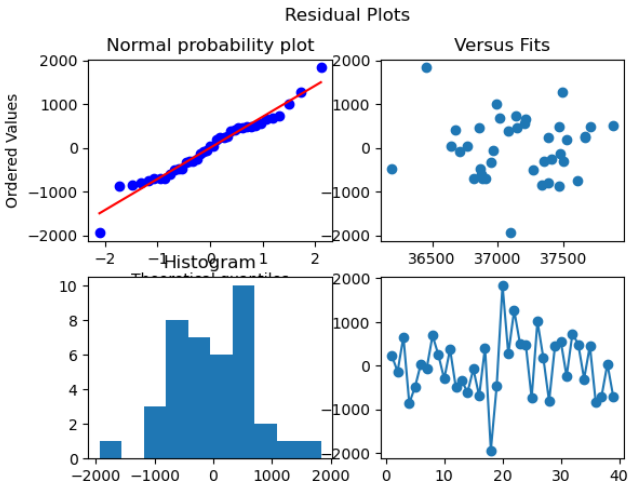
We started with I-MR-R charts, noticing an improvement already, but dealing whit the independence problems and building Fitted-Values chart (FVC) and Special-Cause chart (SCC), where needed, was a better idea.
We present the results a variable at a time as ordered in Table 5.

Area

Looking at the pvalue of the runs test (0.025) and at the auto-correlation plots, we noticed that the independence assumption was not reached.

We decided to build an autoregressive model of order one (AR(1)) to model the data and obtain SCC and FVC control charts. The residuals were indipendent, with a pvalue for the runs test of 0.254, gaussian with pvalue of shapiro-wilk of 0.496, and homoschedastic as shown in the plots below.





The obtained control charts were:

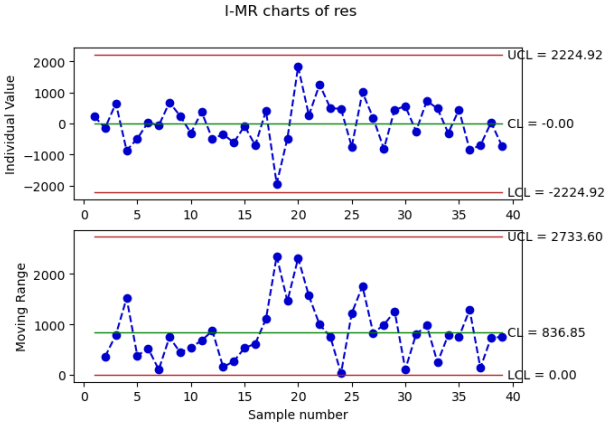


Figure 6: SCC for Area

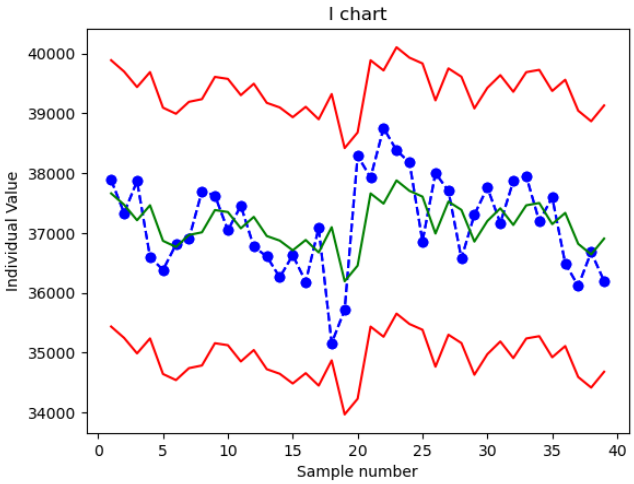


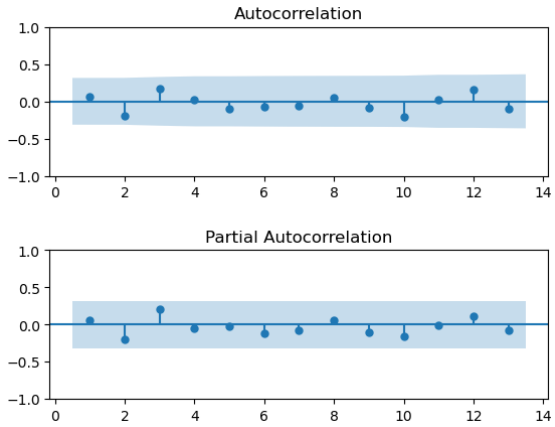
Figure 7: FVC for Area

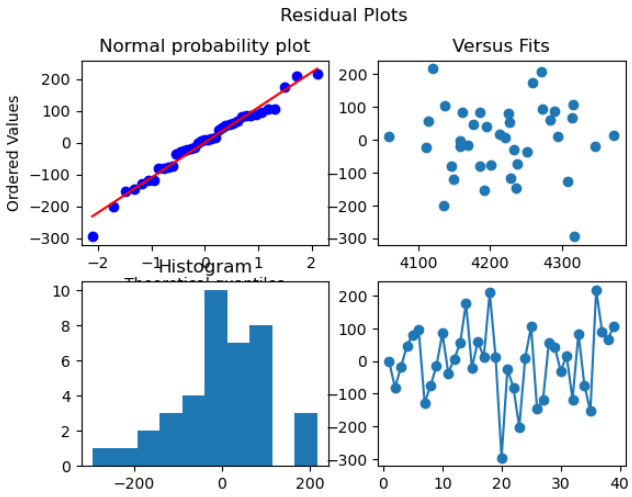
No strange pattern, outliers or out of control observations appear.

Perimeter

In the following variable we observed the same situation we experienced with the `Area` variable: we had Gaussian distribution but we missed independence (the pvalue of the run test was equal to 0.026).

As done before, we modeled the data with an AR(1) model obtaining independent residuals (the pvalue of the run test was equal to 0.436) with gaussian distribution (Shapiro-Wilk test p-value equal to 0.685) and same variance.





The related control charts were:

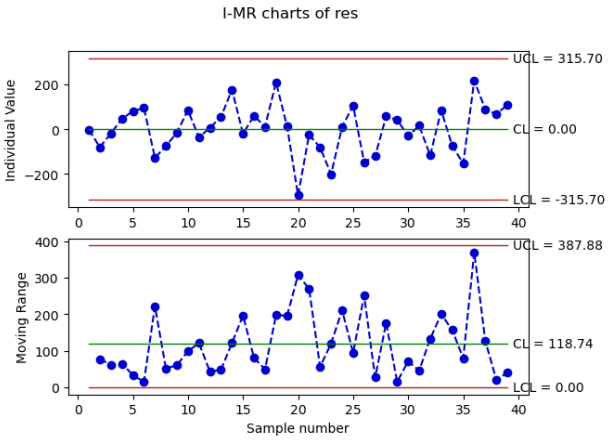


Figure 8: SCC for Area

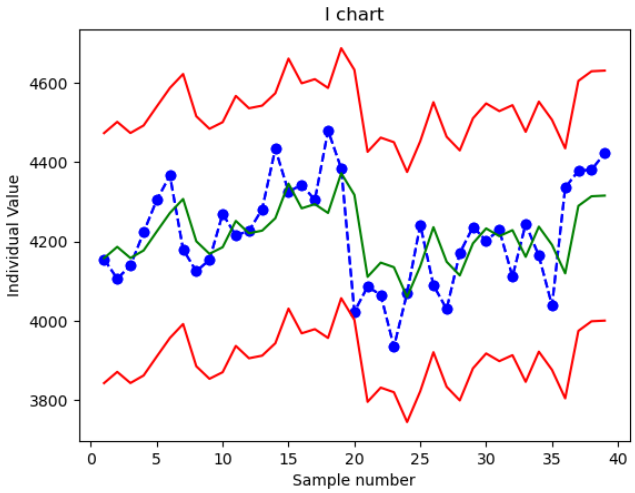


Figure 9: FVC for Area

As before, no strange pattern, outliers or out of control observations appear.

Major Axis Length, Curvature Mean, Max Void Area and Max Void Perimeter

For all the remaining variables considered, both the assumptions, gaussianity and independence were satisfied. We measured, in fact, the following pvalues of the runs test:

Variable	Runs Test pvalues
Major Axis Length	0.749
Curvature Mean	0.522
Max Void Area	0.749
Max Void Perimeter	0.161

Table 6: Pvalues of the runs test to check independence.

So, we were able to build the "classical" SPC control charts on the original variables and not on the residuals.

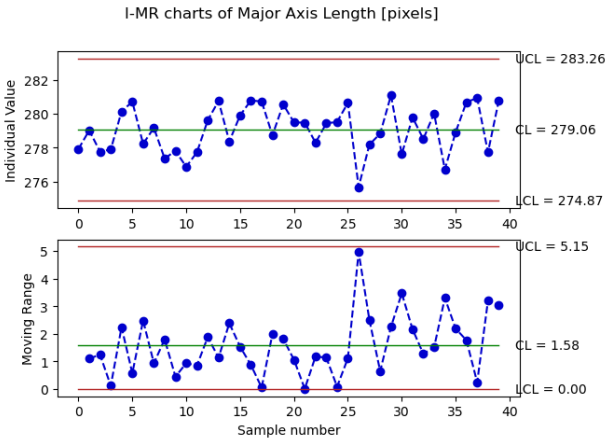


Figure 10: SPC for Major Axis Length

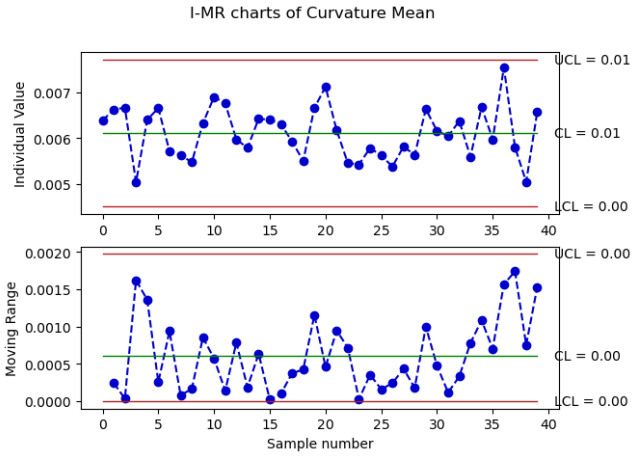


Figure 11: SPC for Curvature Mean

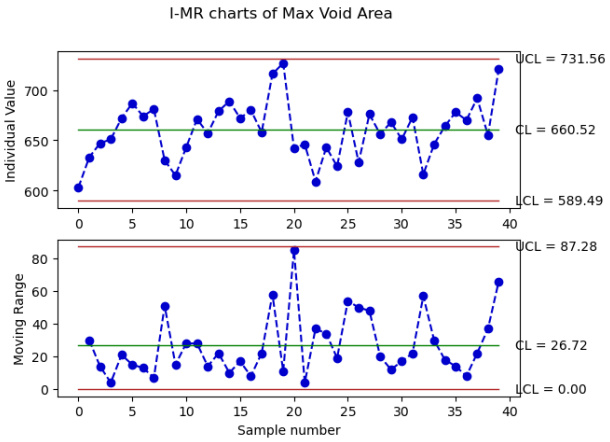


Figure 12: SPC for Max Void Area

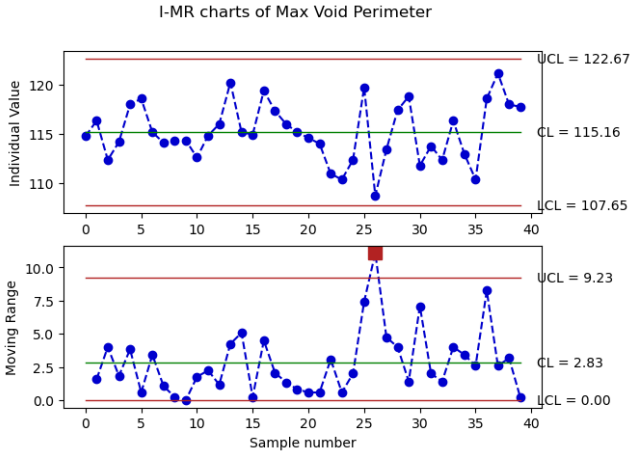


Figure 13: SPC for Max Void Perimeter

As we can see, only an out of control observation seems to be present in our control charts but no assignable cause were found. In addition to this, it is related to the Moving Range control chart that, as explained below, it will not useful for our scope. So, all the previous control charts are acceptable.

1.4. Results

As final models for Phase 1 and in anticipation of Phase 2, we present the following approaches:

1. Individual control charts on shuffled data, i.e Idea 2
2. Multivariate control charts on not shuffled data
3. Multivariate control charts on shuffled data

More details will be given in the sections below.

Idea 2: Charts on shuffled data

Due to the mechanism of the image acquisition, we could assume that the obtained data should not be considered as a time series. In fact, the objects to be placed in the trays were selected in a random order (not "carefully" as if they had a precise time indication). In other words, even if originally existed a particular time ordering, the one in which the objects came out from

the printer, the acquisition procedure made it to be lost; and we would have considered time only if their placement in the trays followed the precise order in which they came out from the printer. Indeed all the objects were in control, by design of the project, and they were disposed in the trays randomly, not according to a particular order.

Therefore, consistently with this idea, we could assert that there should be no problems with independence in the data, considering any independence problem as being caused by chance.

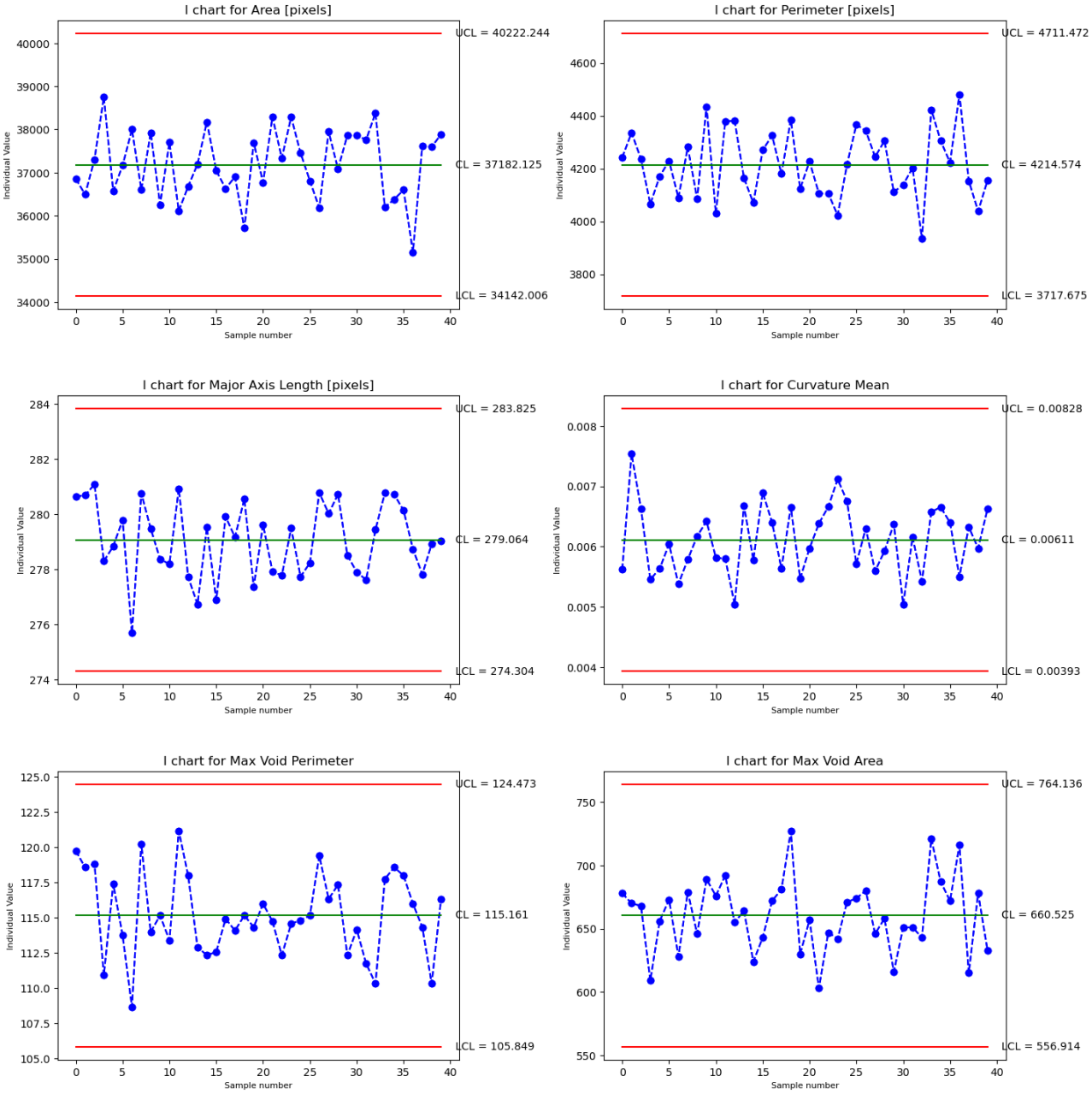


Figure 14: I charts, for the six variables we selected, generated according to the Idea 2.

For this reason, since the order in which objects were originally stored in the dataset caused problems with independence through the Runs test, as mentioned in Idea 1 section, we decided to shuffle the data.

This suited the new context since this procedure had no problems with respect to time issues, and having the objects randomly placed at the beginning meant that we could possibly re-order them in any way, without altering the design and construction of the experiment. This shuffling made us obtain independence (as the Runs tests showed in Table 7) which, as said, according

to the experiment design should have been there "by default".

Variable	Runs Test pvalues
Area	0.521
Perimeter	0.193
Major Axis Length	0.200
Curvature Mean	0.749
Max Void Perimeter	0.789
Max Void Area	0.109

Table 7: Pvalues of the Runs test to check randomness assumption in the shuffled dataset.

As we saw in the previous sections we had no problems with the gaussianity assumption, and therefore we built the control chart using this shuffled `df_new_parts` dataset. The resulting charts are depicted in Figure 14.

The random permutation introduced slight changes in the MR computation, as the random re-ordering made that the moving range values changed from shuffle to shuffle. But on average they were very close so this was not considered as a problem. Due to this slight variability, however, we just focused on the I charts and ignored the MR ones.

We decided to apply a Bonferroni correction in the end, since we made multiple (six) univariate charts on the same data so that each chart was individually built with a $\alpha = \frac{\alpha_{\text{tot}}}{6} = \frac{0.0027}{6}$, which gave a $K = 3.5$. The correction leads to larger value of K, so a more precise choice could be to try a single multivariate chart; but nonetheless the univariate charts would allow for better interpretability in identifying the variable causing the defect.

Even if we were not sure of this Idea 2, lacking some theoretical results to support it, we considered it convincing; a sensation proved by the comparison with the "classical" ones found in Idea 1. Indeed, for the variables which satisfied both the assumptions of normality and independence (`Major Axis Length`, `Curvature Mean`, `Max Void Perimeter`, `Max Void Area`) the limits obtained are very similar. This proves somehow the robustness of this approach. For the remaining variables `Area` and `Perimeter` we cannot compare them since in Idea 1 we built the limits on the residuals of their models. In addition to this, they are the ones with the most plausible and consistent hypothesis, as well as lacking issues of hugging or false alarms.

However we would compare it with the following multivariate charts, to see which method will perform better on Phase 2 data and investigate the strength and weaknesses of the two approaches.

Multivariate Control Charts

After the variable selection we performed, the multivariate charts seemed a natural and valid alternative to the individual charts. As we have seen, we saved six variables, and on them we conducted the multivariate analysis.

Normality assumption (marginally) was satisfied, as we checked before. Regarding the independence assumption not all the variables passed the Runs test, as we saw when describing Idea 1. Nonetheless we proceeded building the charts, considering that time, as described in Idea 2, was not really to be considered for this Phase 1 data.

Regarding the estimate of the variance matrix S, we explored both short and long period approximations, using the upper control limit given by the Beta distribution, as we are in Phase 1. However, the short one caused a false alarm in an observation. This was surely a false alarm, since all the images in this phase were of good objects, and was probably caused by

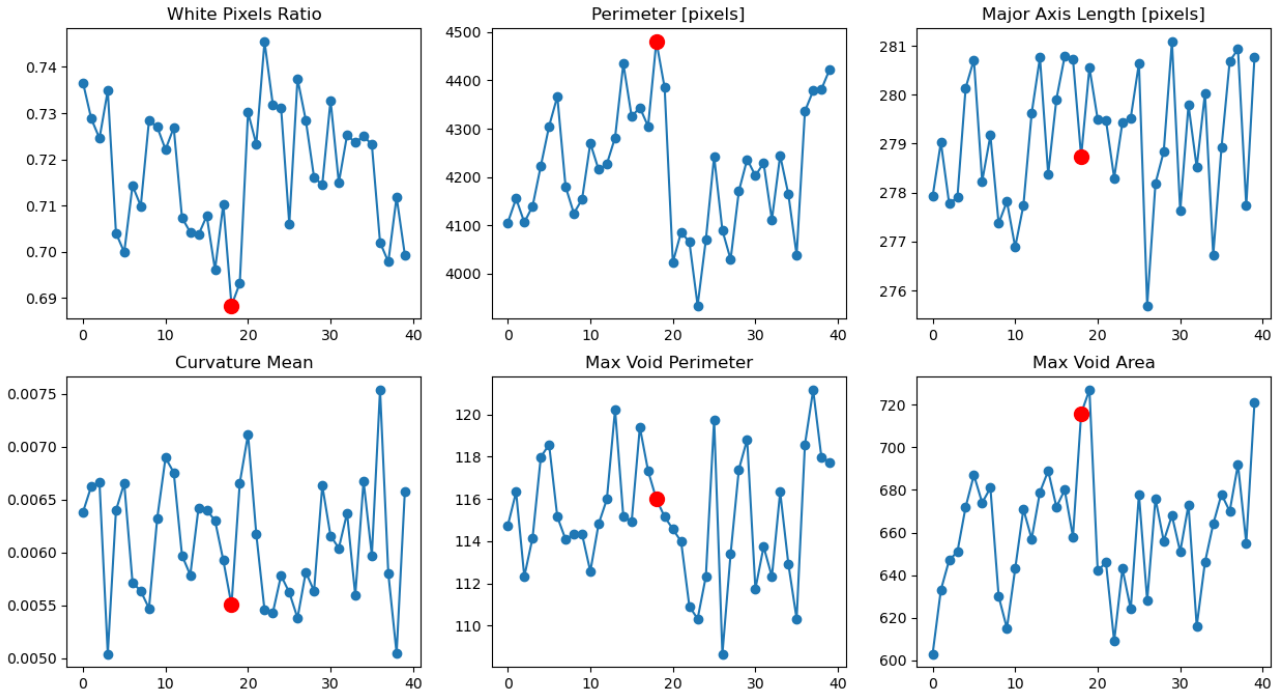


Figure 15: Highlight on the statistics of the object which produced the false alarm.

an object having two quite extreme values on `Area` and `Perimeter` variables, as we saw when inspecting the data in Figure 15 highlighting the values of the signaled object.

We therefore decided to keep the long period one, which did not raise any alarms and should in any case be accurate since we don't have outliers nor out of control data in this design phase. We employed an $\alpha = 0.0027$ and the final chart is reported in Figure 16.

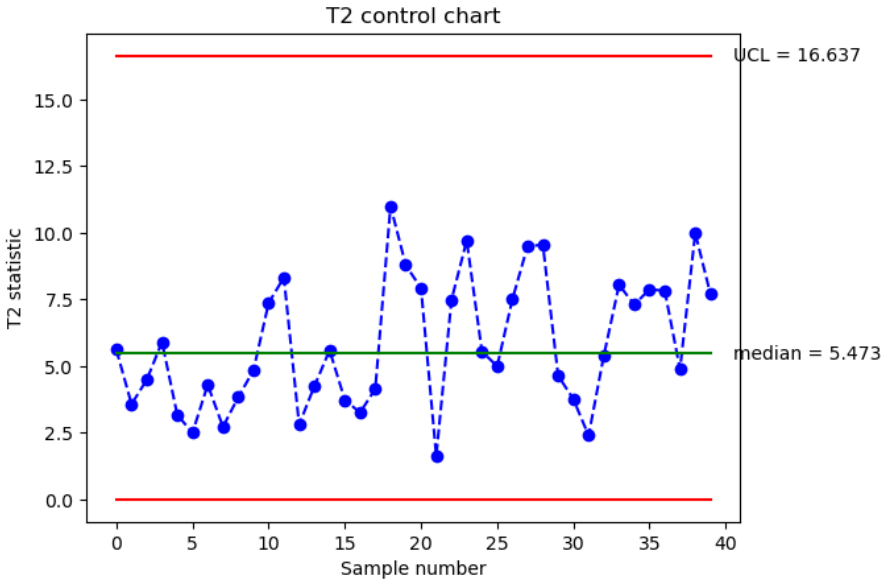


Figure 16: Multivariate chart for Phase 1, combining the six variables we selected, using *long period* estimation for S and the *original version* of `df_new_parts`.

Actually, we saw that the problem of the alarm in the short period estimation case was solved by using the shuffled version of the dataset, in a similar way in which it solved the MR issue. So as a final chart we propose the one of Figure 17, derived from the union of the multivariate and shuffling ideas.

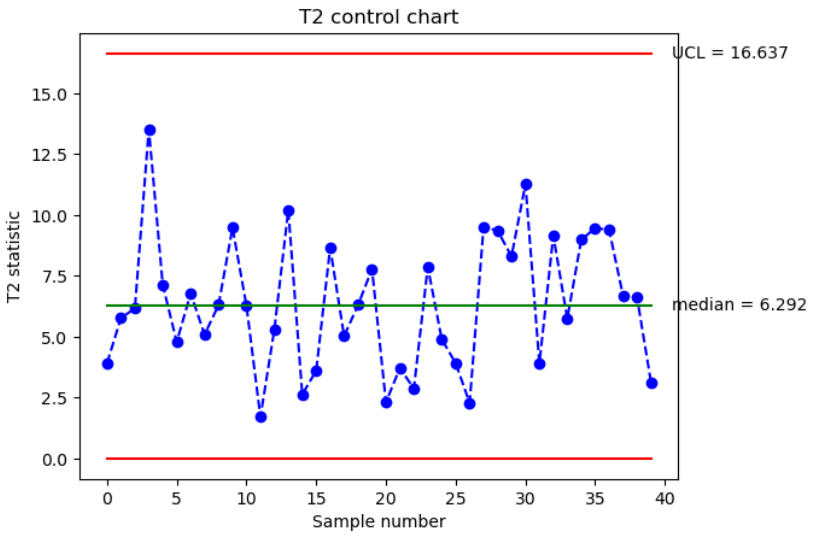


Figure 17: Multivariate chart for Phase 1, combining the six variables we selected, using *short period* estimation for S and the *shuffled version* of `df_new_parts`.

Phase 2

2.1. Preliminary data analysis

The dataset received for the second phase was of the same type as the dataset received in the first phase, where each image showed four objects placed on a tray. The only difference from phase 1 is that in this case, some objects are defective.

To apply the control charts identified in the first phase on the new observations, we proceeded in the same way described in Chapter 1.2.1: we segmented all the new images, generating `df_new`; subsequently, we divided the dataset into a *part* subset and a *void* subset and computed the new variables that we generated to improve the descriptive capabilities of our model and used them in the construction of the control charts.

Once the dataset was constructed according to our requirements, we examined by eye the images to identify the possible defects that could affect each object. We found three different type of defects and, for each, we considered which specific variable (and the related control chart) could identify it. Our ideas were as follows:

- **Edge errors** By edge errors, we mean all objects characterized by an uneven edge as shown in the following image. We thought that this error could be detected using the variables `Perimeter`, `Curvature`, and `Major Axis Length` depending on where the deformation occurred.

Figure 18: Defect on the lower edge.

- **Wider internal segments** Some objects showed wider and thicker internal segments. For this type of defect, the `Area` variable might be useful.

Figure 19: Evidence of a thicker segment.

- **Missing and/or unconnected internal segments** All objects characterized by an incomplete "internal grid" display this type of defect. We were strongly convinced that the variables we created, `Max Void Area` and `Max Void Perimeter`, would be useful in this specific case.

Figure 20: Defect related to missing segments.

The first small problem that we encountered concerned the `Curvature` variable, as after performing the tests on our control charts we saw that it seemed ineffective to signal the edge errors. For that reason we tried to correct the computation to make it more sensitive to defects. In particular, we changed the mean with the max, considered less points on the border, and verified that all assumption were still verified.

In the first implementation, in fact, the curvature was computed considering all the points detected with the contours detection, which however resulted in having too many points. In this way, the curvature of the regions with a anomalous behavior would have been smoothed out by the quantity of points around that region (the large quantity meant that the points "hugged" the region rather than highlighting discontinuities), and using the mean meant that the anomalies detected would have in any case a lower impact, since they would be leveled by of all the other "normal" points. So we decided to filter just 100 equispaced points on which compute the curvature, and to select the maximum curvature value. The result is depicted in Figure 21.

The assumptions were in any case still satisfied: on the Phase 1 data we get 0.5104 as pvalue of Gaussianity, while 0.531 and 0.760 for the pvalues of the Runs test, applied on the original and on the shuffled data respectively. After all this, the `Curvature Mean` (which despite the update kept the same name for the sake of code simplicity) charts started detecting defected objects as expected.

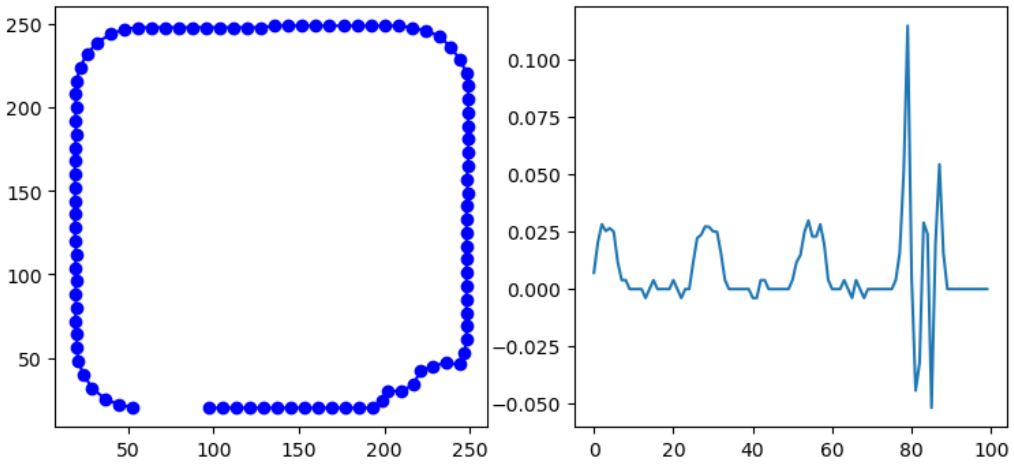


Figure 21: New curvature computation.

2.2. Test of your proposed approach on new data

We will describe the results for the three methods we decided to bring in Phase 2:

- univariate charts on the shuffled data
- multivariate charts using long period estimation for S and original data
- multivariate charts using short period estimation for S and shuffled data

We focused on both the univariate and multivariate approach as to compare them and investigate whether the methods are equivalent or discordant.

Univariate charts

As already stated in Phase 1, we decided to use the shuffled dataset to build the control charts that we will use to monitor the new data, as we deemed this approach more robust and consistent.

In the univariate case with Bonferroni correction, whose charts are represented in Figure 22, 6 unique objects were identified as defects out of the 9 that we noticed by eye. These detections were mainly thanks to `Major Axis Length`, `Curvature Mean`, `Max Void Area` and `Max Void Perimeter`. No false alarms were observed.
The method's interpretability helped in confirming our idea that the newly created variables would be determinant in the building of the charts.

Multivariate charts

For what concerns the multivariate case, we tried both the long period on original data and short period on shuffled data approaches, finding out that they behaved very similarly and detected the same 7 out of 9 errors, again without raising false alarms.
In Figure 23, we report the logarithm of the charts, since the scale of some out of control didn't allow a clear view of the points.

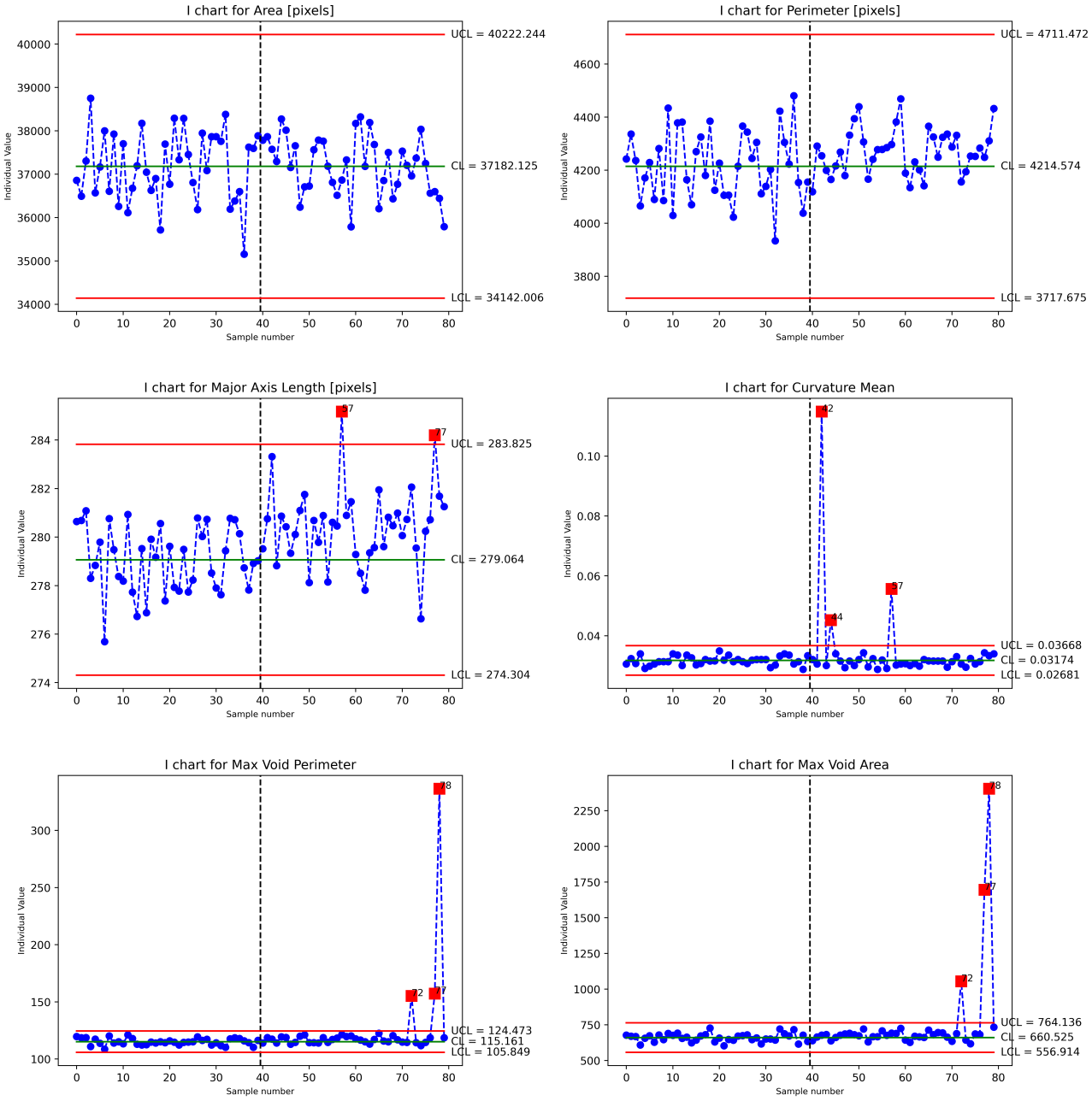


Figure 22: Bonferroni corrected univariate control charts.

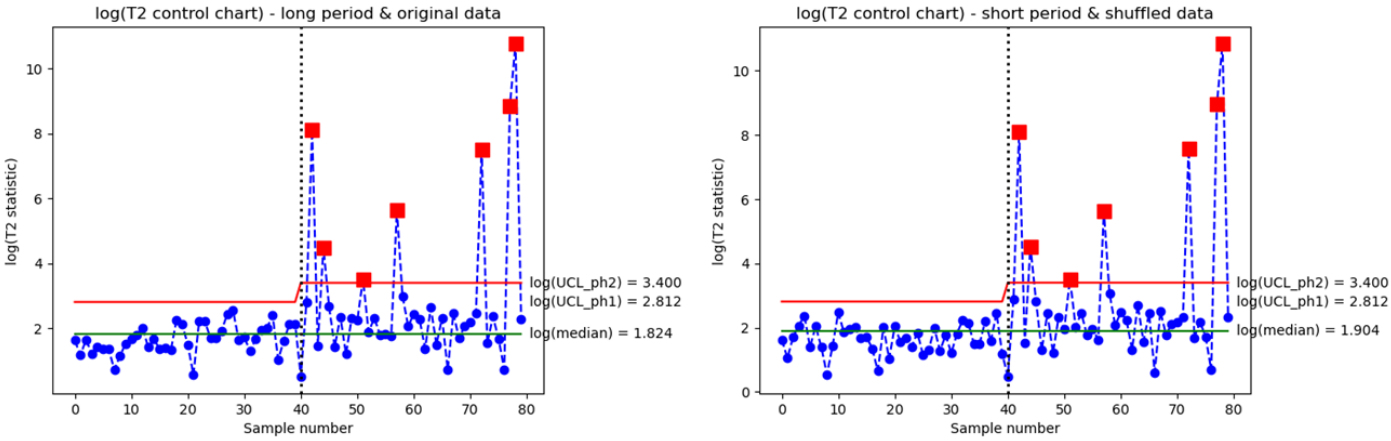


Figure 23: Multivariate log control charts.

A last attempt was made considering different trays as samples and building a chart able to detect defective samples. It proved unsuccessful and superficial since it only signaled the last tray as defective, also a chart only considering trays is probably useless compared to individual chart in this specific context, where we are interested in single defected objects. A last critical point about this method is the fact the chart shows signs of hugging.
This led us to choose not to consider this approach.

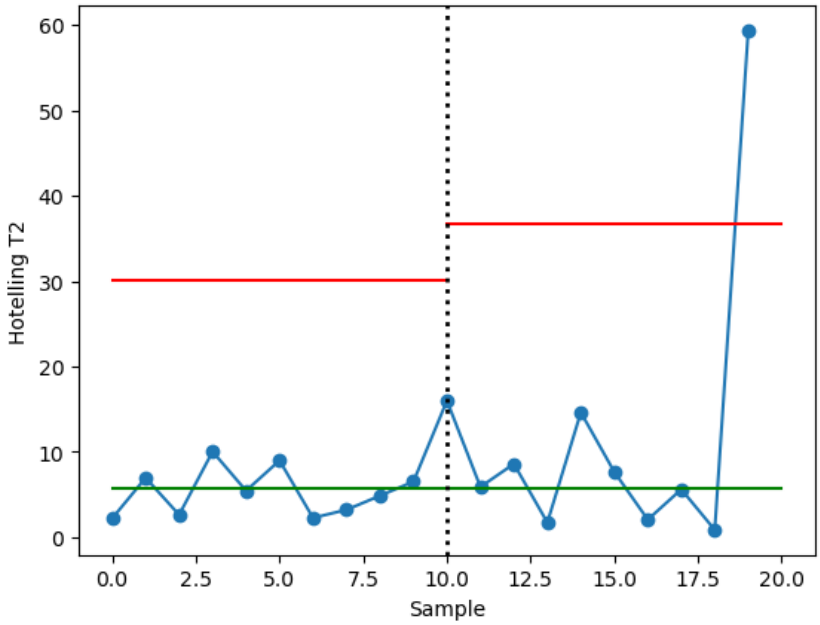


Figure 24: Control chart using trays as samples.

The absence of false alarms gave us the idea to try different and higher values of alpha to see whether we would become able to detect more borderline cases of defected pieces. Unfortunately it didn't work, as the univariate chart just got a false alarm more, while in the multivariate case nothing changed.
This might depend on the fact we need more specific variables to capture the different kinds of defects in the images.

2.3. Discussion

In the end, we believed that the performances of our charts were quite good. From the overall 9 defected objects, we were able to detect 6 and 7 of them using univariate and multivariate charts respectively.

In any case, our approach strongly highlighted the need of an accurate preliminary step in data analysis and variable definition. Indeed, without `Max Void Area`, `Max Void Perimeter` and `Curvature` the performances of our model would have been drastically reduced, since especially in the univariate charts we can see the ineffectiveness of the original `Area` and `Perimeter` variables. But also the other original variables seem to be not strong enough to detect the anomalies in the objects, as we can see in Figure 25, where most of the defected objects do not show anomalous values in original variables such as `Orientation`, `Solidity` or `Equivalent Diameter`.

Our idea of considering individual objects, rather than four objects at once inside each tray, proved to be also effective. Indeed only two couples in the out of control signaled indexes came out from the same tray, while each of the other five remaining detections emerged from different trays.

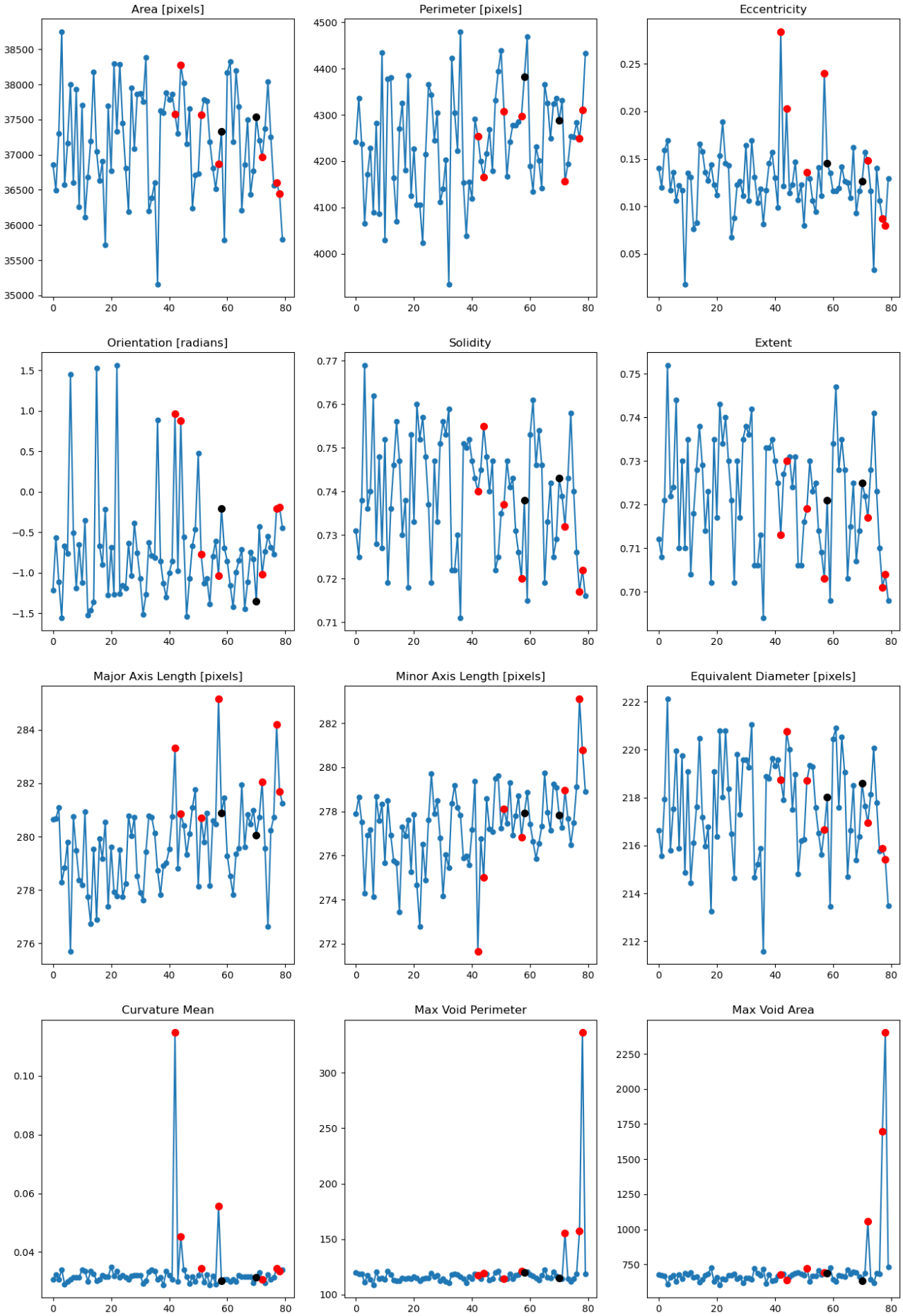


Figure 25: Plot of the defected objects vs all dataset variables. In red the defected objects correctly detected, in black the defected objects that our model missed.

We also saw that no wrong alarms were signaled (in the sense of out of control declared objects that instead were good) in both approaches. Therefore also the univariate charts remain robust and efficient, despite the Bonferroni correction which lowered the power of the model.

Anyway, our analysis in general points out how the task at hand should be properly inspected to generate the most informative and characteristic variables as possible.

We believe in fact that by developing another *ad hoc* variable, say `Max Segments Width`, our models would have detected all 9 faulted objects, since the ones that we missed were characterized by having an anomalous edge in the inside structure, as we saw in Figure 18. To develop this variable there could be different approaches available:

1. Take the image of the inside structure from a perfect object, overlay and fit it to the image at test, so to "center" it with respect to the one below, and compute the pixel-wise difference among the images. This could highlight larger white spots (the wider edges) that in a correct piece there should not have been present.
2. Try to remove (by masking) the junction points of the Voronoi structure by looping on the image and fitting black circles to cover up the vertexes. In this way only the segments would remain and we can proceed doing computations on them.

Or also other ideas that we can hint as a future development track.

References

[1] J. Hao. Segmentation implementation: Given an image and a 4 points in the image (no 3 points are co-linear). find the rotated rectangle enclosing the polygon formed by the 4 points and crop the rotated rectangle from the image. all done by opencv. `https://gist.github.com/jdhao/1cb4c8f6561fbdb87859ac28a84b0201`, 2018.

[2] D. C. Montgomery. *Introduction to Statistical Quality Control*. Wiley, 8th edition, August 2019.

[3] S. User. Segmentation implementation: How to straighten a rotated rectangle area of an image using opencv in python? `https://lc.cx/v8Pl4c`, 2012. The reported link has been reduced.

[4] Wikipedia contributors. Curvature — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Curvature`, 2024.