

SOFTWARE QUALITY EVALUATION PROJECT

GIANLUCA MOLTENI

MAT. 730113

Sommario

INTRODUCTION	3
DATASET DESCRIPTION.....	4
FEATURE SELECTION.....	7
GRAPHS.....	9
DESCRIPTIVE ANALYSIS.....	17
DATA ANALYSIS TECHNIQUES.....	19
LINEAR REGRESSION.....	19
K-NEAREST NEIGHBOR (KNN)	24
VARIATION OF K	25
ANALYSIS RESULTS.....	26
OUTLIERS MANAGEMENT	28
PEARSON, SPEARMAN, KENDALL STATISTIC.....	30
CONCLUSION	32

INTRODUCTION

The project aims to perform a study of a dataset, using two data analysis techniques. The result will serve to understand if there are limitations on use.

The chosen language is Python, excellent for data analysis tasks and implemented through a Notebook file. A support was given by the following libraries: Pandas, numpy, pylab, seaborn, plotly, sklearn and scipy.

The project is divided into 7 sub-objectives:

- Selection of the dataset
- Selection of a dependent variable. In our case is: software faultiness
- Selection of three independent variables
- Diagram visualization
- Computation of descriptive statistics
- Application of data analysis techniques
- Pearson, Kendall and Spearman Correlation

DATASET DESCRIPTION

The dataset used is called jedit-4.3. csv and contains information about the j-edit library. In our project it called df_jedit.

Inside, the data are divided into columns:

- **Name:** dataset name
- **Version:** dataset version
- **Feature_name:** class name
- **wmc:** weighted methods for class - The value of the WMC is equal to the number of methods in the class
- **dit:** depth of inheritance tree - The DIT metric provides for each class a measure of the inheritance levels from the object hierarchy top
- **noc:** number of children - The NOC metric simply measures the number of immediate descendants of the class
- **cbo:** coupling between objects classes - The CBO metric represents the number of classes coupled to a given class.
- **rfc:** response for a class - The RFC metric measures the number of different methods that can be executed when an object of that class receives a message. The value of RFC is the sum of the number of methods called within the class method bodies and the number of class methods.
- **lcom:** lack of cohesion in methods - The LCOM metric counts the sets of methods in a class that are not related through the sharing of some of the class fields.
- **ca:** afferent couplings - The Ca metric represents the number of classes that depend upon the measured class.
- **ce:** efferent couplings - The Ce metric represents the number of classes that the measured class is depended upon.
- **npm:** number of public methods - The NPM metric simply counts all the methods in a class that are declared as public.
- **lcom3:** Lack of cohesion in methods -

$$LCOM3 = \frac{\left(\frac{1}{a} \sum_{j=1}^a \mu(A_j) \right) - m}{1 - m}$$

m - number of methods in a class
a - number of attributes in a class
 $\mu(A)$ - number of methods that access the attribute A

- **loc:** line of code - It is the sum of number of fields, number of methods and number of instructions in every method of the investigated class
- **dam:** data access metric - This metric is the ratio of the number of private (protected) attributes to the total number of attributes declared in the class.
- **moa:** measure of aggregation - This metric measures the extent of the part-whole relationship, realized by using attributes.
- **mfa:** measure of functional abstraction - This metric is the ratio of the number of methods inherited by a class to the total number of methods accessible by the member methods of the class.
- **cam:** cohesion among methods of class - This metric computes the relatedness among methods of a class based upon the parameter list of the methods. The metric is computed using the summation of the number of different types of method parameters in every method divided by a multiplication of the number of different method parameter types in the whole class and number of methods.
- **ic:** inheritance coupling - This metric provides the number of parent classes to which a given class is coupled.

- **cbm**: coupling between methods - The metric measures the total number of new/redefined methods to which all the inherited methods are coupled.
- **amc**: average method complexity - This metric measures the average method size for each class.
- **cc**: McCabe's cyclomatic complexity
- **max_cc**: maximum cyclomatic complexity value
- **avg_cc**: average cyclomatic complexity value
- **bug**: the quantity of fault in a module.

After the dataset has been imported, some preliminary analyses have been made to have a general view of the contents.

The list of attributes, shown above, has been obtained through the *columns* method, while the dataset size, that is number of records and attributes, through the *shape()* method. 492 records and 24 attributes were counted.

More information about attributes such as average, minimum and maximum value or their type, are derived from the *info()* and *describe()* methods respectively.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 492 entries, 0 to 491
Data columns (total 24 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   name        492 non-null    object 
 1   version     492 non-null    float64
 2   feature_name 492 non-null    object 
 3   wmc         492 non-null    int64  
 4   dit         492 non-null    int64  
 5   noc         492 non-null    int64  
 6   cbo         492 non-null    int64  
 7   rfc         492 non-null    int64  
 8   lcom        492 non-null    int64  
 9   ca          492 non-null    int64  
 10  ce          492 non-null    int64  
 11  npm         492 non-null    int64  
 12  lcom3       492 non-null    object 
 13  loc         492 non-null    int64  
 14  dam         492 non-null    float64
 15  moa         492 non-null    int64  
 16  mfa         492 non-null    float64
 17  cam         492 non-null    float64
 18  ic          492 non-null    int64  
 19  cbm         492 non-null    int64  
 20  amc         492 non-null    object 
 21  max_cc      492 non-null    int64  
 22  avg_cc      492 non-null    float64
 23  bug         492 non-null    int64  
dtypes: float64(5), int64(15), object(4)
memory usage: 92.4+ KB
None
```

This screen shows the name of the attribute if it contains null values and its type. We also have general information about the dataset

A verification has been made for a possible presence of null values. The use of the method *isnull().sum()* returns the sum of null values for each attribute. Fortunately, the dataset does not present this problem.

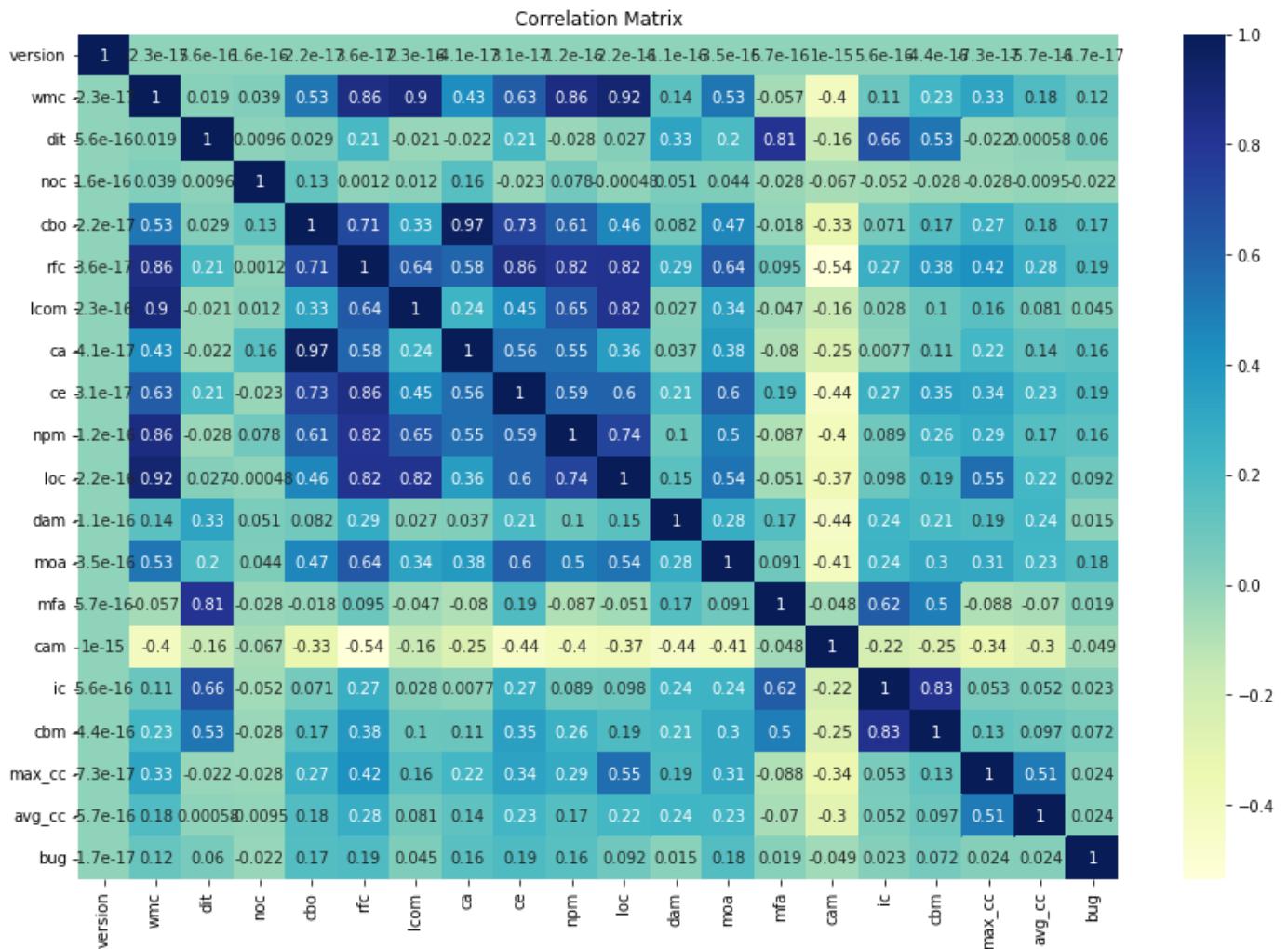
Having only 3 bug classes and 492 records, elements belonging to each class have been counted.

With the `value_counts()` method we know that:

Bug class	Number of elements
0	481
1	10
2	1

Last step is creating a matrix to see the correlation between two variables. The correlation between -1 and 1 can be positive, negative or zero. If the result is positive, there is a proportional relationship between the two variables, while if it is negative the correlation is inverse.

We will use the correlation matrix to see the relationship between two variables



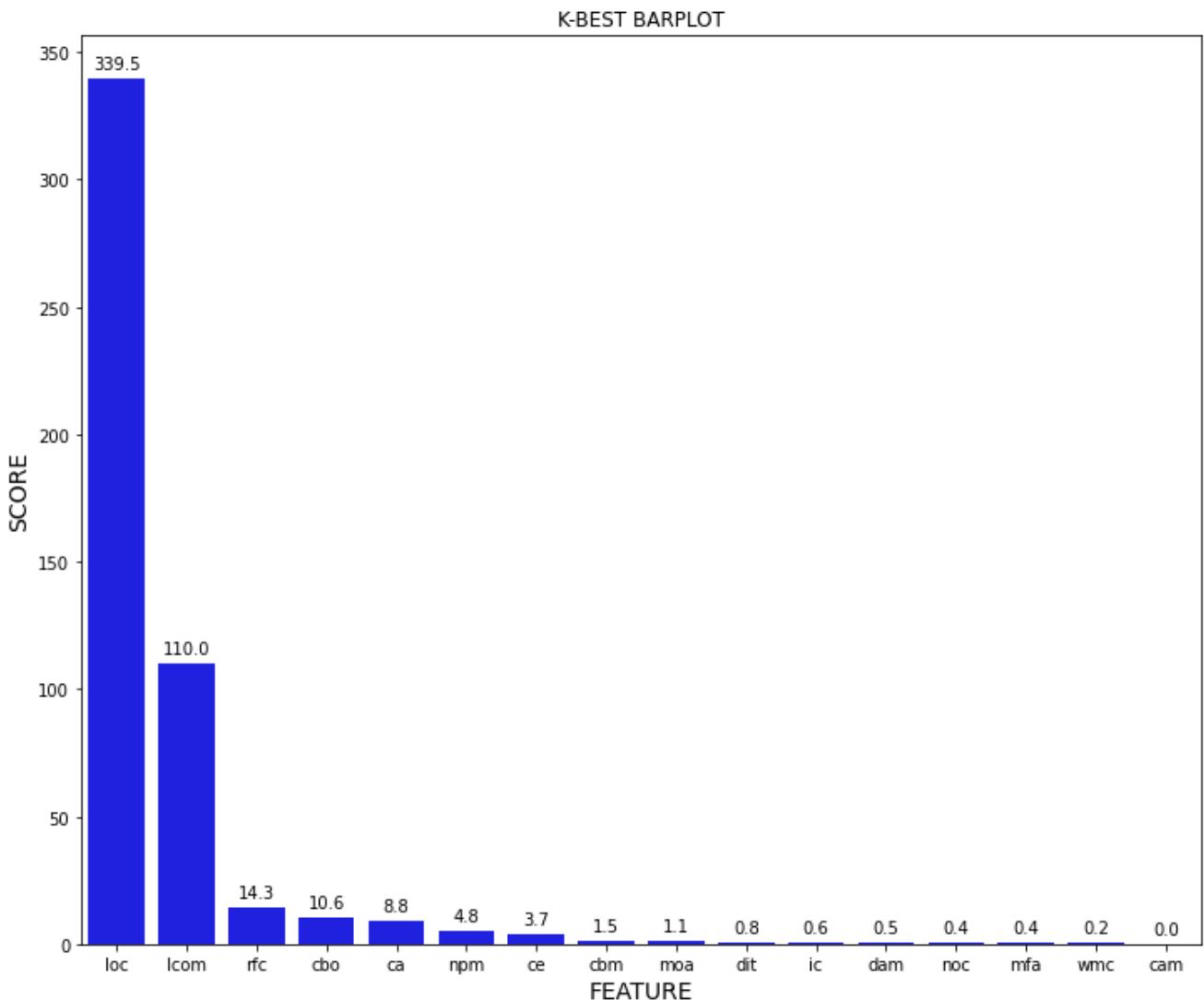
FEATURE SELECTION

For the feature selection operation we relied on the SelectKBest feature selector, coming from the `sklearn.feature_selection.SelectKBest` library. The selector uses chi2, so it computes chi-squared stats between each non-negative feature and class, and it is used to select the n features with the highest values for the test chi-squared statistic.

In our case n=3 because the request requires the use of 3 features as independent variables.

	feature	score	pval
9	loc	339.507727	8.167310e-76
5	lcom	109.991521	9.841076e-26
4	rfc	14.302820	1.556315e-04
3	cbo	10.618041	1.119896e-03
6	ca	8.757637	3.083082e-03
8	npm	4.824414	2.805940e-02
7	ce	3.668591	5.544703e-02
15	cbm	1.501018	2.205147e-01
11	moa	1.081466	2.983697e-01
1	dit	0.791813	3.735524e-01
14	ic	0.596741	4.398241e-01
10	dam	0.497931	4.804105e-01
2	noc	0.448065	5.032552e-01
12	mfa	0.416190	5.188442e-01
0	wmc	0.220430	6.387124e-01
13	cam	0.004968	9.438091e-01

The results show that the three features are: *loc* - *lcom* - *rfc*.



A new dataset was created with the k-best features resulting from the algorithm and it was called `df_kbest`.

	loc	lcom	rfc	bug
0	113	1	16	0
1	82	4	8	0
2	687	0	55	1
3	464	1	36	0
4	213	6	29	0

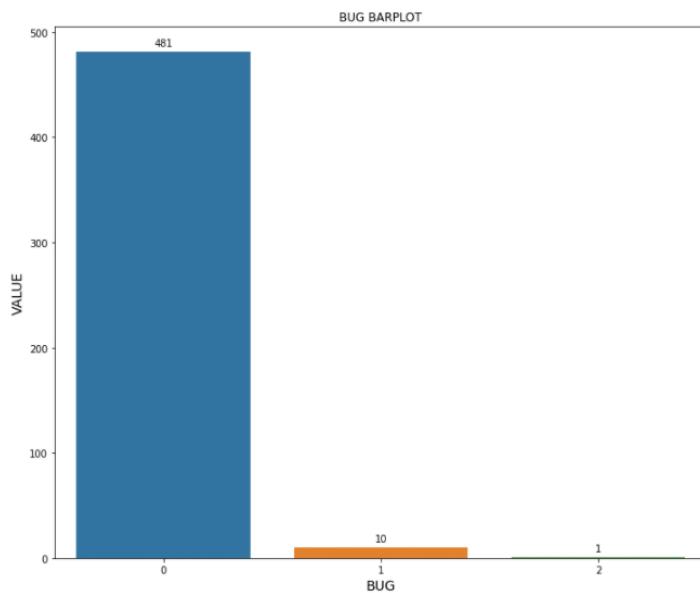
This is only a part of the dataset. It only shows the structure of the new dataset.

GRAPHS

A visual analysis was carried out on the original dataset and on the result of k-best through the following types of graphs:

- **BOXPLOT:** it provides a very useful and standardized way of visualizing the distribution of data based on a summary of five numbers (minimum, first quartile, second quartile (median), third quartile, maximum). It helps to understand these data distribution parameters and is extremely useful for detecting outliers.
- **HISTOGRAM:** is a diagram that provides a representation of a set of statistical data using a bar graph.
- **SCATTER PLOT:** IT is a type of graph that represents the relationship between two variables
- **AREAPLOT:** graphically represents the amount of data, based on a line graph. The area created between the axes and the line, will be colored. This graph represents the quantitative total using numbers or percentages.
- **PIE CHART:** Pie charts can be used to show percentages of a whole and represents percentages at a set point in time.

In particular:

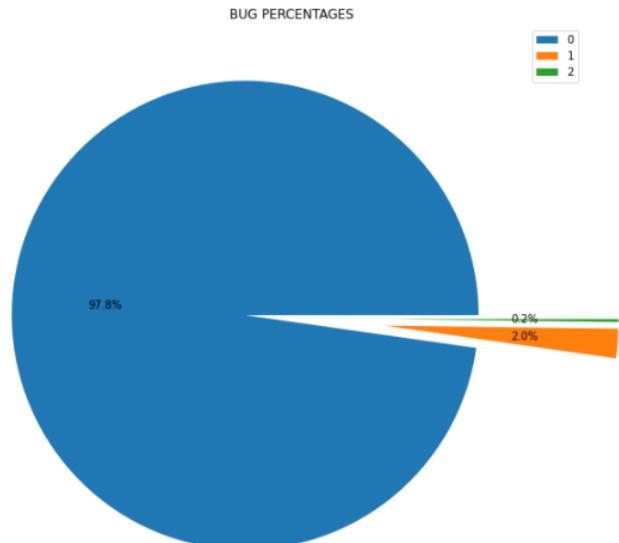


BUG BARPLOT

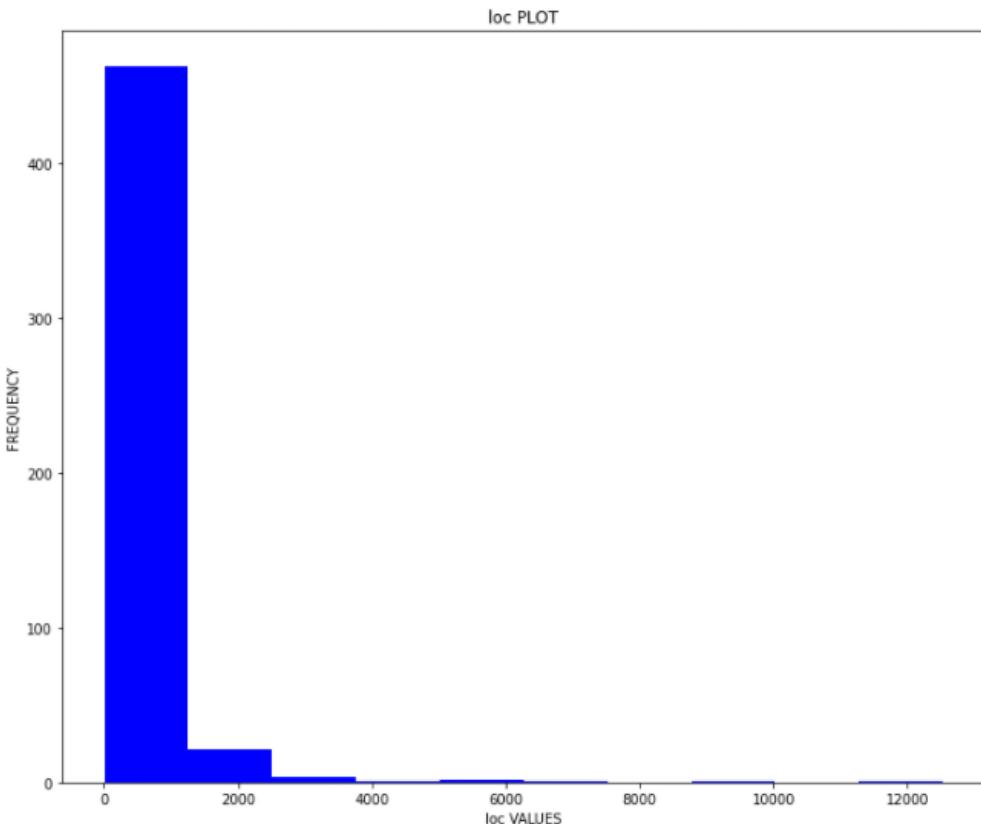
This graph shows us the comparison between the various classes of bugs. The class with 0 bugs has 481 elements. A big gap exists with classes 1 and 2, which have very few elements, respectively 10 and 1.

BUG PERCENTAGE

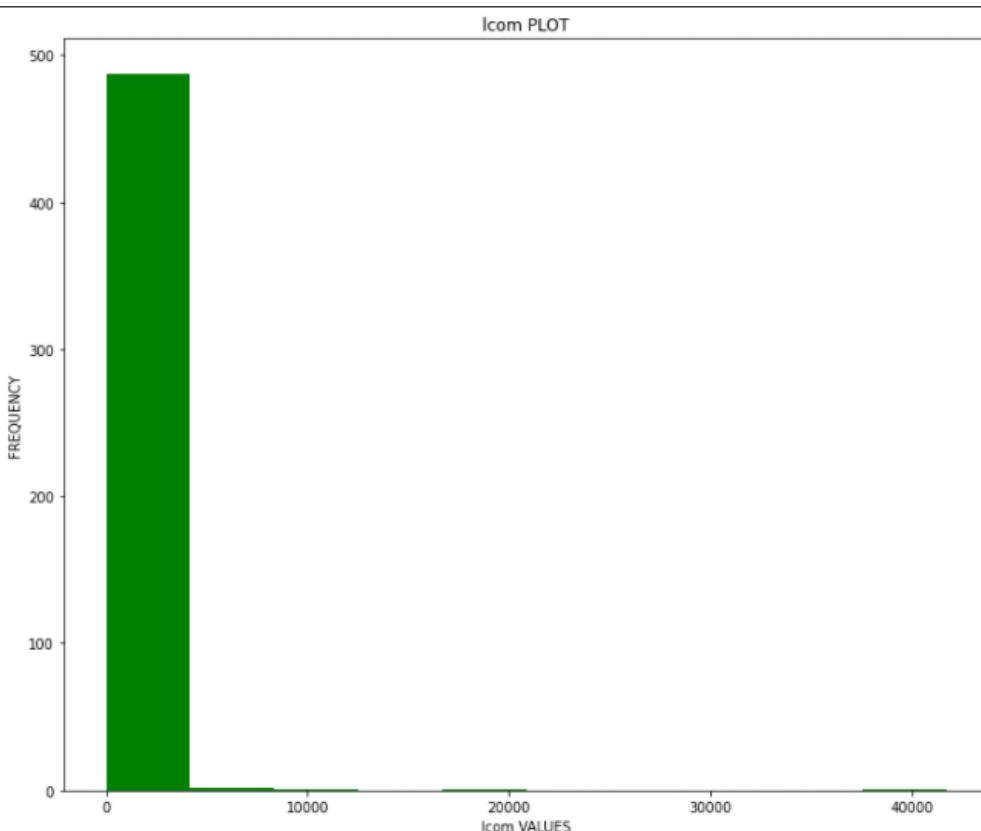
It represents the percentages of each class with respect to the set of elements present in the dataset. Class 0 prevails heavily over the others, with 97.8%. Class 1 represents 2% and class 2 only 0.2%



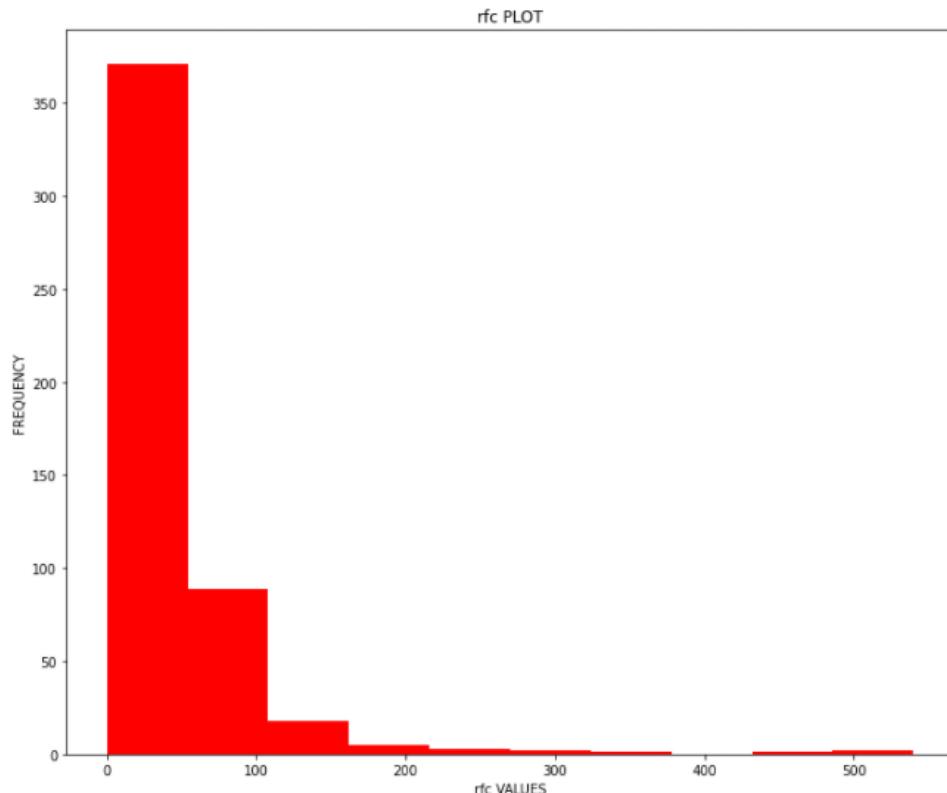
HISTOGRAM



loc PLOT: We begin the hist plot part with the loc attribute. Most of the recorded values, or rather lines of code, fall into the range [0, 1000]. By increasing the number of lines, the presence decreases. Less than 50 elements have a loc number between [1000 -2500]. From the 2500 lines on, there are very few records.



Icom PLOT: The same analysis was done for Icom. In this case a large number of elements have a Icom between 0 and 3000ca. We have a minimum presence for the range [3000 - 8000], for the range [8000 - 12000], [16000 -21000] and for [37000 - 4200].



rfc PLOT: Last histogram concerns the response for a class. The greater frequency of ha in the range 0-50 and even here increasing the range the total decreases.

SCATTER PLOT

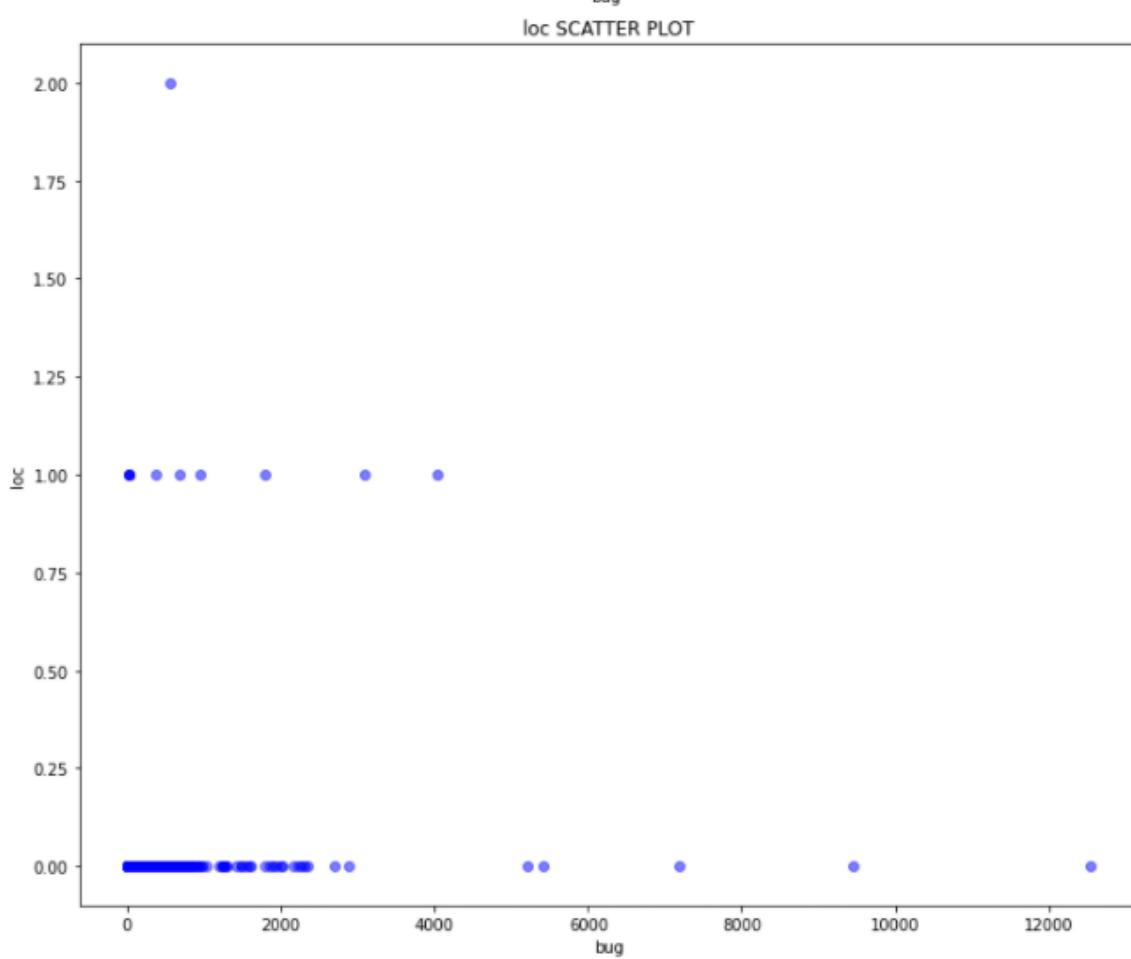
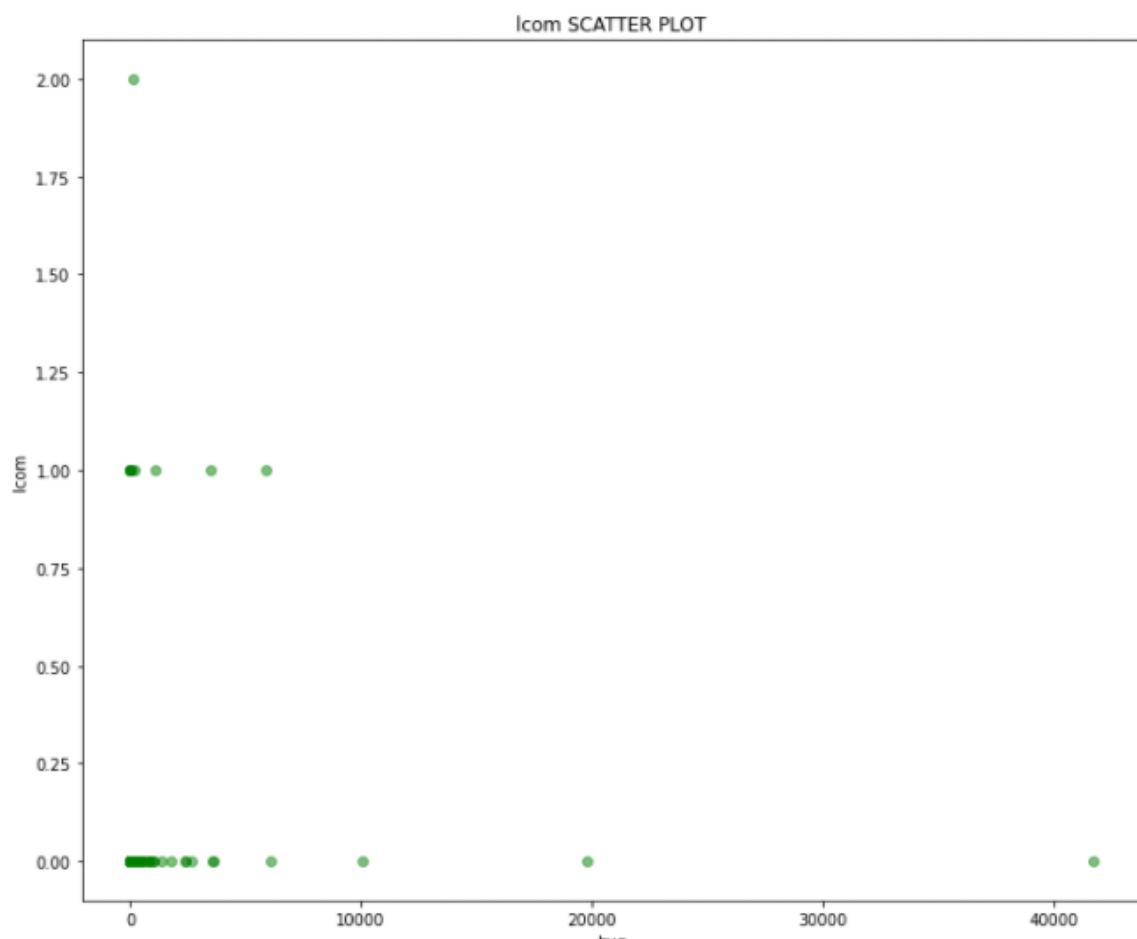
The scatterplot shows the relationships between the k-best features and the bug variable. The darker the color, the higher the presence at that point.

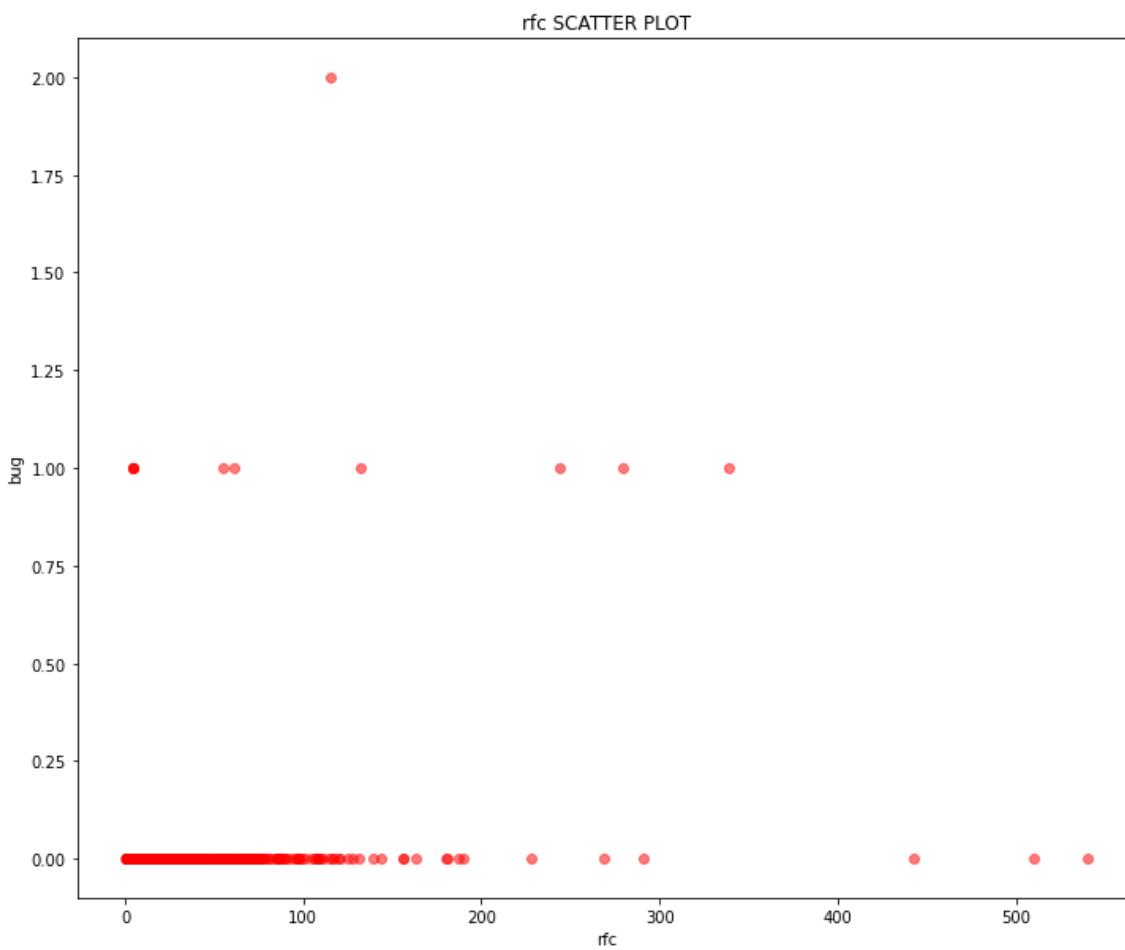
Starting from the first graph (loc SCATTER PLOT), most of the values are concentrated in class 0 with a range between 0 and 2200 ca and in class 1 between 0 and 100 ca.

Sporadic occurrences occur in class 3 (1 element), class 2 over 101 and class 0 after 2500.

The second graph, Icom SCATTER PLOT, shows a situation like the first but with different ranges. The concentration is in class 0 and classe1.

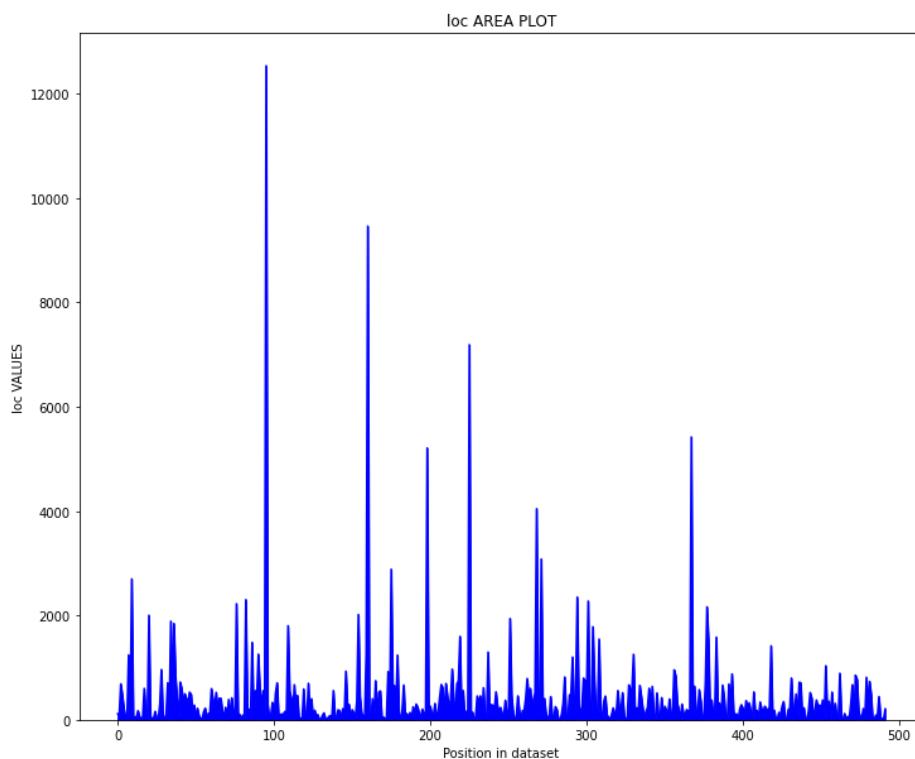
In rfc SCATTER PLOT, the elements are distributed mostly in class 0 between 0 and 150, after which the distribution is smaller. In class 1 the distribution is almost regular.

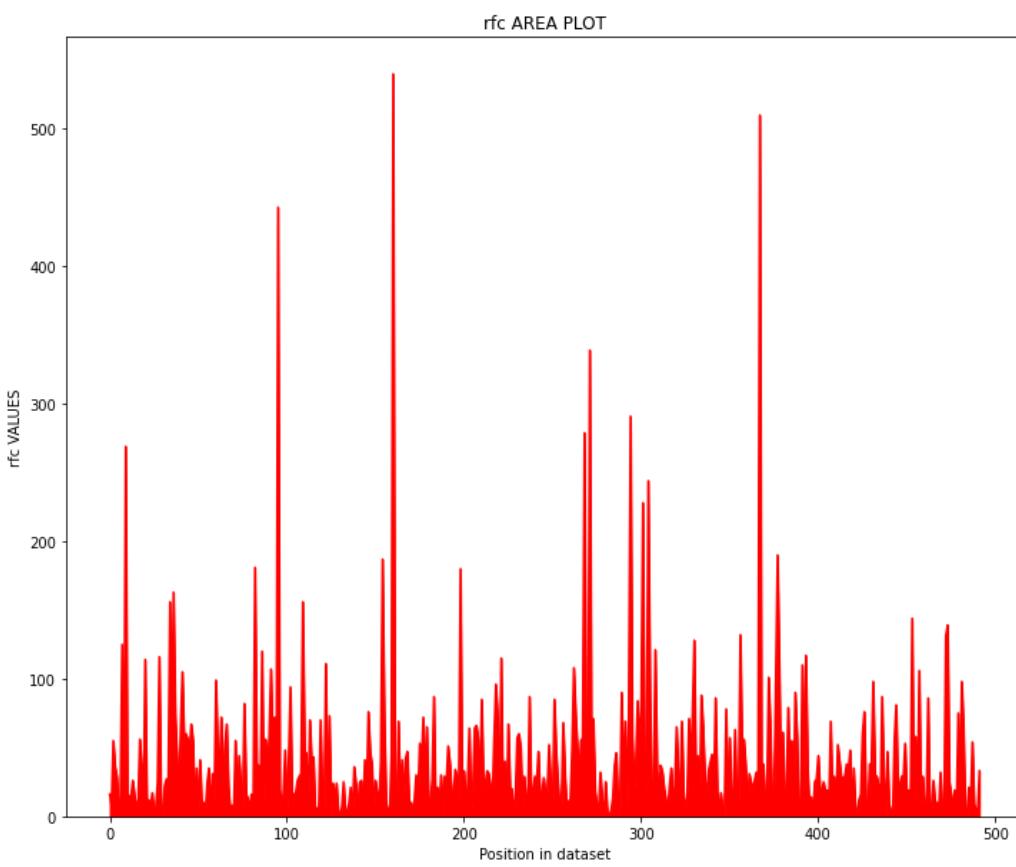
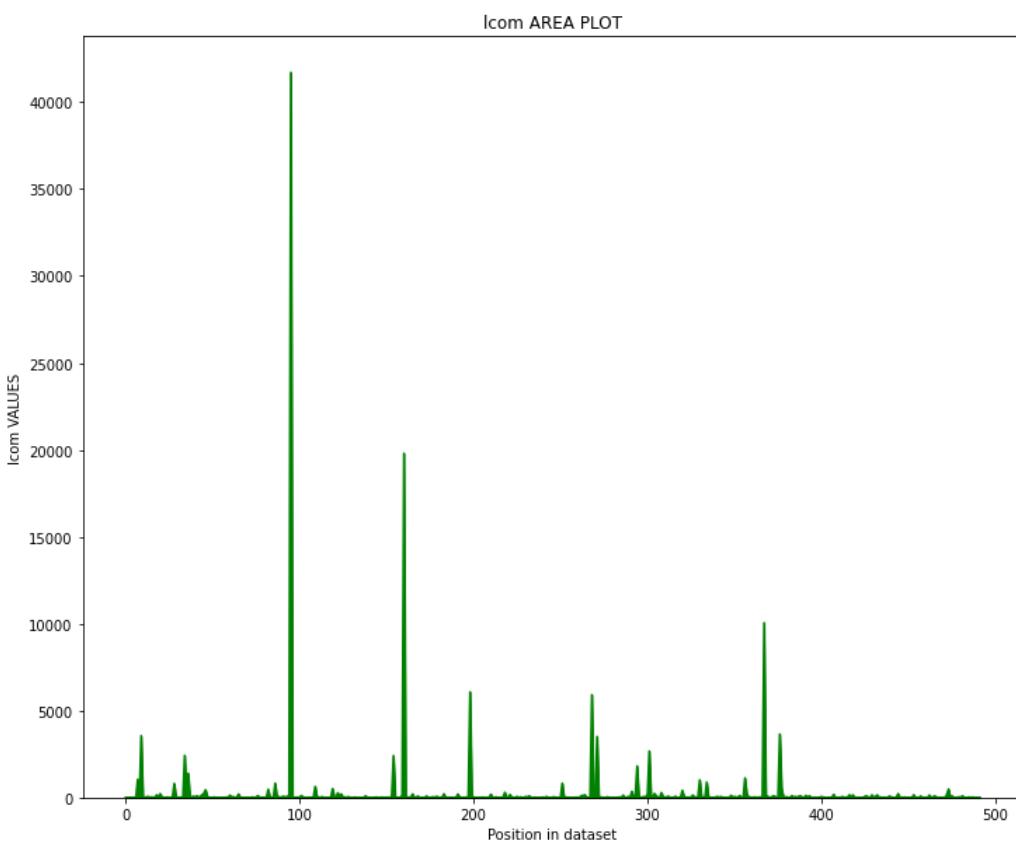




AREA PLOT

Like histograms, they show individual values concurrently with the position in the dataset.

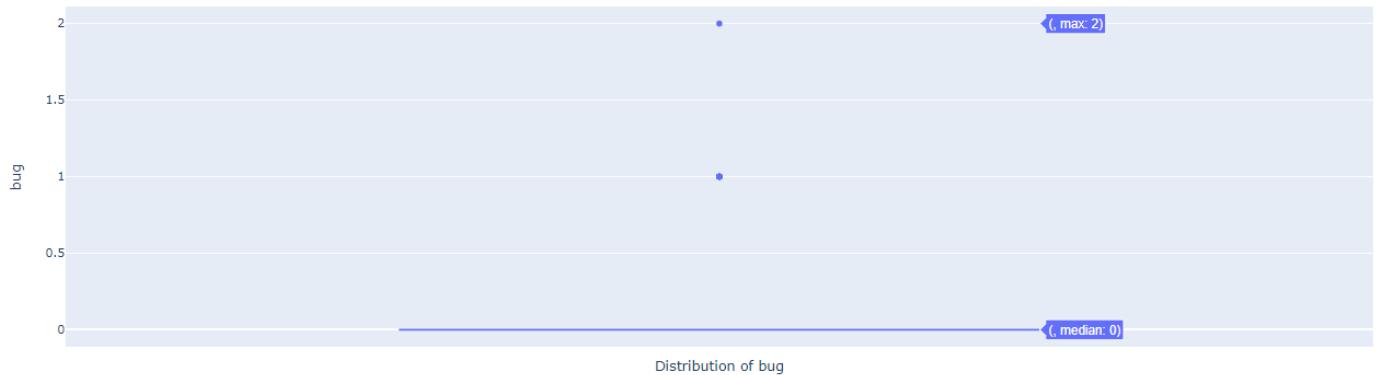




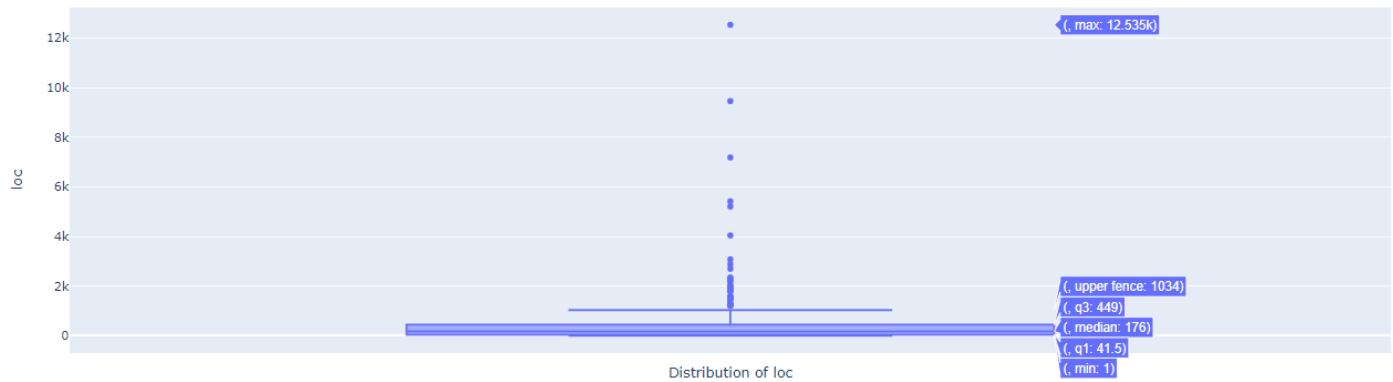
BOXPLOT

Last set of graphs are the boxplots. This chart type, created with the plotly express library, shows the distribution of values, with statistical information such as minimum, maximum, quartiles, and upper/lower fence.

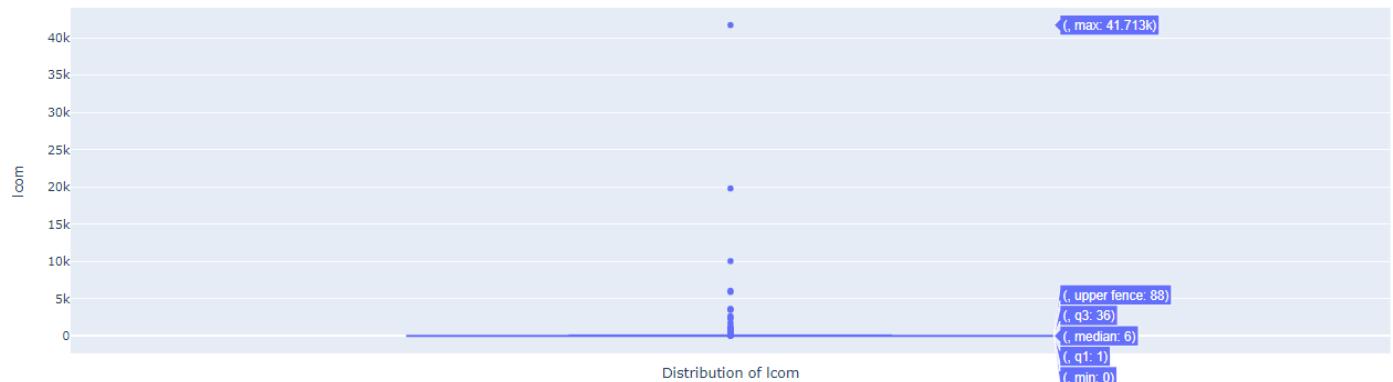
Distribution of bug



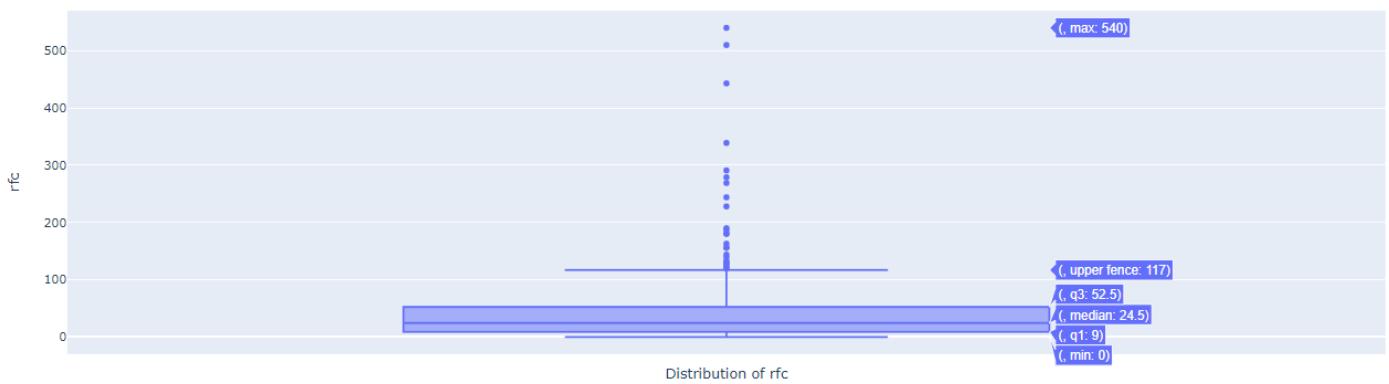
Distribution of loc



Distribution of lcom



Distribution of rfc



DESCRIPTIVE ANALYSIS

Statistics were performed on both the original dataset and the dataset derived from k-best.

On the first one the *describe()* method has been executed. As a result, we have a table that shows various information such as:

- mean
- standard deviation
- minimum value
- 25%
- 50%
- 75%
- maximum value

This is the result on the j-edit dataset with some attributes:

	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	loc
count	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000
mean	12.351626	2.367886	0.447154	14.317073	39.849593	259.906504	8.739837	7.097561	7.851626	411.306911
std	24.512359	1.931839	2.432647	25.004054	56.341461	2184.685294	21.846190	9.112257	15.292942	946.964597
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	3.000000	1.000000	0.000000	4.000000	9.000000	1.000000	1.000000	2.000000	2.000000	41.750000
50%	7.000000	1.500000	0.000000	8.500000	24.500000	6.000000	3.000000	5.000000	4.000000	176.000000
75%	13.000000	3.000000	0.000000	15.000000	52.250000	36.000000	7.000000	9.000000	8.000000	447.000000
max	351.000000	8.000000	38.000000	346.000000	540.000000	41713.000000	291.000000	101.000000	218.000000	12535.000000

On the second dataset the above information was taken. The quartiles were also added.. A quartile in is position index that provides information on the structure of the distribution of a quantitative statistical character.

Specifically, the first, second and third quartiles were calculated.

The execution of the *describe()* method returns the following table :

	loc	lcom	rfc	bug	class
count	492.000000	492.000000	492.000000	492.000000	492.000000
mean	411.306911	259.906504	39.849593	0.024390	0.022358
std	946.964597	2184.685294	56.341461	0.167084	0.147995
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	41.750000	1.000000	9.000000	0.000000	0.000000
50%	176.000000	6.000000	24.500000	0.000000	0.000000
75%	447.000000	36.000000	52.250000	0.000000	0.000000
max	12535.000000	41713.000000	540.000000	2.000000	1.000000

From the manual calculation of the quartiles, we get the following values:

```
The maximum value of loc is: 12535  
The minimum value of loc is: 1  
The avarage of loc is: 411.31  
First quartile value of loc is: 41.75  
Second quartile value of loc is: 176.0  
Third quartile value of loc is: 447.0
```

```
The maximum value of lcom is: 41713  
The minimum value of lcom is: 0  
The avarage of lcom is: 259.91  
First quartile value of lcom is: 1.0  
Second quartile value of lcom is: 6.0  
Third quartile value of lcom is: 36.0
```

```
The maximum value of rfc is: 540  
The minimum value of rfc is: 0  
The avarage of rfc is: 39.85  
First quartile value of rfc is: 9.0  
Second quartile value of rfc is: 24.5  
Third quartile value of rfc is: 52.25
```

DATA ANALYSIS TECHNIQUES

The techniques used are linear regression and KNN

LINEAR REGRESSION

Linear regression allows to predict the value of a variable (called dependent), starting from another variable (defined independent). Linear regression corresponds to a straight line or a surface that minimizes discrepancies between expected and actual output values.

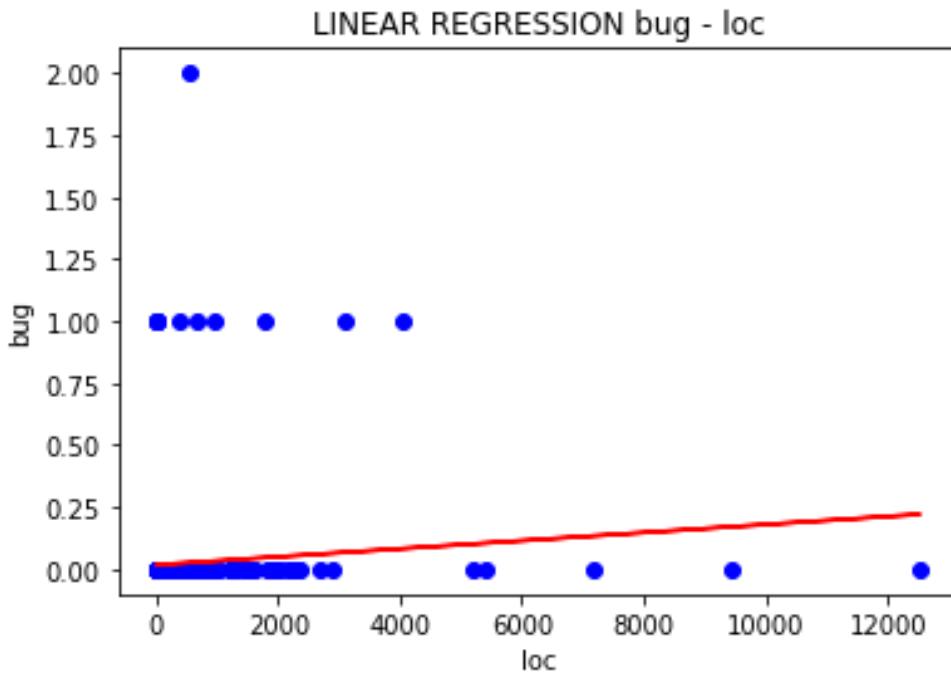
We will comment on the values of the coefficient of determination (R^2), the slope and finally the intercept, taking into consideration the general equation $y = mx + q$.

- The coefficient of determination (denoted by R^2) is a key result of the regression analysis: it is interpreted as the proportion of variance in the dependent variable that is predictable from the independent variable. An R^2 between 0 and 1 indicates the extent to which the dependent variable is predictable.
- The slope indicates the angle of the straight line.
- The intercept indicates how much the line deviates from the 0 point of the y axis, therefore how much it is shifted up or down.

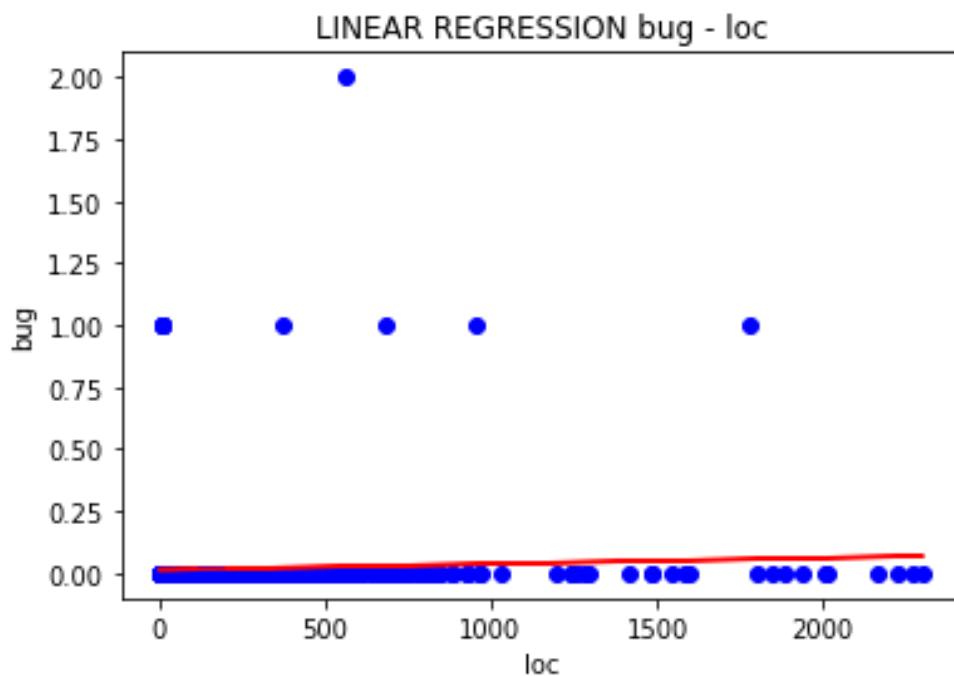
The regression in our project will be performed on the elements of the dataset df_kbest and then on the same elements after the removal of outliers.

The coefficients, slope and intercept are also calculated to create the line and the value of R^2 to assess the accuracy.

BUG – LOC REGRESSION

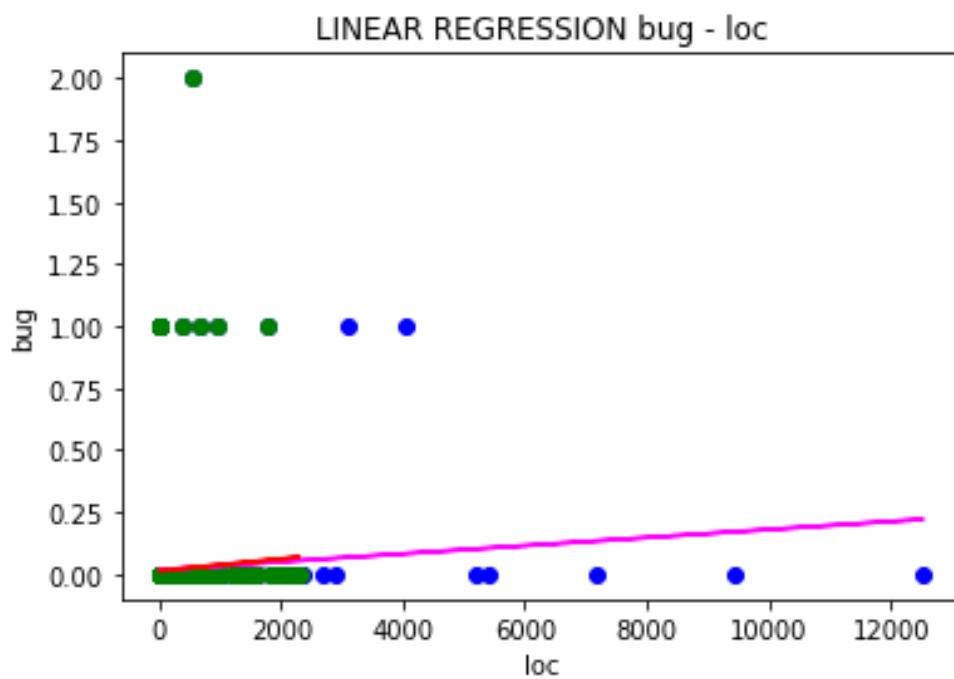


The graph on the original dataset shows that regression increases as loc values increase. The line will have formula $y = 1,62x + 0,017$. The accuracy is 0.85%.



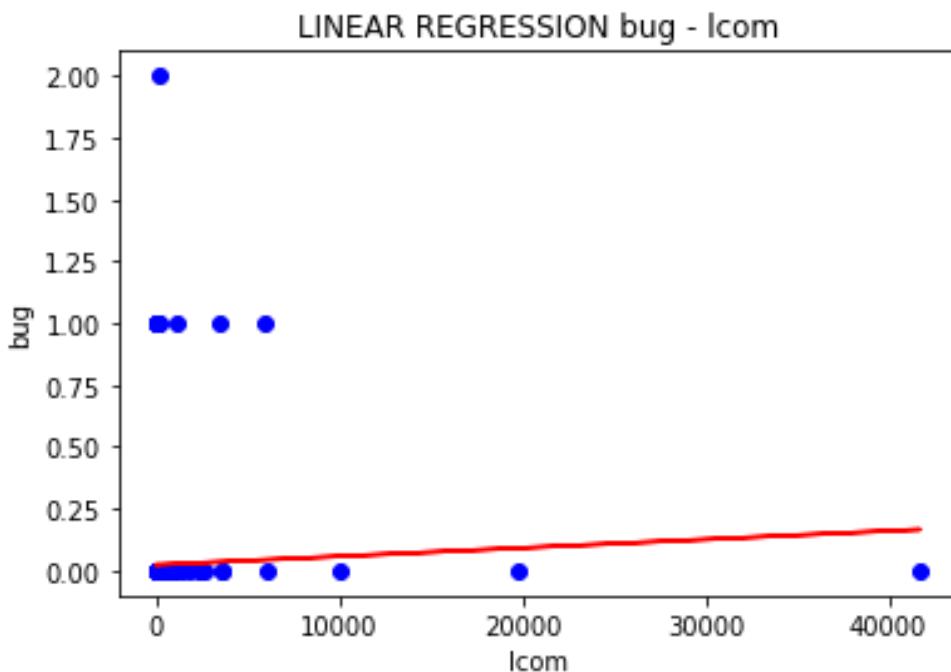
After the outliers have been removed, the direction of the line does not change. Even here it grows, but the angular coefficient is greater than the previous one.

In fact, the line has the formula $y = 2,41x + 0,013$ and the accuracy is 0,3%



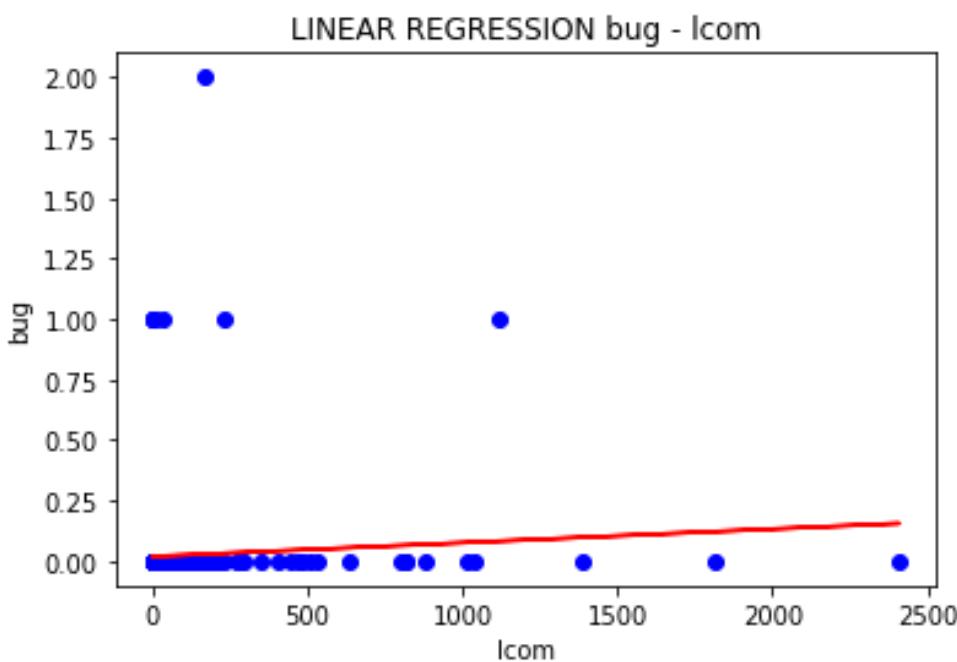
Overlapping the two graphs we notice that the line with the data without outliers (colored red) has a greater angle than the original line (colored pink). The accuracy percentage has decreased.

LCOM – BUG REGRESSION



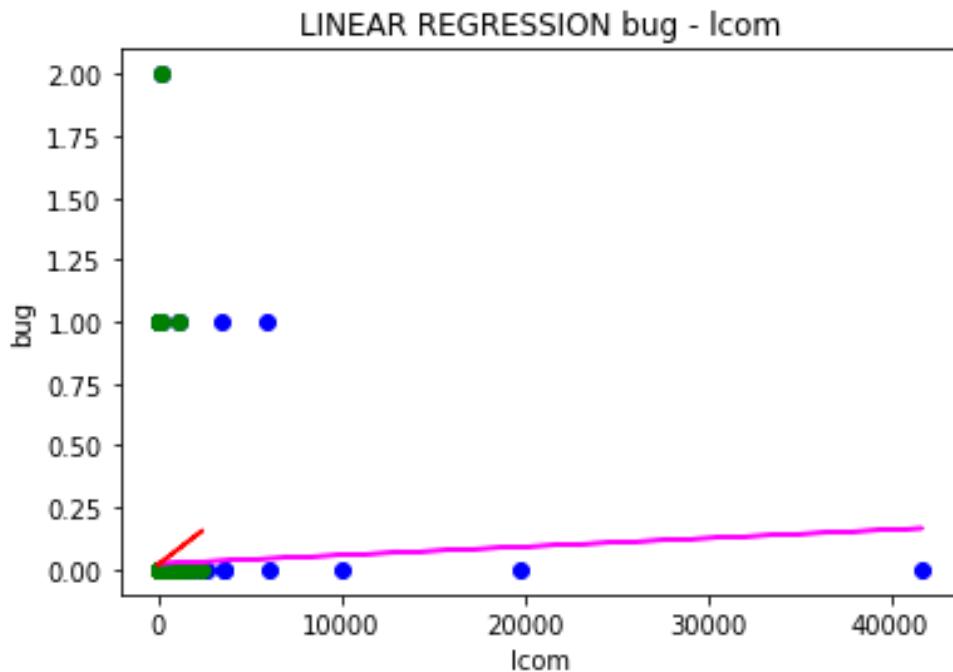
Here again we have a case of rising straight line as the value of lcom increases.

The line has the formula $y = 3,426x + 0,0234$. It has an accuracy of 0,2%



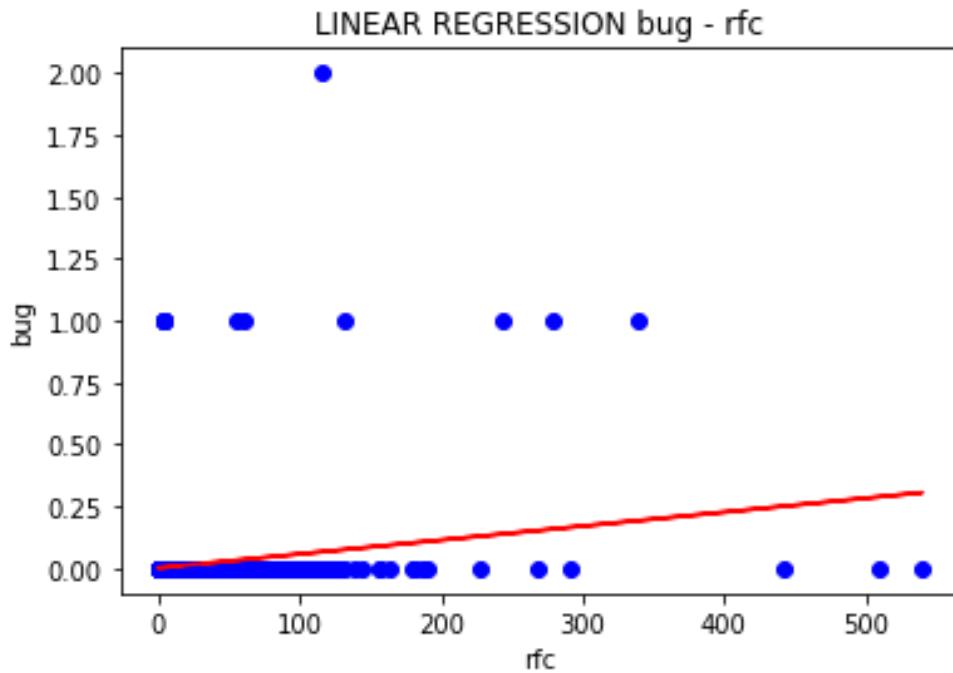
After the removal of the outliers, the straight line shows an increasing trend. We still notice the increase of the angular coefficient. The line is $y = 5,68x + 0,019$. The accuracy increases to 0,5%.

In this case the percentage has changed, increased by 0,3 points.

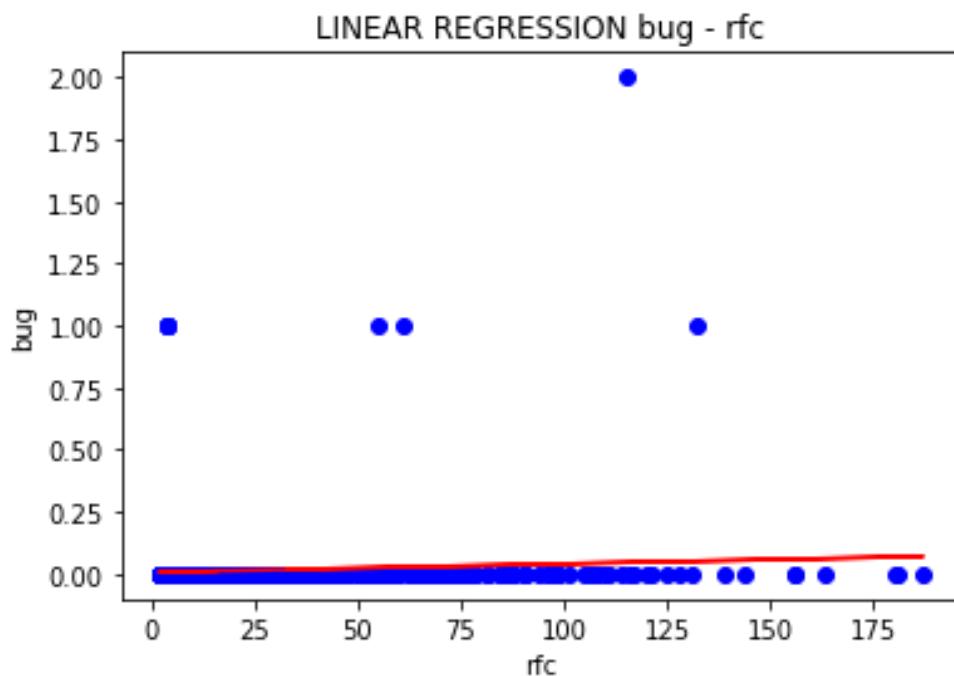


Comparing the two graphs we notice that the straight line deriving from the dataset without outliers (red), grows quickly in contrast to that deriving from the original data (pink).

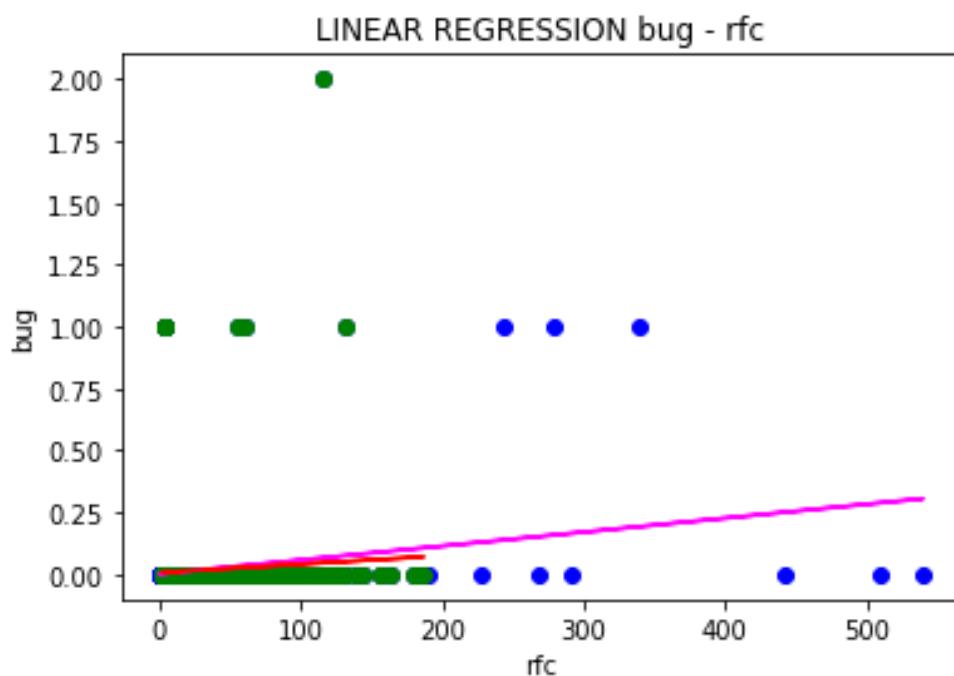
RFC – BUG REGRESSION



In the last set of graphs, we notice an increase of the straight line in the regression between rfc and bug. The formula is $y = 0,00056x + 0,0019$. Unlike the other graphs, the accuracy is higher. We speak of 3.6%



The line in this is $y = 0,00034x + 0.0074$. The accuracy percentage dropped to 0,3%



The union of the two graphs shows a less marked growth of the line after the removal of the outliers (red), compared to the original (pink).

And even then, the percentage has decreased.

K-NEAREST NEIGHBOR (KNN)

The K Nearest Neighbor or KNN is a supervised classification algorithm that associates a given x to the class based on the quantity of elements of a given class in a given range. KNN is also based on Minkowski distances and the most used for this algorithm is the Euclidean distance.

KNN is a lazy learner because it does not learn a discriminative function from the training data but memorizes the training dataset instead.

Given W classes and N supervised training samples. To classify a pattern x , the distance is computed (using predefined metrics) between the pattern x and k nearest samples. The class that has the most elements will be the class associated with the pattern.

ADVANTAGES	DISADVANTAGES
Easy to design and understand	Computational time at test time
No time to train	

Instead of Minkowski's metrics, we use Euclidean metric.

At the beginning we divided the dataset into a training set and test set, with a proportion of 70% and 30%.

```
● ● ●  
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.30)
```

Once the classification model (KNN) has been defined, we train the model with the train set and make the prediction on the test set.

We finish by printing the confusion matrix, the classification report and the accuracy value.

VALIDATION MODEL

We finish by printing the confusion matrix, the classification report, and the accuracy value.

- Confusion matrix:

The confusion matrix is a summary of the prediction results on a classification problem. In this matrix, the number of correct and incorrect predictions are summarized with count values and broken down for each class to understand the ways in which the ranking model is confused when making predictions.

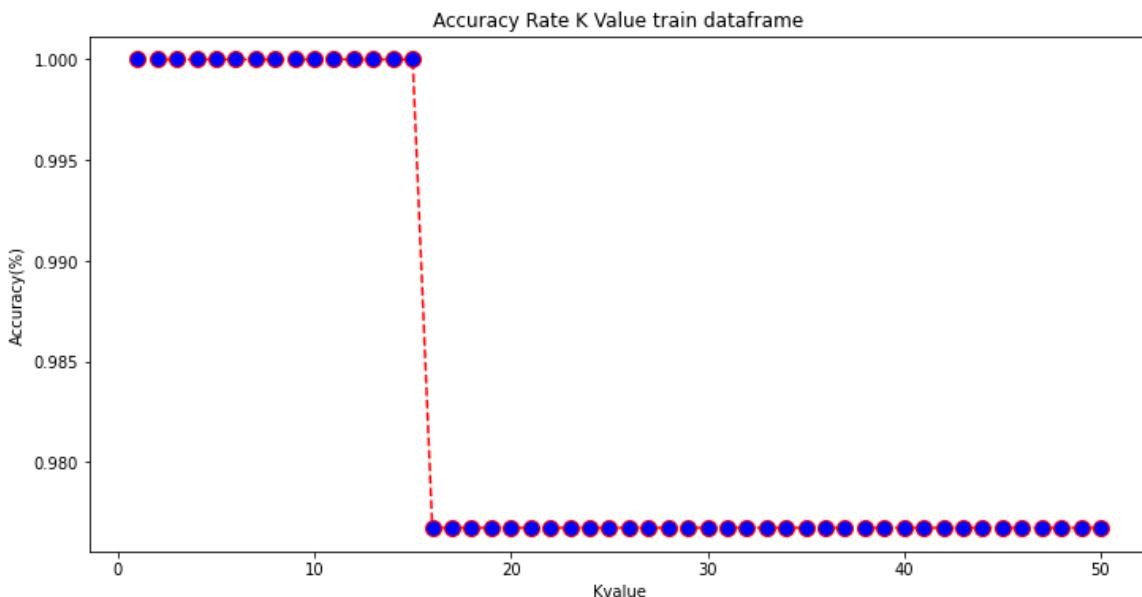
- Classification report

It provides us with the following parameters:

- **Precision** is the ability of the classifier not to label as positive a sample that is negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.
- **Recall** indicates the percentage of true positives and the value. It is the ability of a classifier to find all positive instances.
- The **F1 score** is a weighted harmonic average of the two previous metrics such that the best score is 1.0 and the worst is 0.0
- **Support** is the number of actual occurrences of the class in the specified dataset.
- **Accuracy**: by evaluation we mean the count of test records that are correctly (or incorrectly) predicted by the classification model.
- **Macro-AVG**, says the function to compute f1 for each label, and returns the average without considering the proportion for each label in the dataset
- **Weighted-AVG**, it calculates the F1 score for each class independently, but when adding them it uses a weight that depends on the number of true labels in each class.

VARIATION OF K

We applied KNN with k increasing from 1 to 51 inclusive. We thus obtain $\langle \text{accuracy} - k \rangle$ pairs and we have analyzed a general case ($k=5$), the worst case and the best case. This is implemented to see how the classifier performs in various scenarios.

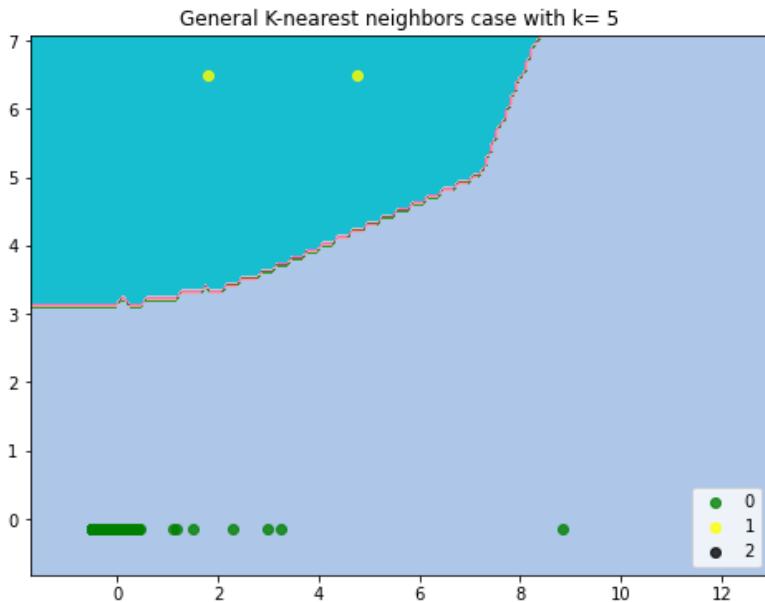


ANALYSIS RESULTS

- GENERAL CASE (k=5)

With a general case k=5 we have excellent results. Accuracy is close to 100% (0.99). Of the 148 samples selected, only one was placed in the wrong class.

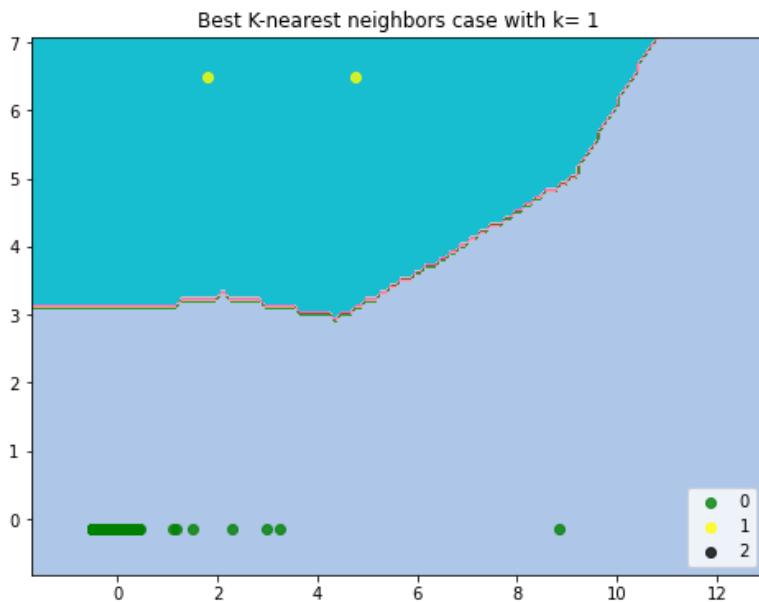
[[145 0 0]
[0 2 0]
[0 1 0]]
precision recall f1-score support
0 1.00 1.00 1.00 145
1 0.67 1.00 0.80 2
2 0.00 0.00 0.00 1
accuracy 0.99 148
macro avg 0.56 0.67 0.60 148
weighted avg 0.99 0.99 0.99 148



BEST CASE (k = 1)

The results are similar to the previous one. The accuracy levels are maximum when k takes the value 1. The accuracy is maximum: it is 99%. In fact, there is only one wrong classification. It is a class 2 element classified in class 1.

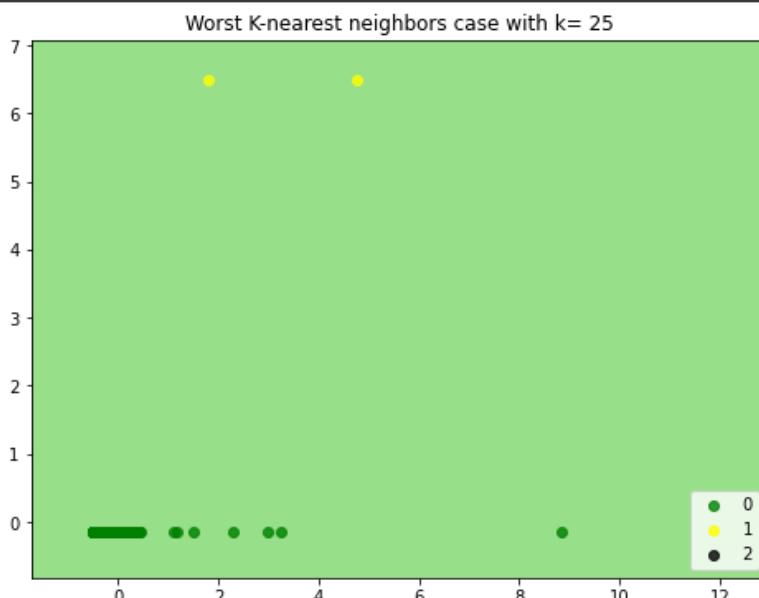
[[145 0 0]
[0 2 0]
[0 1 0]]
precision recall f1-score support
0 1.00 1.00 1.00 145
1 0.67 1.00 0.80 2
2 0.00 0.00 0.00 1
accuracy 0.99 148
macro avg 0.56 0.67 0.60 148
weighted avg 0.99 0.99 0.99 148



WORST CASE (k = 25)

With k = 25 there is a loss of accuracy, but not such as to deviate much from the previous k. The elements of class 1 and class 2 have been inserted in class 0. This results in an accuracy of 98% and a graph with only one class, without edges that outlines it.

[[145 0 0]
[2 0 0]
[1 0 0]]
precision recall f1-score support
0 0.98 1.00 0.99 145
1 0.00 0.00 0.00 2
2 0.00 0.00 0.00 1
accuracy
macro avg
weighted avg



OUTLIERS MANAGEMENT

Inside the dataset, the distribution of the values is very wide, that is it has a very wide range of values. This could create problems while running algorithms because some recordings may be anomalies. To get a more homogeneous set we must remove some data.

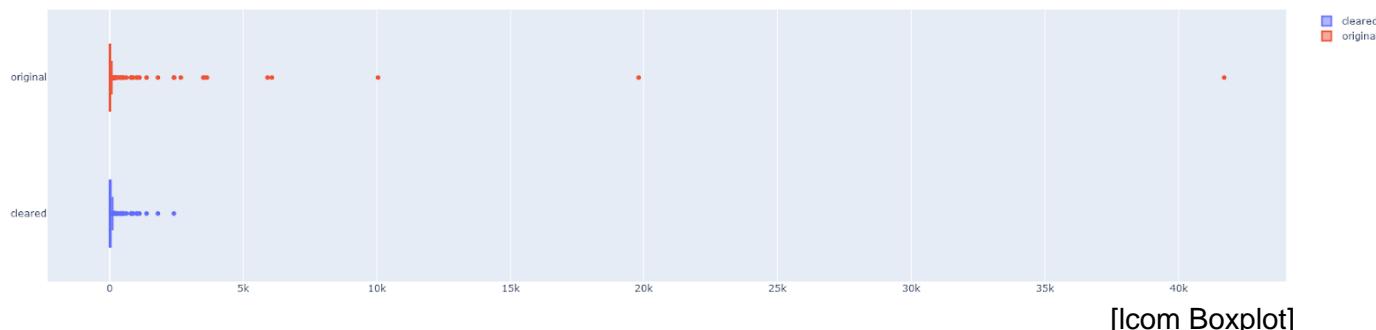
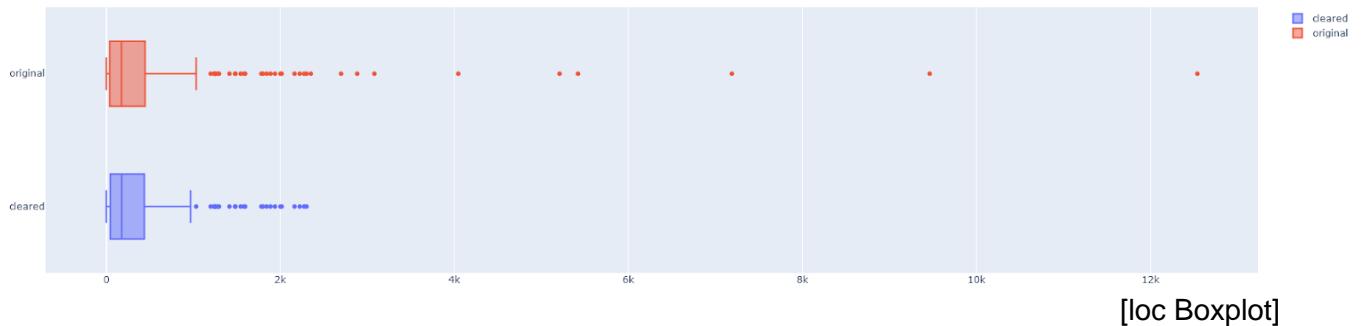
To do this we define two quartiles (Q1 and Q3) and we only keep the values within this range (interquartile discard - IQR).

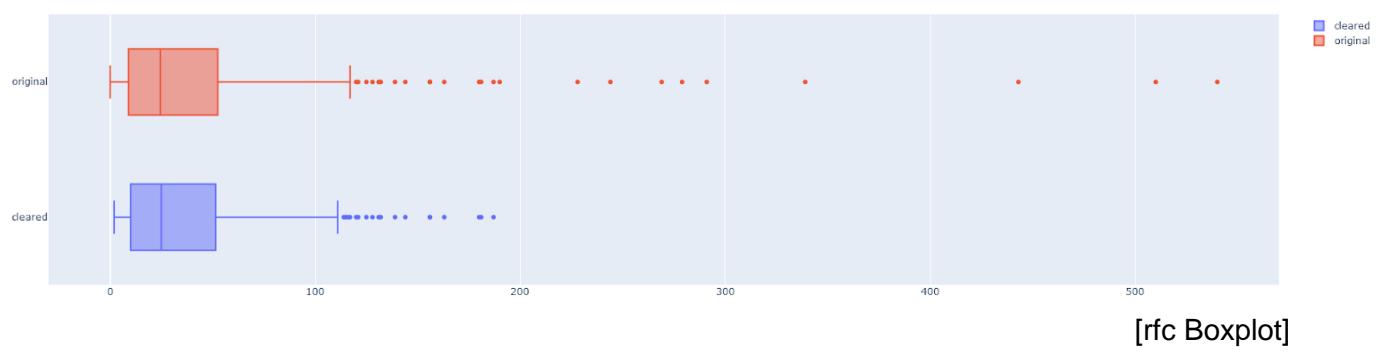
In our case, 3 ad hoc datasets were created, with the pairs <loc - bug>, <lcom - bug> and <rfc - bug>, and the outliers were removed.

Specifically, the limit values are a calculation that includes IQR, Q1, Q3 and a constant.

```
● ● ●  
data_clean = df[(df.iloc[:,0] > (Q1 - 3 * IQR)) & (df.iloc[:,0] < (Q3 + 3 * IQR))]
```

After performing this skim, using a boxplot, I show the range before and after. This operation was done for each independent variable.





PEARSON, SPEARMAN, KENDALL STATISTIC

Before arriving at conclusions, we analyze the dataset with univariate or multivariate statistics.

Per fare ciò, eseguiremo la correlazione con la tecnica di Spearman, Kendall e Pearson.

- Pearson Correlation

Pearson r correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables.

$$r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

r_{xy} = Pearson r correlation coefficient between x and y

n = number of observations

x_i = value of x (for ith observation)

y_i = value of y (for ith observation)

The range of values is -1 to 1. If the correlation is close to 0 then the relationship is weak, while if it approaches 1 or -1, it will be strong

- Spearman Correlation

Spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. The Spearman rank correlation test does not carry any assumptions about the distribution of the data and is the appropriate correlation analysis when the variables are measured on a scale that is at least ordinal.

The results range from -1 to 1. If the index is close to zero, the relationship will be neutral. But if the index is close to 1 or -1, we have a positive correlation if the result will be closed to 1 and negative if it is closed to -1.

$$\rho_s = 1 - \frac{6 \sum_i D_i^2}{N(N^2 - 1)}$$

$D_i = r_i - s_i$ = difference of the ranges

N = number of observations

- Kendall Correlation

Kendall rank correlation is a non-parametric test that measures the strength of dependence between two variables.

When the values of both variables increase it, there is a positive correlation and consequently a positive result. When you have a negative result, it means that the increase of one variable leads to a descrescence of the other. We speak of negative correlation.

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\binom{n}{2}}.$$

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

is the binomial coefficient

The numerator is the difference between the number of equal values and the number of different values

In our project the correlation takes place through a fixed variable (Bug) and the remaining ones.

The table below derives from the original values

	Feature	Pearson	Spearman	kendall
0	loc	0.092207	0.062697	0.051132
0	lcom	0.044798	0.062384	0.052489
0	rfc	0.189912	0.070574	0.057876

In this case:

Pearson and Spearman claim a good correlation between variables as the results tend to zero. Kendall tells us that all three variables have a positive correlation and therefore if the value of Bug increases, their value will also increase.

But when we remove the outliers, the situation changes

	Feature	Pearson	Spearman	kendall
0	loc	0.060975	0.016831	0.013612
0	lcom	0.074201	-0.004444	-0.003976
0	rfc	0.074955	-0.002995	-0.002595

From the table we note that there is still a strong correlation between bugs and the three variables considered.

Unlike previous results, Spearman is negative for lcom and rfc.

With Kendall it remains a positive correlation for the loc variable, but it changes for lcom and rfc. Now these two variables turn out to have a negative correlation.

CONCLUSION

The proposed dataset contains information about the jedis library 4.3.

The descriptive analysis of the dataset showed us the composition of the dataset (492 records and 24 attributes) and that it does not contain null values.

The visualization of the data has been possible using different types of diagrams.

Then linear regression was applied to see how far the data deviates from the mean value and the KNN gave us an idea of how the records are classified.

The removal of outliers, led to the creation of datasets with values within a limited range, eliminating any anomalies.

Finally, the correlation was performed through the techniques of Pearson, Spearman and Kendall to understand the correlation of independent variables with dependent.

From regression with and without outliers, you can see a positive relationship between the variables.

But with Spearman, Kendall and Pearson's correlation, the situation changes. With the original data the correlation remains positive, but by removing the possible anomalies, a negative correlation with the lcom and rfc variables occurs using the Kendall and Spearman techniques.