UNIVERSITA' DEGLI STUDI DELL'INSUBRIA
DIPARTIMENTO DI SCIENZE TEORICHE E APPLICATE
(DiSTA)
CORSO DI LAUREA MAGISTRALE IN INFORMATICA

SOFTWARE QUALITY EVALUATION PROJECT

RELAZIONE DI:
Vladi VALSECCHI matr 730030

ANNO ACCADEMICO 2021/2022

Summary

Introduction

In this project, the analysis of the ant-1.7.csv dataset will be carried out using two regression models and various metrics. The results obtained with the various metrics will then be analyzed. The purpose is to understand if there are limitations on the use of the data contained in the dataset. The project contains a main file made with Google Colab in Python language (.ipybn format) containing all the functions and the data analysis performed. Various methods of choice of features have been implemented in the project which were subsequently used for the analysis.

The project is divided into 7 sub-objectives:

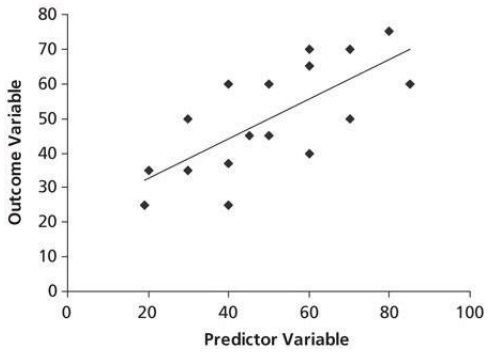
- Selection of the dataset (ant-1.7)
- Selection of a dependent variable (bug)
- Selection of three independent variables
- Data visualization
- Computation of descriptive statistics
- Application of data analysis techniques
- Correlation with Pearson’s, Kendall’s and Spearman’s methods

Used techniques

Linear regression

Linear regression is an approach for modeling the relationship between two (simple linear regression) or more variables (multiple linear regression). In simple linear regression, one variable is considered the predictor or independent variable, while the other variable is viewed as the outcome or dependent variable.

Linear regression equation: $Y=b_0+b_1x_1+b_2x_2+\ldots+b_nx_n$



In the picture, you can see a linear relationship. That is, if one independent variable increases or decreases, the dependent variable will also increase or decrease.
Linear regression can be used to make simple predictions such as predicting exams scores based on the number of hours studied, the salary of an employee based on years of experience, and so on.

The coefficient of determination (denoted by R2), 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 R2 between 0 and 1 indicates the extent to which the dependent variable is predictable.

Logistic regression

Logistic Regression is a Machine Learning algorithm used to make predictions to find the value of a dependent variable such as the condition of a tumor (malignant or benign), classification of email (spam or not spam), or admission into a university (admitted or not admitted) by learning from independent variables (various features relevant to the problem).

For example, for classifying an email, the algorithm will use the words in the email as features and based on that make a prediction whether the email is spam or not.

Logistic Regression is a supervised Machine Learning algorithm, which means the data provided for training is labeled i.e., answers are already provided in the training set. The algorithm learns from those examples and their corresponding answers (labels) and then uses that to classify new examples.

In mathematical terms, suppose the dependent variable is Y and the set of independent variables is X, then logistic regression will predict the dependent variable P(Y=1) as a function of X, the set of independent variables.

Evaluation of the logistic regression values

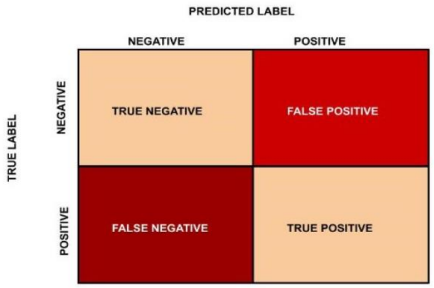
For the evaluation of the performance of the Logistic Regression model, we relied on accuracy. Accuracy is the number of correctly classified data instances out of the total number of data instances.

$$accuracy = \frac{true\ negative\ +\ true\ positive}{true\ negative\ +\ true\ positive\ +\ false\ positive\ +\ false\ negative}$$

We also made use of the *confusion_matrix* and *classification_report* functions. Classification report provided us with the following parameters:

- Precision, is the ability of the classifier not to label as positive a sample that is negative.

$$precision = \frac{true\ positive}{true\ positive\ +\ false\ positive}$$



Example of true/false positive/negative

- Recall, indicates the percentage of true positives and the value;

$$recall = \frac{true\ positive}{(true\ positive\ +\ false\ negative)}$$

- F1-Score, is the harmonic mean between precision and recall and is a better measure of precision. The value fluctuates between 0 and 1;

$$f1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Support, is the number of occurrences of each class in y_test;
- Accuracy;
- Macro-AVG, calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.
- Weighted-AVG, calculate metrics for each label, and find their average weighted by support (the number of true instances for each label).

```
[[10  1  0]
 [ 1 15  0]
 [ 0  3  6]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        11
           1       0.79      0.94      0.86        16
           2       1.00      0.67      0.80         9

    accuracy                           0.86        36
   macro avg       0.90      0.84      0.86        36
weighted avg       0.88      0.86      0.86        36
```

Example of output from the two functions

Correlation coefficient

A correlation coefficient is a numerical measure of some type of correlation, meaning a statistical relationship between two variables. The variables may be two columns of a given data set of observations, often called a sample, or two components of a multivariate random variable with a known distribution.

Several types of correlation coefficient exist, each with their own definition and own range of usability and characteristics. They all assume values in the range from −1 to +1, where ±1 indicates the strongest possible agreement and 0 the strongest possible disagreement.

Pearson's R

Pearson's R is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalized measurement of the covariance, such that the result always has a value between −1 and 1. As with covariance itself, the measure can only reflect a linear correlation of variables, and ignores many other types of relationship or correlation.

$$\mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{xy}) - (\sum \mathbf{x})(\sum \mathbf{y})}{\sqrt{[\mathbf{n}\sum \mathbf{x^2} - (\sum \mathbf{x})^2][\mathbf{n}\sum \mathbf{y^2} - (\sum \mathbf{y})^2]}}$$

Spearman's RHO

Spearman's RHO assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not). If there are no repeated data values, a perfect Spearman correlation of +1 or −1 occurs when each of the variables is a perfect monotone function of the other.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

Kendal’s TAU

Kendal’s TAU is a statistic used to measure the ordinal association between two measured quantities. A τ test is a non-parametric hypothesis test for statistical dependence based on the τ coefficient.

It is a measure of rank correlation: the similarity of the orderings of the data when ranked by each of the quantities.

Kendall correlation between two variables will be high when observations have a similar (or identical for a correlation of 1) rank between the two variables, and low when observations have a dissimilar (or fully different for a correlation of −1) rank between the two variables.

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

Dataset description

This dataset contains 745 elements and their mesurements.

The csv file contains 24 fields:

- name: name of the source
- version: version of the source
- name.1: class name
- wmc: weighted methods for class
- dit: depth of inheritance tree noc: number of children
- cbo: coupling between objects classes
- rfc: response for a class
- lcom: lack of cohesion in methods
- ca: afferent couplings
- ce: efferent couplings
- npm: number of public methods
- lcom3: loc: line of code
- dam: data access metric
- moa: measure of aggregation
- mfa: measure of functional abstraction
- cam: cohesion among methods of class
- ic: inheritance coupling
- cbm: coupling between methods
- amc: average method complexity
- cc: mcCabe’s cyclomatic complexity
- max_cc: max cyclomatic complexity
- avg_cc: average cyclomatic complexity
- bug: # of bugs

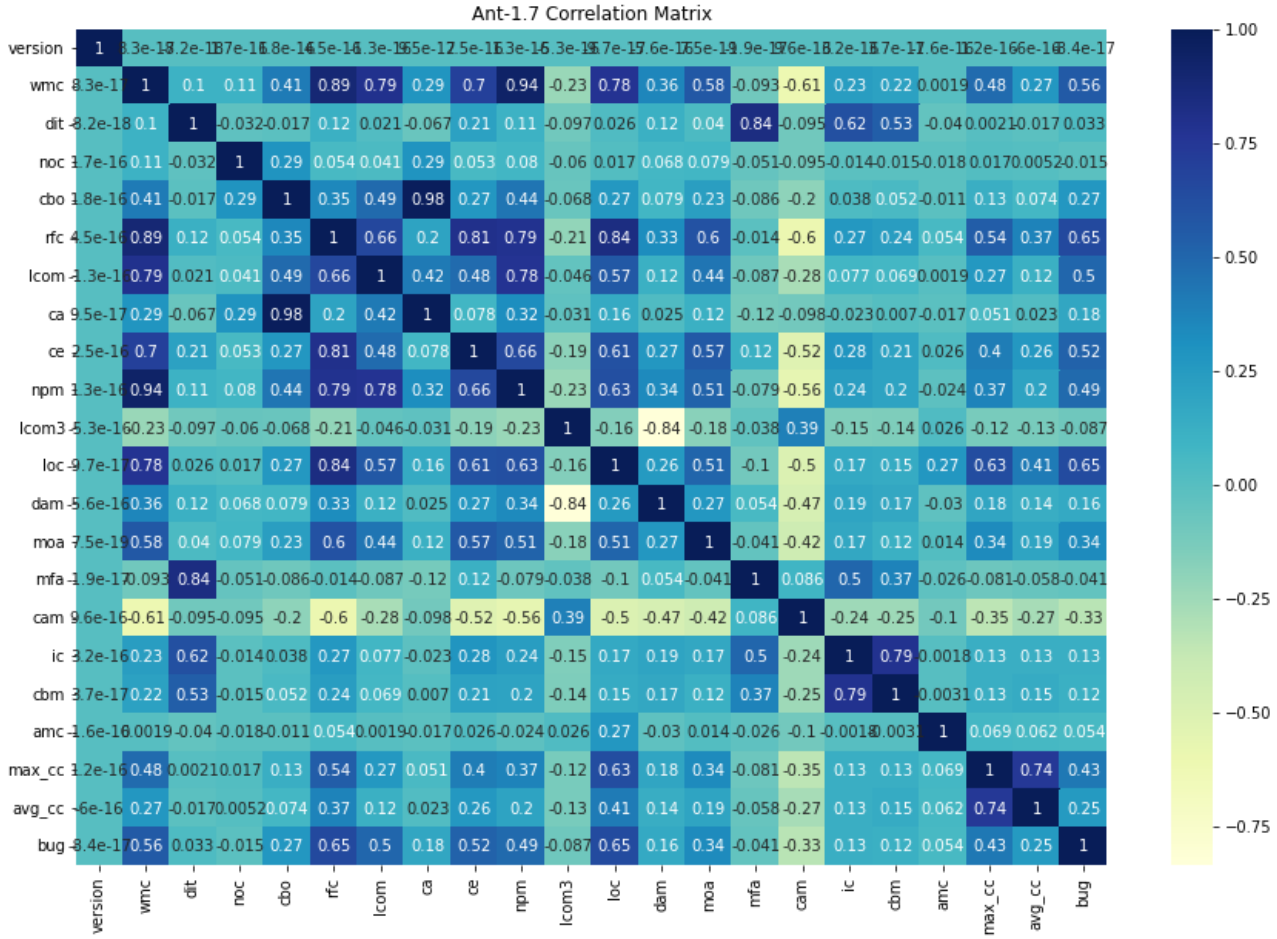
The dataset don’t contain null values.

Feature selection

The dependent feature chosen is "bug".
Two approaches were used for the choice of independent features, correlation matrix approach and k-best approach.

Correlation matrix approach

The first step is to create a correlation matrix in order to evaluate the degree of correlation between the possible independent features and the bug dependent feature.



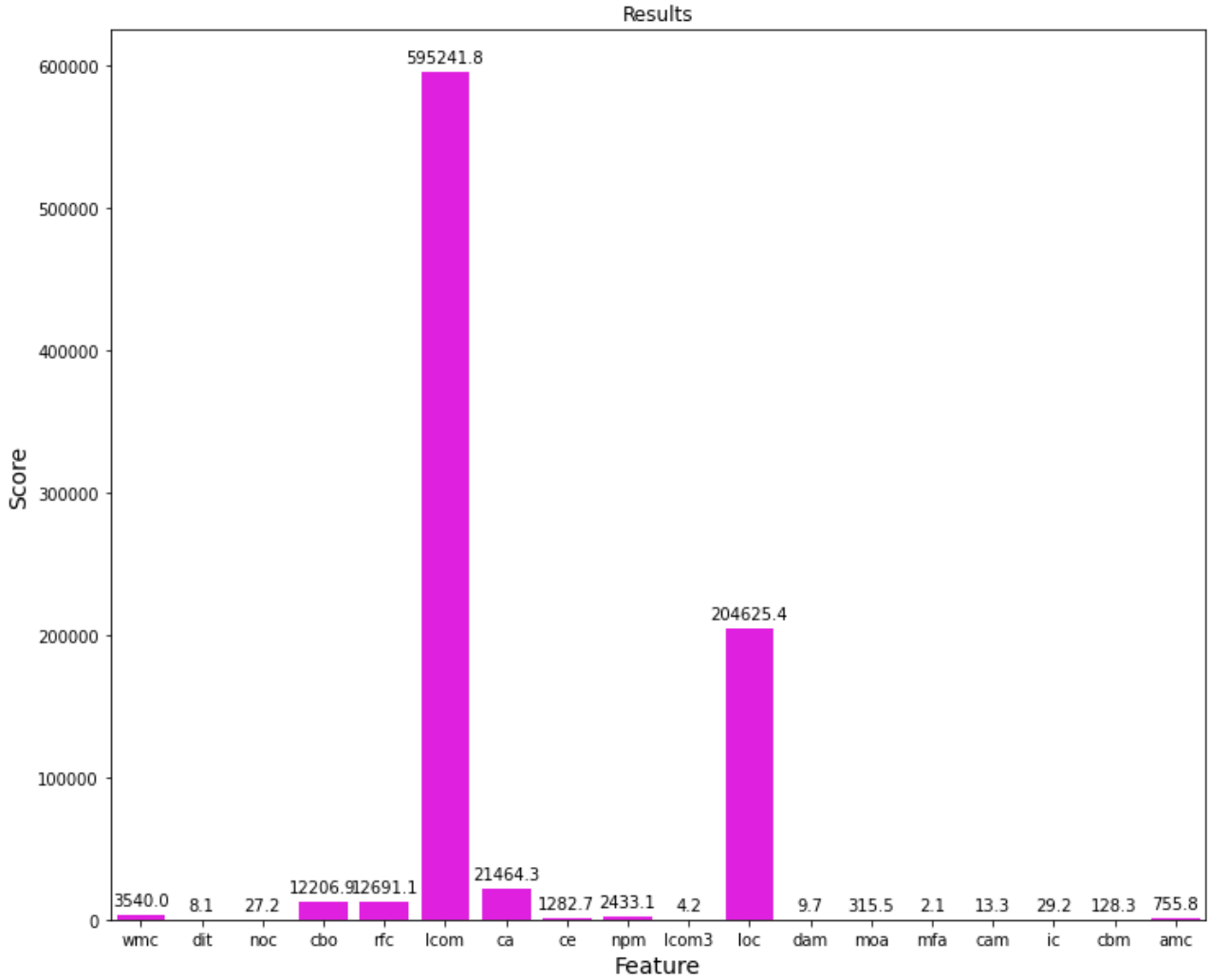
The best independent features from this approach are:

- rfc
- lcom
- wmc

These features were chosen due to their high bug correlation value.

K-best approach

This approach is based on the chi2 score. The function computes the chi2 statistic between each feature X and Y (bug). The selection of the k (3) features is based on the scores obtained.



The best independent features from this approach are:

- lcom
- ca
- loc

Selected features

After obtaining the two lists of features, a test was carried out which consists in creating and training a linear regression model, with a fixed random state, in order to evaluate the value of R2. This will only take into account the features that fit best.

Correlation matrix feature test:

Feature	R2
Rfc	0.39
Loc	0.34

wmc	0.29

K best feature test:

Feature	R2
lcom	0.17
ca	-0.30
loc	0.34

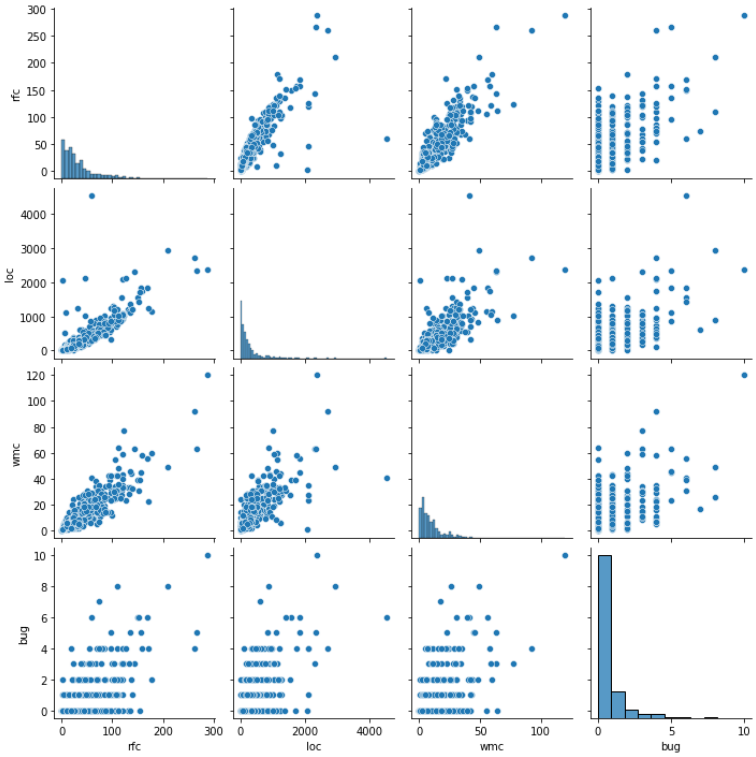
The set of features selected is one obtained through the correlation matrix approach as some R2 values of the features selected with the k best approach are extremely low or negative.

At this point, a dataframe is created containing only the three selected independent features and the bug dependent feature.

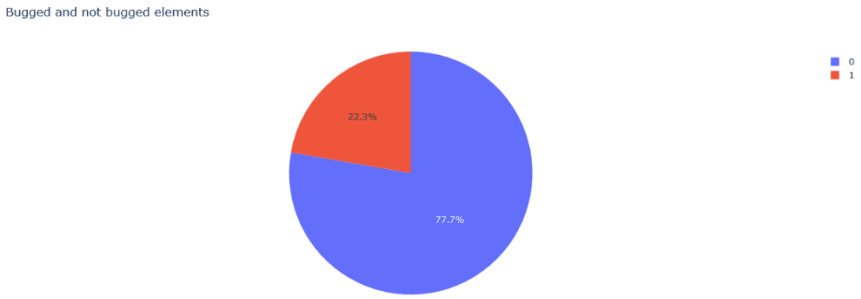
Data visualization

Pairwise relationship in the dataframe.

The diagonal plots are a univariate distribution plot, made to show the marginal distribution of the data in each column.

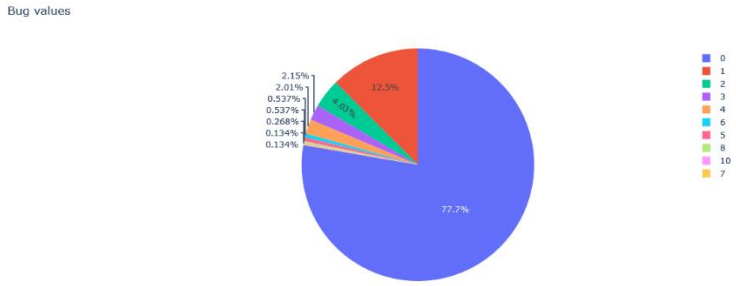


Piechart of bugged and non bugged elements



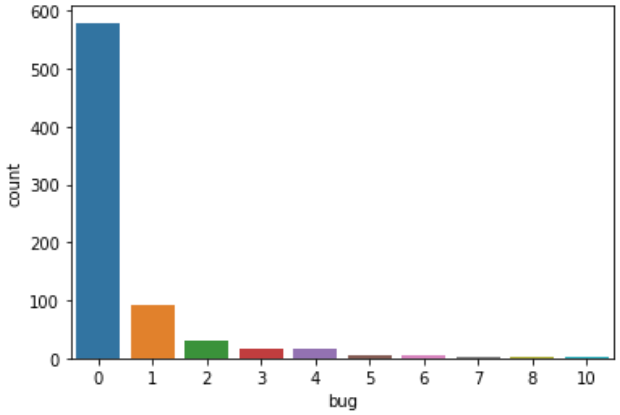
77.7% of the elements are not bugged and 22.3% are bugged.

Piechart of bug#



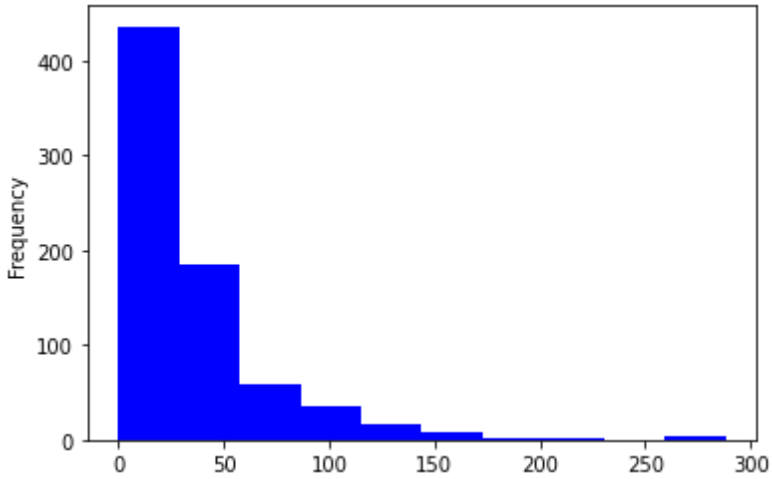
%	# of bugs
77.7	0
12.5	1
4.03	2
2.15	3
2.01	4
0.537	6
0.537	5
0.268	8
0.134	10
0.134	7

Countplot of bug feature



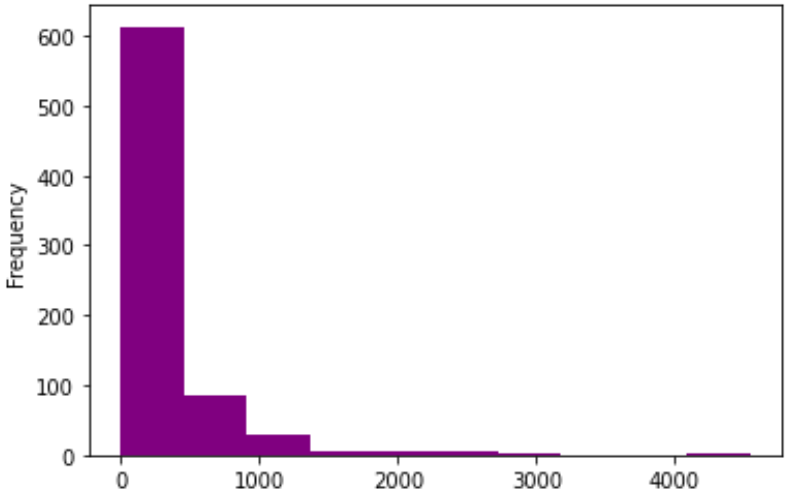
The majority value of the bug feature il 0.

Histplot of rfc feature



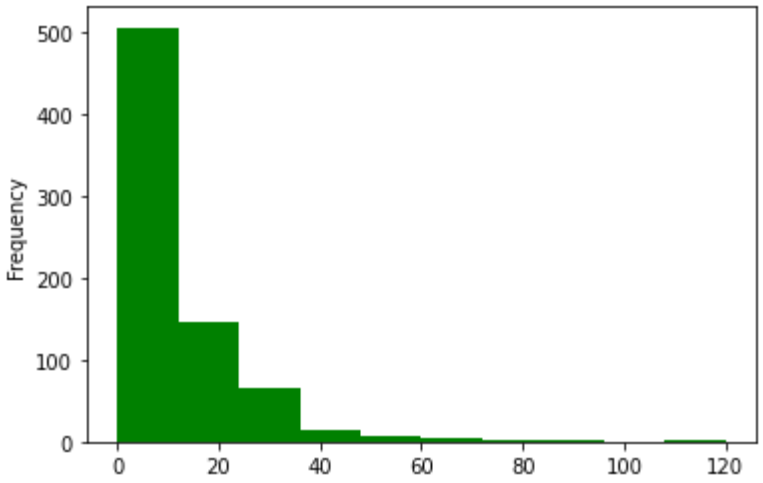
The most frequent rfc values are between 0 and 25

Histplot of loc feature



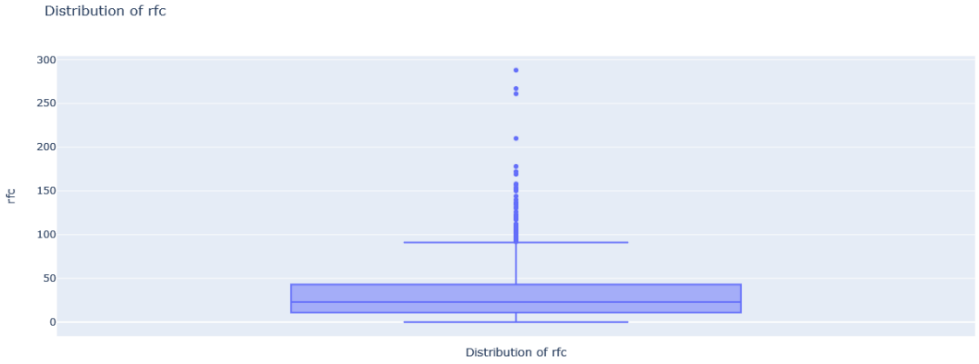
The most frequent loc values are between 0 and 50

Histplot of wmc feature



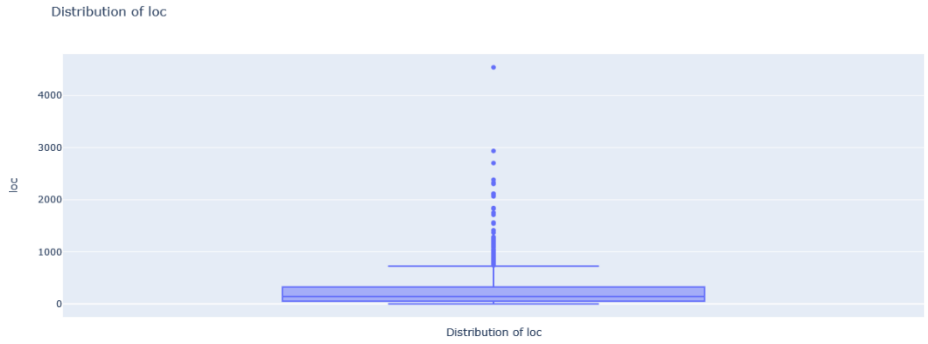
The most frequent wmc values are between 0 and 10

Distribution of rfc



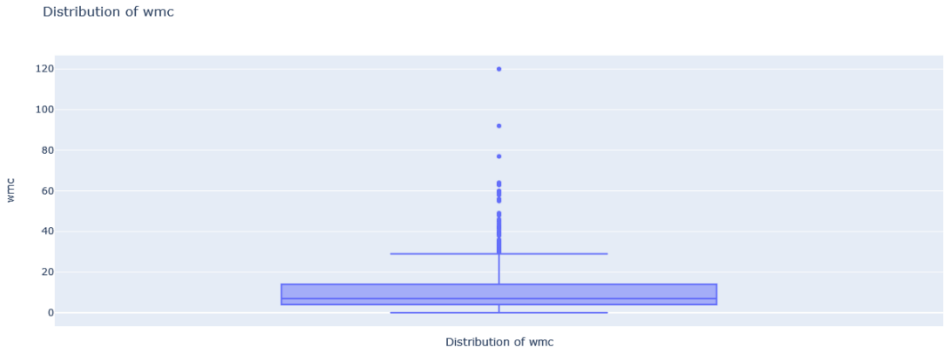
- min: lower fence: 0
- q1: 11
- median: 23
- q3: 43
- upper fence: 91
- max: 288

Distribution of loc



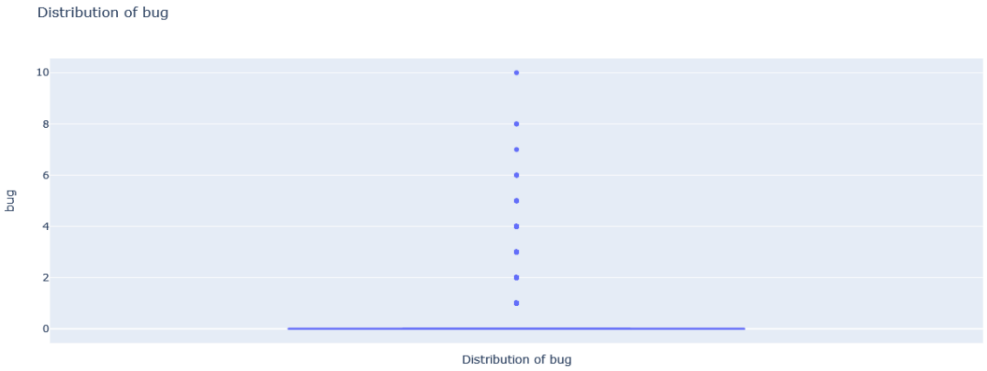
- min: lower fence: 0
- q1: 52
- median: 143
- q3: 324
- upper fence: 725
- max: 4541

Distribution of wmc



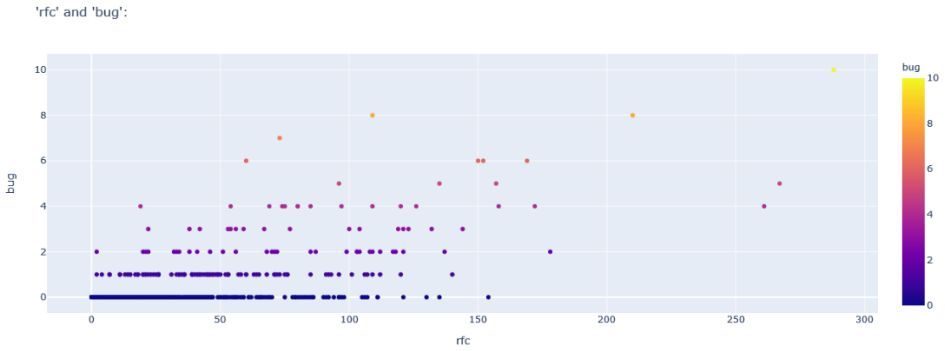
- min: lower fence: 0
- q1: 4
- median: 7
- q3: 14
- upper fence: 29
- max: 120

Distribution of bug



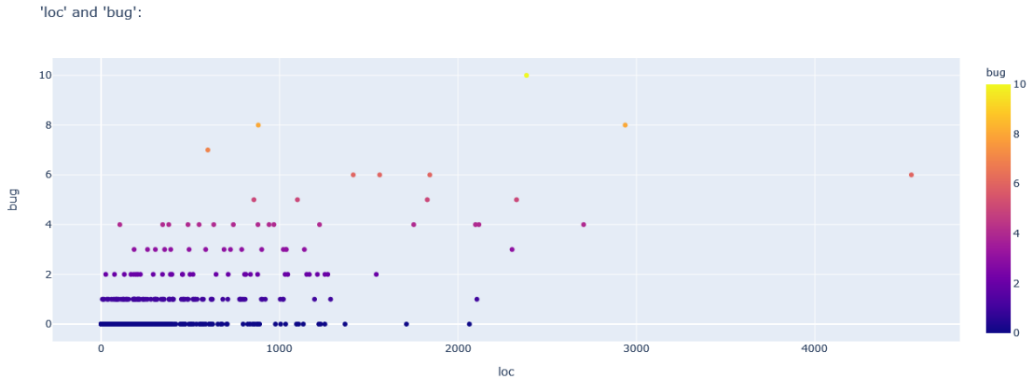
- min: median: 0
- max: 10

Scatterplot of rfc and bug



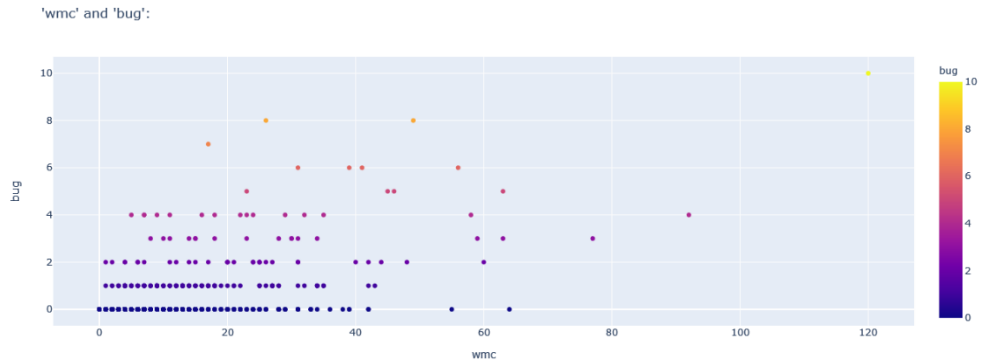
The highest concentration of values is near the x axis with a x value lower than 100

Scatterplot of loc and bug



The highest concentration of values is near the x axis with a x value lower than 400

Scatterplot of wmc and bug



The highest concentration of values is near the x axis with a x value lower than 40

Descriptive statistics

Two dataframes are created, one with only bugged elements and one with only non bugged elements from the dataframe with the three chosen independent feature and the dependent feature bug.

Elements without bugs	579
Elements with bugs	166

# of bugs	Total records
0	579
1	93
2	30
3	16
4	15
6	4
5	4
8	2
10	1
7	1

Full dataframe description

The minimum number of rfc : 0
The maximum number of rfc : 288
The average number of rfc : 34.36241610738255
The value of the standard deviation of the feature rfc : 36.02497169398523

The minimum number of loc : 0
The maximum number of loc : 4541
The average number of loc : 280.07114093959734
The value of the standard deviation of the feature loc : 411.87207539635864

The minimum number of wmc : 0
The maximum number of wmc : 120
The average number of wmc : 11.071140939597315
The value of the standard deviation of the feature wmc : 11.97596324330988

Dataframe with only bugged elements description

The minimum number of rfc : 2
The maximum number of rfc : 288
The average number of rfc : 68.6987951807229
The value of the standard deviation of the feature rfc : 49.779692814655164

The minimum number of loc : 7
The maximum number of loc : 4541
The average number of loc : 643.7168674698795
The value of the standard deviation of the feature loc : 637.9721267435144

The minimum number of wmc : 1
The maximum number of wmc : 120
The average number of wmc : 20.759036144578314
The value of the standard deviation of the feature wmc : 17.2430682507991

Dataframe with only non bugged elements description

The minimum number of rfc : 0
The maximum number of rfc : 154
The average number of rfc : 24.518134715025905
The value of the standard deviation of the feature rfc : 22.966296939877136

The minimum number of loc : 0
The maximum number of loc : 2064
The average number of loc : 175.8134715025907
The value of the standard deviation of the feature loc : 230.87637255245437

The minimum number of wmc : 0
The maximum number of wmc : 64
The average number of wmc : 8.293609671848014
The value of the standard deviation of the feature wmc : 8.065699766683172

Comparison

The biggest differences are in the maximum values of the features and in the average values of the features.
In the only bugged dataframe maximum and average value are higher than in the non bugged dataframe.

Data analysis

Linear regression

As a first step for the analysis by linear regression, 3 data frames were created containing an independent feature and the dependent feature.

Subsequently through the following function

```
#get 5 random states based on R2 value
def find_top5_random_states(df,feature):
  X = df[[feature]]
  y = df['bug']
  list1=[]
  bar = progressbar.ProgressBar()
  for cyc in bar(range(60000)):
    list2=[]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
3, random_state = cyc)
    lm = LinearRegression()
    lm.fit(X_train, y_train)
    prediction = lm.predict(X_test)
    list2.append(r2_score(y_test, prediction))
    list2.append(metrics.mean_squared_error(y_test, prediction))
    list2.append(cyc)
    list1.append(list2)
  list1.sort(reverse=True)
  print("\nTop 5 random states for ",feature," by R2 score:")
  print("[R2 value, MSE value, random state]\n")
  show_top5(list1)
```

it was possible to obtain the 5 best random states for the 3 linear regressions.

Rfc results:

R2 value	MSE value	Random state
0.65	0.54	5379
0.63	0.45	46705
0.62	0.44	54184
0.62	0.52	7810
0.62	0.39	3138

Loc results:

R2 value	MSE value	Random state
0.66	0.39	17280
0.65	0.44	49149
0.65	0.47	938

0.64	0.49	29064
0.64	0.54	25564

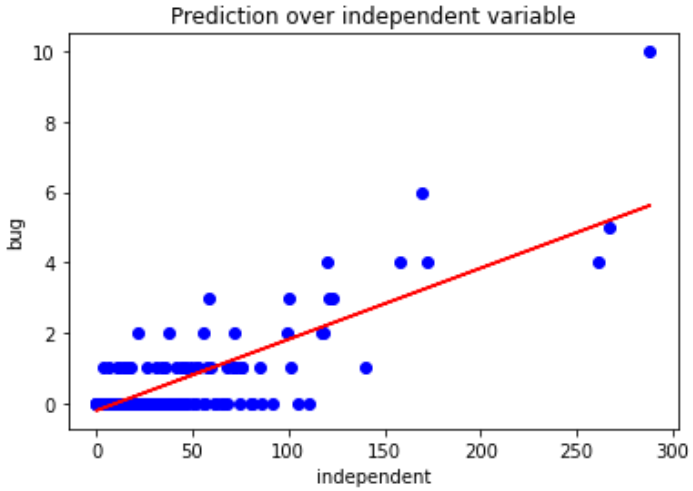
Wmc results:

R2 value	MSE value	Random state
0.57	0.50	54184
0.56	0.47	59405
0.56	0.45	3138
0.55	0.60	13051
0.55	0.55	46180

For the choice of the random state to use, the mean squared error value was also taken into consideration, which must be as small as possible and the R2 score as high as possible.

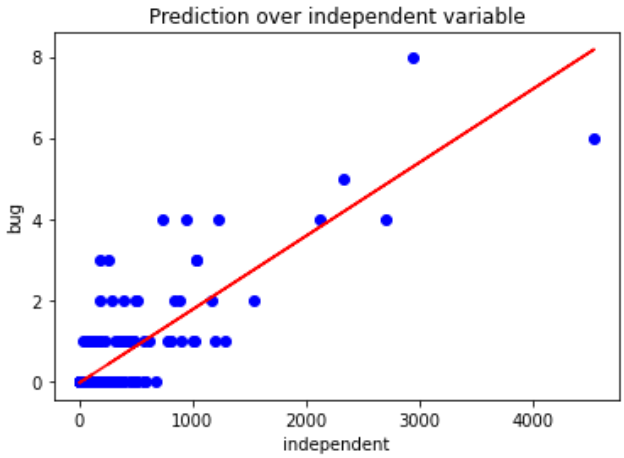
Rfc random state: 46705
Loc random state: 17280
Wmc random state: 54184

Prediction of bug with rfc



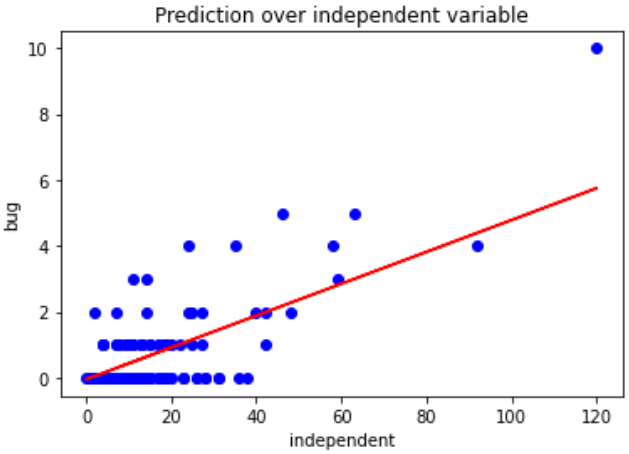
The prediction of the bug value with rfc as independent feature is accurate at 63%

Prediction of bug with loc



The prediction of the bug values with loc as independent feature is accurate at 66%

Prediction of bug with wmc



The prediction of the bug values with wmc as independent feature is accurate at 57%

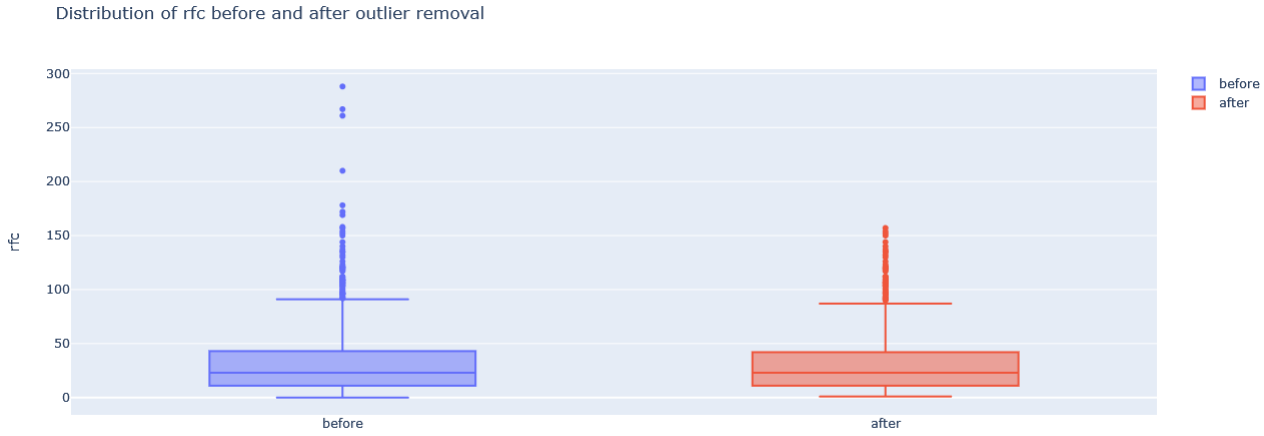
Outlier removal

Three new data frames were created using the following function from the three data frames containing an independent feature and the dependent feature

```
def outlier_remover(df,feature):
  q_low = df_selected[feature].quantile(0.01)
  q_hi  = df_selected[feature].quantile(0.99)
  df_filtered = df[(df[feature] < q_hi) & (df[feature] > q_low)]
  return df_filtered
```

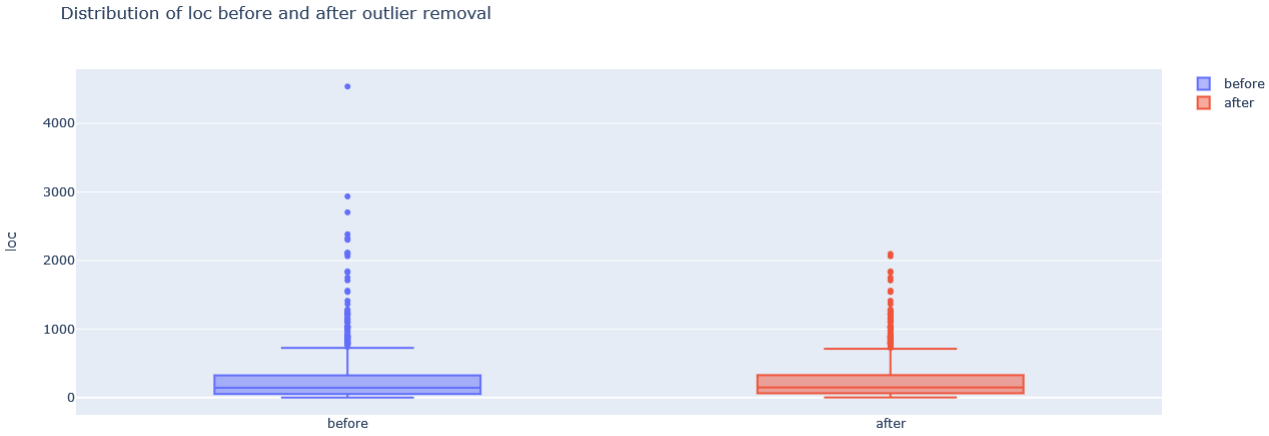
This way removing outlier values for one feature did not also cause the removal of non-outlier data for other features.

Distribution of rfc before and after outlier removal



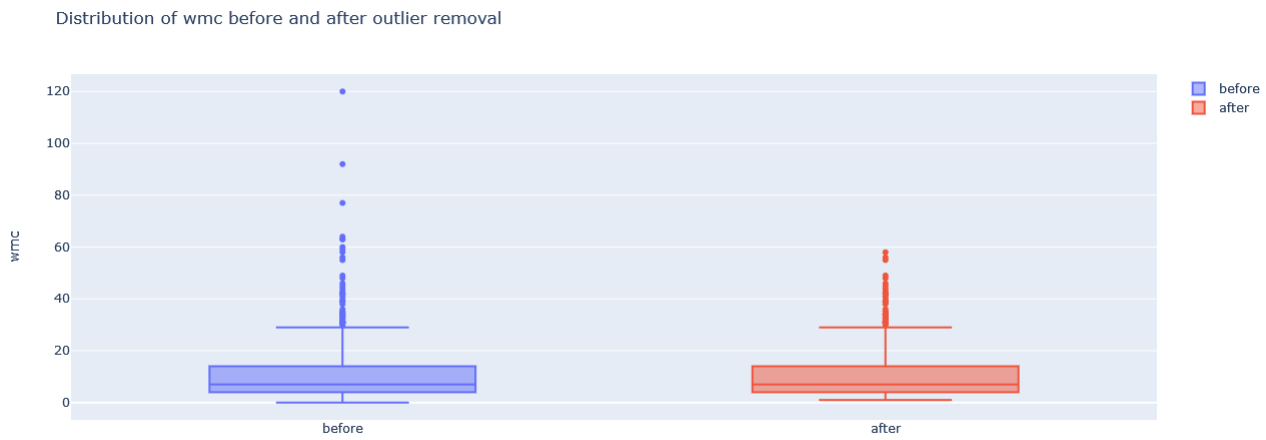
With the graph is possible to see that only the values over 157 where removed.

Distribution of loc before and after outlier removal



With the graph is possible to see that only the values over 2098 where removed.

Distribution of wmc before and after outlier removal



With the graph is possible to see that only the values over 58 where removed.

Prediction without outliers

After removing the outliers, the 5 best random states for each dataframe were obtained.

Rfc filtered results:

R2 value	MSE value	Random state
0.51	0.39	56096
0.5	0.39	12436
0.5	0.45	45212
0.49	0.47	59365
0.49	0.31	58551

Loc filtered results:

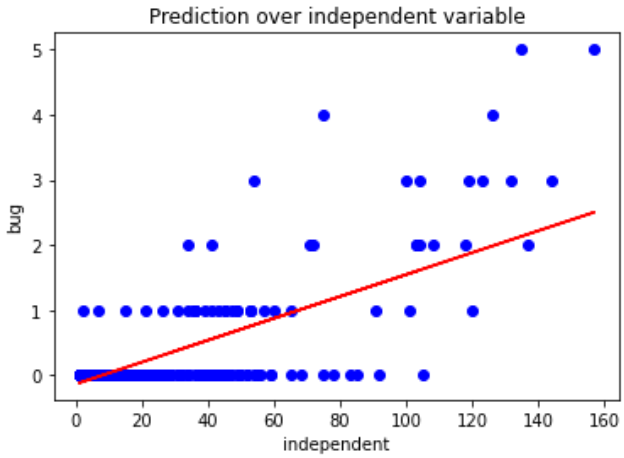
R2 value	MSE value	Random state
0.53	0.41	43624
0.53	0.24	18443
0.52	0.36	28471
0.52	0.38	3548
0.52	0.36	9814

Wmc filtered results:

R2 value	MSE value	Random state
0.42	0.49	55037
0.41	0.6	30601
0.41	0.48	25776
0.4	0.51	58197
0.4	0.65	12855

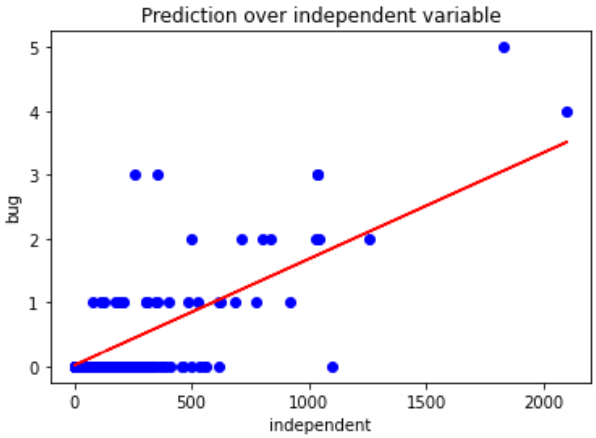
Rfc filtered random state: 56096
Loc filtered random state: 18443
Wmc filtered random state: 55037

Prediction of bug with rfc filtered



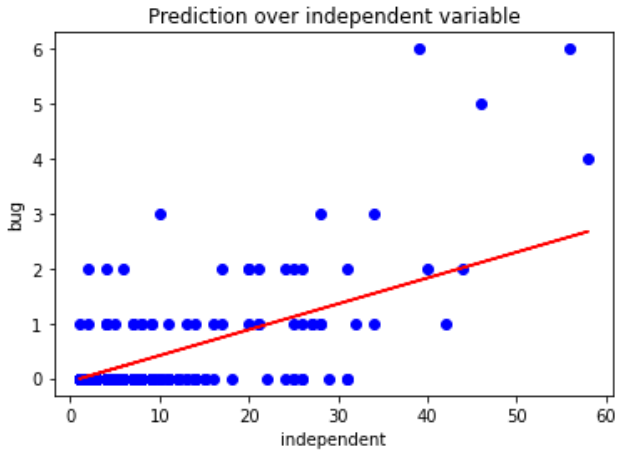
The prediction of the bug values with rfc as independent feature is accurate at 51% after remiving the outlier values. The R2 score is decreased after the outlier removal.

Prediction of bug with loc filtered



The prediction of the bug values with loc as independent feature is accurate at 53% after removing the outlier values. The R2 score is decreased after the outlier removal.

Prediction of bug with wmc filtered

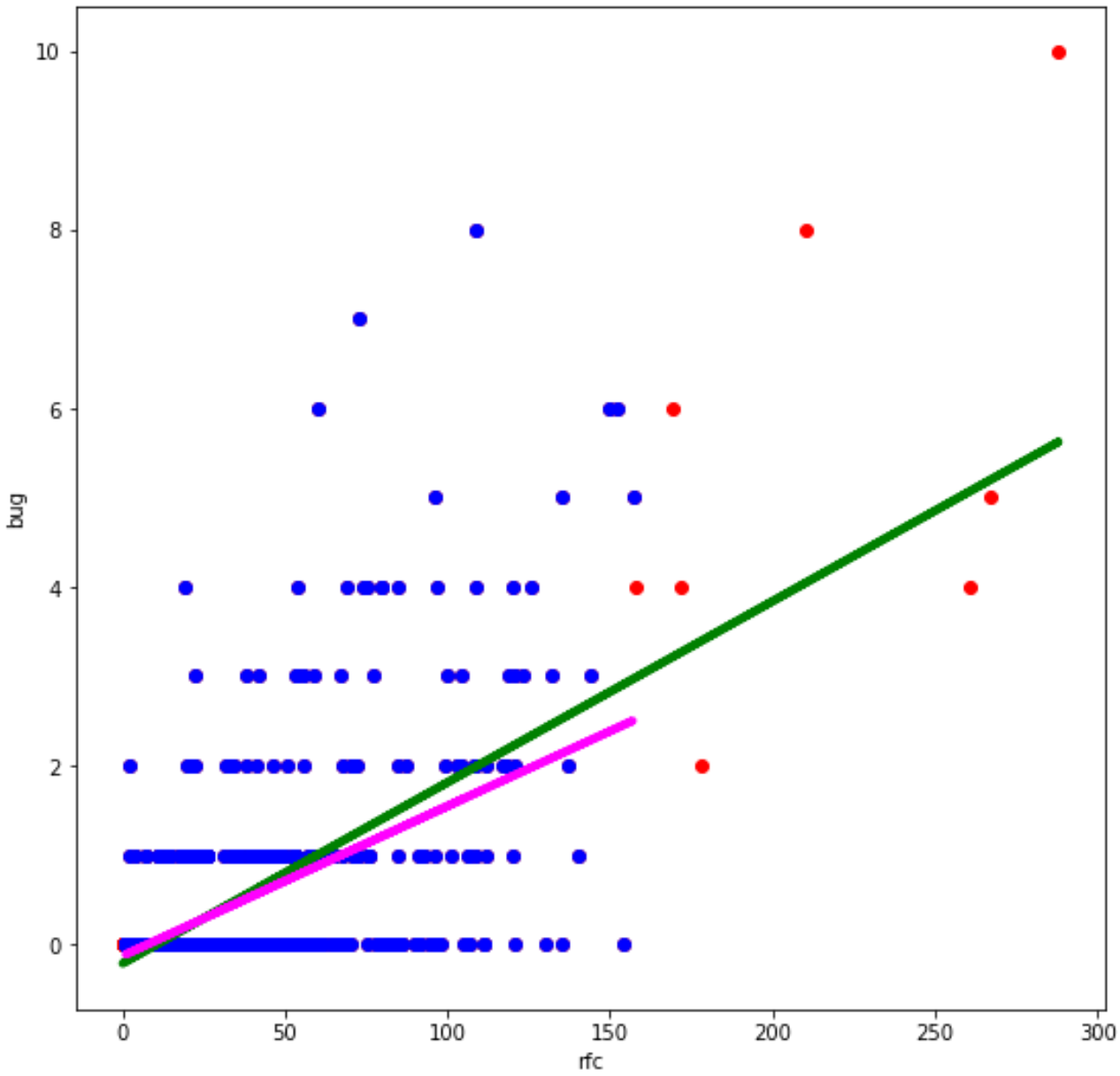


The prediction of the bug values with wmc as independent feature is accurate at 42% after removing the outlier values. The r2 error is decreased after the outlier removal.

Regression result comparison (before and after outlier removal)

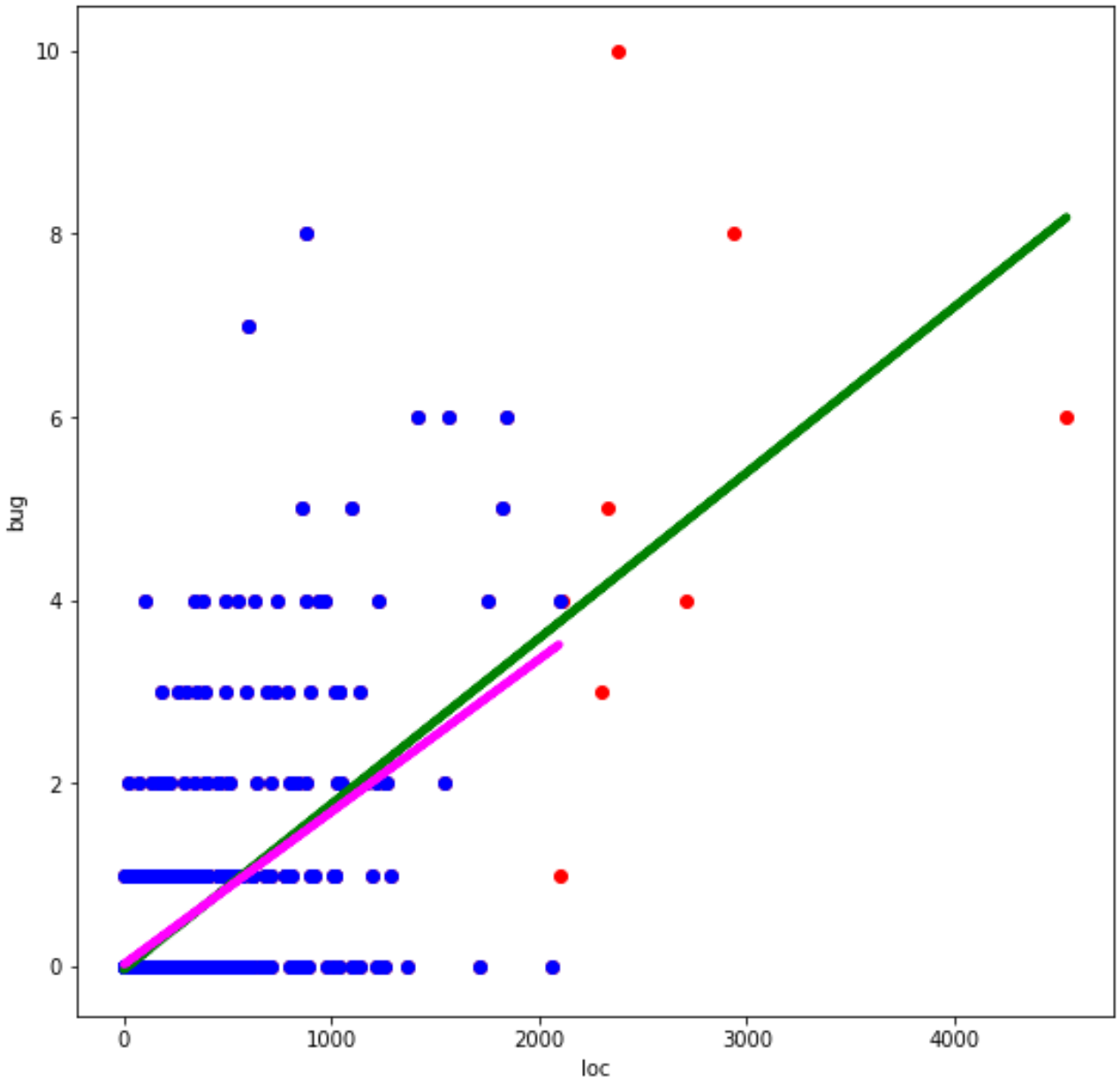
The following graphs were generated by overlapping the dataframe elements with and without outliers and the results of the predictions made by linear regression.
The blue points are the non-outlier elements, the red points are the outlier elements, the green line is the prediction made with the complete dataframe and the purple line is the prediction made with the dataframe without outliers.

Rfc before and after outlier removal comparison



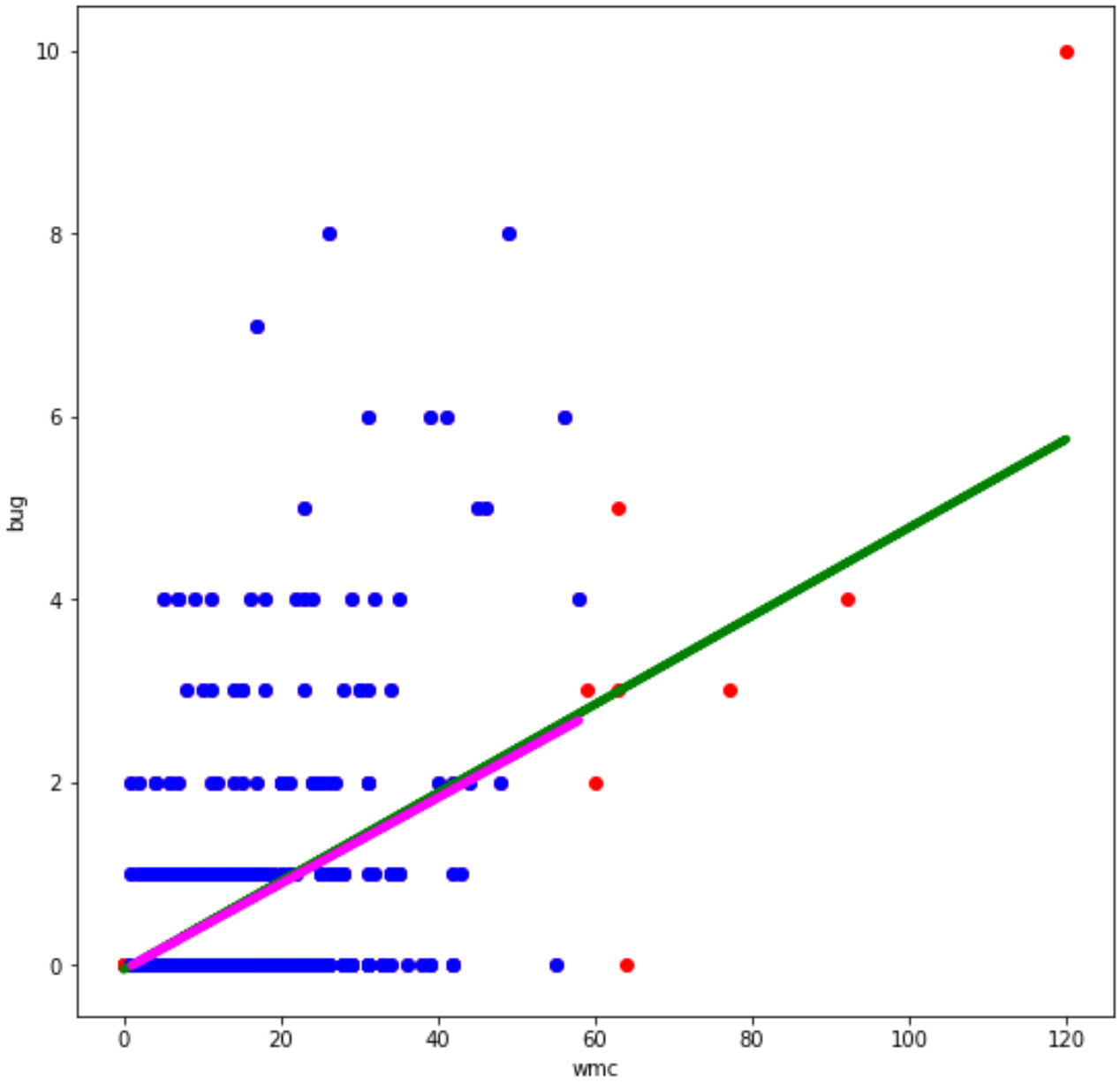
The purple line is quite different from the green line so there is a slight improvement in the predicted bug values after removing the outliers.

Loc before and after outlier removal comparison



In this case the purple line is slightly different from the green line so the removal of the outliers has brought a very small improvement in the predicted bug values.

Wmc before and after outlier removal comparison



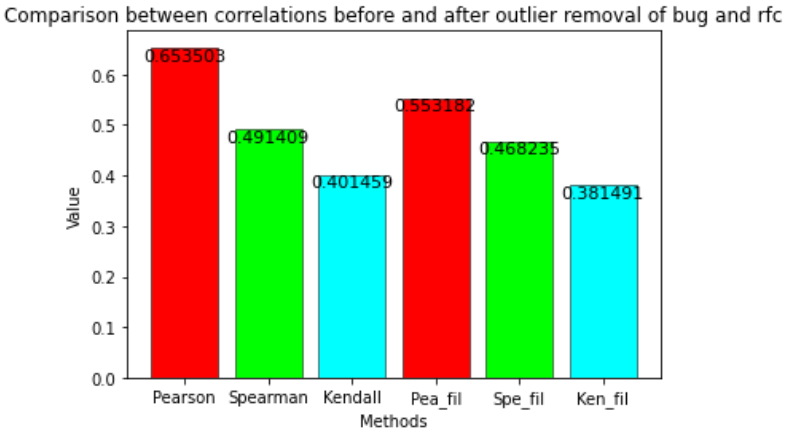
In this case the purple line differs very little from the green line making the predictions practically identical. Removing the outliers in this case did not lead to improvements in the predicted bug values.

Correlation analysis

The correlation values were calculated before and after the removal of the outliers in the three data frames created for the linear regression, with the methods:

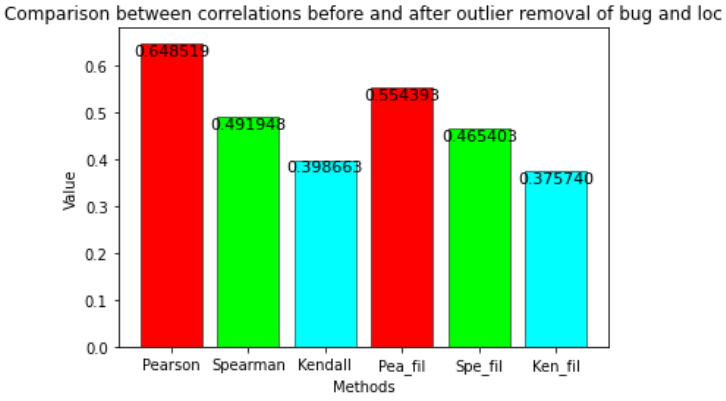
- Pearson's R
- Spearman's RHO
- Kendall's TAU

Correlation degree between rfc and bug before and after outlier removal



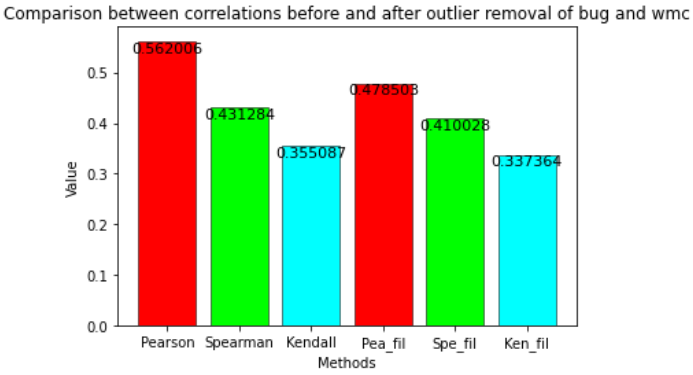
The degree of correlation after removal of the outliers has dropped. The correlation calculated with the Pearson method had the largest loss (about 0.1). On the other hand, for the degrees of correlation calculated with the Spearman and Kendal method, the loss was lower (about 0.02).

Correlation degree between loc and bug before and after outlier removal



The degree of correlation after removal of the outliers has dropped. The correlation calculated with the Pearson method had the largest loss (about 0.1). On the other hand, for the degrees of correlation calculated with the Spearman and Kendal method, the loss was lower (about 0.3).

Correlation degree between wmc and bug before and after outlier removal



The degree of correlation after removal of the outliers has dropped. The correlation calculated with the Pearson method had the largest loss (about 0.08). On the other hand, for the degrees of correlation calculated with the Spearman and Kendal method, the loss was lower (about 0.02 and about 0.01).

Comments about correlation comparisons

In general, the removal of the outlier values of the independent feature and the respective values of the dependent feature resulted in a fairly high loss of correlation in the case of the one calculated with the Pearson method and slight for that calculated with the other two methods. By modifying the parameters of the outlier removal function it is possible to further lower the value of the correlations, removing even more elements.

Logistic regression

Unlike Linear Regression, Logistic Regression does not need separate dataframes as it is a classifier. The dataframe used contains the values of the three independent variables, the bug values, and a bug-derived value, is_bugged, which indicates whether or not a given record contains bugs, regardless of the number of bugs it contains. The bug value was not used for model training as it leads to maximum accuracy.

Also in this case the following function was used to find the 5 best random states

```
#get 5 random states based on accuracy value
def find_top5_random_states_LR(df):
  list1=[]
  bar = progressbar.ProgressBar()
  for cyc in bar(range(60000)):
    list2=[]
    X_train, X_test, y_train, y_test=feature_selection_lr(df,cyc)
    lrm=lrm_train(X_train,y_train)
    predictions = lrm_test(lrm,X_test)
    accuracy = accuracy_score(y_test, predictions)*100
    fixed_accuracy=round(accuracy, 5)
    list2.append(fixed_accuracy)
    list2.append(cyc)
    list1.append(list2)
  list1.sort(reverse=True)
  print("\nTop 5 random states for Logistic Regression by accuracy score:")
  print("[Accuracy, random state]\n")
  show_top5(list1)
```

Results:

Accuracy	Random state
90.62	3856
90.17	42396
90.17	16042
90.17	15526
89.73	59123

The accuracy value was taken into consideration for the choice of the random state.
Chosen random state: 3856

```
[[184   6]
 [ 15  19]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       190
           1       0.76      0.56      0.64        34

    accuracy                           0.91       224
   macro avg       0.84      0.76      0.80       224
weighted avg       0.90      0.91      0.90       224
```

As can be seen from the confusion matrix, the values of class 0 are almost 10 times those of class 1. Compared to the elements of class 1, the misclassifications of elements of class 0 classified as elements of class 1 are extremely few, unlike the elements of class 1. classified as class 0 which are almost half of the actual class 1 elements used for model evaluation. This model manages to almost perfectly classify items without bugs while having difficulty classifying items with bugs. This is due to the fact that 77.7% of the total elements of the entire dataset have no bugs so the model is more likely to classify non-bugged elements well and therefore to misclassify bugged elements or non-bugged elements that have similar characteristics to elements that have bugs.

Result description

Using linear regression models with the three independent features selected and the bug dependent feature works well enough for predicting bug values most of the time. Rfc is the feature that performs better to predict bug values with linear regression.

In order to perform better, the logistic regression model would need more varied train data as almost 80% of the total data has no bugs, so the trained model better recognizes non-bugged elements and misclassifies the bugged elements more.