A NONLINEAR REGRESSION MODEL FOR EARLY LOC ESTIMATION OF OPEN-SOURCE KOTLIN-BASED APPLICATIONS

Prykhodko S. B. – Dr. Sc., Professor, Head of the Department of Software for Automated Systems, Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine.
Prykhodko N. V. – PhD, Associate Professor, Associate Professor of the Finance Department, Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine.
Koltsov A. V. – Post-graduate student of the Department of Software for Automated Systems, Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine.

ABSTRACT

Context. The early lines of code (LOC) estimation in software projects holds significant importance, as it directly influences the prediction of development effort, covering a spectrum of different programming languages, and open-source Kotlin-based applications in particular. The object of the study is the process of early LOC estimation of open-source Kotlin-based apps. The subject of the study is the nonlinear regression models for early LOC estimation of open-source Kotlin-based apps.

Objective. The goal of the work is to build the nonlinear regression model with three predictors for early LOC estimation of open-source Kotlin-based apps based on the Box-Cox four-variate normalizing transformation to increase the confidence in early LOC estimation of these apps.

Method. For early LOC estimation in open-source Kotlin-based apps, the model, confidence, and prediction intervals of nonlinear regression were constructed using the Box-Cox four-variate normalizing transformation and specialized techniques. These techniques, relying on multiple nonlinear regression analyses incorporating multivariate normalizing transformations, account for the dependencies between variables in non-Gaussian data scenarios. As a result, this method tends to reduce the mean magnitude of relative error (MMRE) and narrow confidence and prediction intervals compared to models utilizing univariate normalizing transformations.

Results. An analysis has been carried out to compare the constructed model with nonlinear regression models employing decimal logarithm and Box-Cox univariate transformation.

Conclusions. The nonlinear regression model with three predictors for early LOC estimation of open-source Kotlin-based apps is constructed using the Box-Cox four-variate transformation. Compared to the other nonlinear regression models, this model demonstrates a larger multiple coefficient of determination, a smaller value of the MMRE, and narrower confidence and prediction intervals compared to models utilizing univariate normalizing transformations. The prospects for further research may include the application of other datasets to construct the nonlinear regression model for early LOC estimation of open-source Kotlin-based apps for other restrictions on predictors.

KEYWORDS: estimation, lines of code, open-source app, Kotlin, nonlinear regression model, Box-Cox transformation, class, weighted methods per class, depth of inheritance tree.

ABBREVIATIONS

DIT is a depth of inheritance tree;
KLOC is a thousand lines of code;
LB is a lower bound;
LCOM is a lack of cohesion of methods;
LOC are lines of code;
MMRE is a mean magnitude of relative error;
MRE is a magnitude of relative error;
PRED is a percentage of prediction;
RFC is a response for class;
SMD is a squared Mahalanobis distance;
UB is an upper bound;
WMC are weighted methods per class.

NOMENCLATURE

\( \hat{b} \) is an estimator for a vector of linear regression equation parameters;
\( \hat{b}_i \) is an estimator for the \( i \)-th parameter of linear regression equation;
\( k \) is a number of predictors (independent variables);
\( N \) is a number of data points;
\( P \) is a non-Gaussian random vector;
\( R^2 \) is a multiple coefficient of determination;
\( S_Z \) is a sample covariance matrix for normalized data;
\( S_{MDZ} \) is a squared Mahalanobis distance for normalized data;
\( T \) is a Gaussian random vector;
\( t_{\alpha/2,v} \) is a quantile of the student’s \( t \)-distribution with \( v \) degrees of freedom and \( \alpha/2 \) significance level;
\( X_1 \) is a number of classes;
\( X_2 \) is a WMC metric at the app level (a WMC mean value per class);
\( X_3 \) is a DIT metric at the app level (a DIT mean value per class);
\( Y \) is an actual software size in KLOC;
\( Z_j \) is a \( j \)-th Gaussian variable that is obtained by transforming the variable \( X_j \);
\( Z_Y \) is a Gaussian variable that is obtained by transforming variable \( Y \);
\( \bar{Z}_Y \) is a sample mean of the \( Z_Y \) values;
\( \hat{Z}_Y \) is a prediction result by linear regression equation for normalized data;
\( \alpha \) is a significance level;
\[ \beta_1 \text{ is a multivariate skewness}; \]
\[ \beta_2 \text{ is a multivariate kurtosis}; \]
\[ \varepsilon \text{ is a Gaussian random variable that defines residuals}; \]
\[ \nu \text{ is a number of degrees of freedom}; \]
\[ \sigma_\varepsilon \text{ is a standard deviation of } \varepsilon; \]
\[ \psi \text{ is a vector of multivariate normalizing transformation.} \]

**INTRODUCTION**

As we know [1], Lines of Code (LOC) is the number of lines of code excluding comments. Early software size estimation, including LOC, is one of the project managers’ significant problems in evaluating software development efforts using mathematical models like COCOMO II [2].

The multi-platform nature of Kotlin language simplifies the development of cross-platform apps, primarily mobile ones. That is why, Kotlin Multiplatform Mobile (KMM) already has a handful of successful apps on the market [3].

Despite a large number of currently existing methods and models for estimating the software size [4–9], research in this direction does not stop [10–15]. This is primarily due to the low accuracy of estimating the size of the software in the early stages of its development. One way to solve this problem is to develop appropriate models for estimating the size of the software developed in a specific programming language. Today some LOC estimation models based on the software metrics that can be measured from the class diagram are known [4, 6, 8, 10–12]. The above models are constructed for such languages as Java [4, 6, 10, 11], C++ [8], PHP [4, 6, 12], and Visual Basic [4, 6]. However, there are no models, both linear and nonlinear ones, for early LOC estimation of open-source Kotlin-based apps. This demands the construction of the models for early LOC estimation of open-source Kotlin-based apps.

**The object of study** is the process of early LOC estimation of open-source Kotlin-based apps.

**The subject of study** is the regression models for early LOC estimation of open-source Kotlin-based apps.

**The purpose of the work** is to increase confidence in early LOC estimation of open-source Kotlin-based apps.

**1 PROBLEM STATEMENT**

Suppose given the original sample as the four-dimensional non-Gaussian data set: actual software size in the thousand lines of code (KLOC) \( Y \), the total number of classes \( X_1 \), the WMC metric at the app level \( X_2 \), the DIT metric at the app level \( X_3 \) from \( N \) open-source Kotlin-based apps. Suppose that there are four-variate normalizing transformation of non-Gaussian random vector 
\[ \mathbf{P} = (Y, X_1, X_2, X_3)^T \] to Gaussian random vector 
\[ \mathbf{T} = (Z_Y, Z_1, Z_2, Z_3)^T \] is given by

\[ T = \psi(P) \] (1)

and the inverse transformation for (1)

\[ P = \psi^{-1}(T). \] (2)

It is required to build the nonlinear regression model in the form \( Y = Y(X_1, X_2, X_3, \varepsilon) \) based on the transformations (1) and (2).

**2 REVIEW OF THE LITERATURE**

In paper [6] the linear regression equations were proposed for LOC estimation of software of open-source PHP- and Java-based information systems. These equations are developed based on three metrics that can be gained from a conceptual data model derived from a class diagram: the total number of classes, the total number of relationships, and the average number of attributes per class. However, the application of linear regression models is grounded on four primary assumptions, one of which relates to the normality of the error distribution. Nevertheless, this assumption is applicable only in specific scenarios. Therefore, in paper [10], the nonlinear regression model was constructed using the same above metrics for LOC estimation of software of Java-based information systems. However, the size of software apps may depend on other metrics. That is why in [11] the nonlinear regression model was constructed for early LOC estimation of Java-based apps. This model depends on four factors (predictors), namely the total number of classes, the number of static methods, the LCOM metric, and the RFC metric. However, the size of open-source Kotlin-based apps may depend on other metrics too. This leads to the need to build the nonlinear regression model for early LOC estimation of open-source Kotlin-based apps.

Although machine learning methods are becoming increasingly popular for the estimation of various software metrics [13, 15–22], including software size [13, 15], methods and models based on regression analysis have not yet reached their full potential [12, 25–27]. We suggest using the nonlinear regression models for early LOC estimation of open-source Kotlin-based apps because, firstly, there are two random variables, both a dependent variable (response) and an error term (residuals), in a regression model, and, secondly, the error distribution is not Gaussian.

One should note, that employing a normalizing transformation is frequently an effective approach to construct nonlinear regression models for early LOC estimation of various software apps [10–12]. As commonly understood, transformations serve essentially four purposes, with two main aims: firstly, to attain an approximate normal distribution for the error term in linear regression with normalized data, and secondly, to modify the response and/or predictor variables to enhance the linear relationship strength between new variables (normalized variables) compared to the original relationship between dependent and independent variables.
Commonly utilized methods for constructing nonlinear regression models typically rely on univariate normalizing transformations, such as the decimal logarithm and the Box-Cox transformation. However, these techniques fail to consider the correlation between dependent and independent variables. As a result, using such univariate normalizing transformations in the construction of nonlinear regression models does not consistently ensure optimal normality and linear relationships between normalized variables [12]. This emphasizes the necessity of employing multivariate normalizing transformations. Thus, following the methodology outlined in [12], we employ the technique for constructing nonlinear regression models based on multivariate normalizing transformations and prediction intervals to develop a model with three predictors for early estimation of lines of code (LOC) in open-source Kotlin-based applications. In this approach, prediction intervals from nonlinear regression models are applied to identify outliers during model construction. We detect the outliers due to residuals according to [29]. Typically, this procedure is iterative as we rebuild the model for new data after outlier removal. If there are no outliers, the process of constructing the model ends.

**3 MATERIALS AND METHODS**

The technique to build nonlinear regression models based on multivariate normalizing transformations and prediction intervals is comprised of six steps. The first step involves normalizing multivariate non-Gaussian data through a dedicated transformation (1). To do this, as in [12], we use the four-variate Box-Cox transformation with components

\[ Z_j = x_j^{\lambda_j} = \left\{ \begin{array}{ll}
\frac{X_j^\lambda - 1}{\lambda_j}, & \text{if } \lambda_j \neq 0; \\
\ln(X_j), & \text{if } \lambda_j = 0.
\end{array} \right. \]  

(3)

Here \( Z_j \) is a Gaussian variable; \( \lambda_j \) is a parameter of the Box-Cox transformation, \( j = 1, 2, 3 \). The variable \( Z_j \) is defined analogously (3) with the only difference that instead of \( Z_j \), \( X_j \), and \( \lambda_j \) should be put respectively \( Z_j \), \( Y \), and \( \lambda_Y \).

In the second step, we determine whether one multidimensional data point of a multivariate non-Gaussian data set is a multidimensional outlier. If there is a multidimensional outlier in a multivariate non-Gaussian data set then we discard the one and go to step 1, else continue.

To determine whether one data point of a multivariate non-Gaussian data set is a multidimensional outlier, we apply the statistical technique based on the normalizing transformations and the squared Mahalanobis distance (SMD) as in [12].

In the third step, we build the linear regression model for normalized data in the form

\[ Z_Y = \hat{Z}_Y + \varepsilon = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \hat{b}_3 Z_3 + \varepsilon, \]  

(4)

\( \varepsilon \) is a Gaussian random variable that defines residuals, \( \varepsilon \sim N(0, \sigma^2) \).

In the fourth step, we test the normality of the distribution of residuals in the linear regression model for normalized data. If the distribution of the residuals in the linear regression model for the normalized data is not Gaussian, then we discard the multivariate data point for which the modulus of the residual in the model is the maximum and go to step 1 otherwise continue.

The nonlinear regression model using the transformation (1) and (2) for the linear regression model for normalized data as in [12] is constructed in the fifth step

\[ Y = \psi^1 \left( \hat{Z}_Y + \varepsilon \right), \]  

(5)

For the four-variate Box-Cox transformation with components (3), the model has the form [12]

\[ Y = \left[ \hat{Z}_Y (\hat{Z}_Y + 1)^{1/\lambda_Y} \right], \]  

(6)

where \( \varepsilon \) is a Gaussian random variable, \( \varepsilon \sim N(0, \sigma^2) \), with the estimate \( \hat{\sigma}_\varepsilon \); \( \hat{Z}_Y \) is a prediction result by the linear regression equation

\[ \hat{Z}_Y = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \hat{b}_3 Z_3 \]  

for normalized data, which are transformed by the four-variate Box-Cox transformation with components (3).

Finally, in the sixth step, we build the prediction interval of nonlinear regression and determine whether one or more values of the response (dependent random variable) are outliers (its values are outside the prediction interval). If there are outliers in the data for the nonlinear regression model then we discard these and go to step 1, otherwise we complete constructing the nonlinear regression model.

We define the prediction interval of nonlinear regression as in [12]

\[ \psi^1_Y \left( \hat{Z}_Y + t_{\alpha/2, v} S_{Z_Y} \left[ \frac{1}{N} + \left( X^T \hat{\Sigma}_{X}^{-1} \right)^T S_{Z} \left( X^T \hat{\Sigma}_{X}^{-1} \right)^{1/2} \right] \right), \]  

(7)

where \( t_{\alpha/2, v} \) is a student’s \( t \)-distribution quantile with \( \alpha/2 \) significance level and \( v \) degrees of freedom; \( v = N - k - 1 \); \( k \) is the number of independent variables (in our case, \( k = 3 \)); \( X^T \) is a vector with components \( Z_{1i} - \bar{Z}_1, \; Z_{2i} - \bar{Z}_2, \; \ldots, \; Z_{ki} - \bar{Z}_k \) for \( i \)-row;

\[ \bar{Z}_j = \frac{1}{N} \sum_{i=1}^{N} Z_{ji}, \; j = 1, 2, \ldots, k; \quad S_{Z_Y} = \frac{1}{v} \sum_{i=1}^{v} (Z_{yi} - \hat{Z}_{yi})^2; \]  

\( v = N - k - 1; \; S_{Z} \) is a \( k \times k \) matrix.
According to this metric at the application level 2, the skewness 1, posed by Mardia and based on measures of the multivariate distribution. We applied a multivariate normality test based on the squared Mahalanobis distance (SMD), statistical methods, including multivariate outlier detection 1. This preliminary check was essential, as common methods 2. We checked the four-dimensional data from Table 1 for multivariate outliers, for multivariate outliers. Before analyzing the four-dimensional data from Table 1, we assessed the normality of the multivariate data in Table 1, and the DIT metric at the same level X_3. Table 1 contains that data set. We chose the above predictors X_1, X_2, and X_3 for two reasons. Firstly, these predictors can be obtained from the class diagram, and, secondly, there is no multicollinearity between these predictors since variance inflation factors for predictors X_1, X_2, and X_3 are equal to 1.06, 1.46, and 1.40, respectively.

We constructed a nonlinear regression model for early LOC estimation of open-source Kotlin-based apps by the above technique from 54 apps hosted on GitHub (https://github.com). We acquired the dataset utilizing the above technique from 54 apps hosted on GitHub. We constructed a nonlinear regression model for early LOC estimation of open-source Kotlin-based apps by the above technique from 54 apps hosted on GitHub. We acquired the dataset utilizing the above technique from 54 apps hosted on GitHub. We constructed a nonlinear regression model for early LOC estimation of open-source Kotlin-based apps by the above technique from 54 apps hosted on GitHub.

The parameter estimates of the four-variate Box-Cox transformation with components (3) was applied. The parameter estimates of the four-variate Box-Cox transformation for the data from Table 1 are calculated by the maximum likelihood method and are

$$\lambda_Y = -0.137228, \ \hat{\lambda}_1 = -0.138740, \ \hat{\lambda}_2 = -0.220743, \ \hat{\lambda}_3 = -1.067093.$$ 

There are two multivariate outliers in the four-dimensional non-Gaussian data in Table 1 since the SMD values for rows 35 and 47 exceed 14.86, which is the quantile of the Chi-Square distribution, applicable for a significance level of 0.005. In Table 1, rows that should be considered outliers are highlighted in bold. There are two iterations in step 1, next, we go to step 1 of the third iteration.

In step 1 of the third iteration, we discard the outliers (rows 35 and 47) and normalize 52 rows of data from Table 1 (without rows 35 and 47). In this case, the parameter estimates of the four-variate Box-Cox transformation for the data from Table 1 (without rows 35 and 47) are calculated by the maximum likelihood method and are

$$\hat{\lambda}_Y = -0.136019, \ \hat{\lambda}_1 = -0.156183, \ \hat{\lambda}_2 = -0.210099, \ \hat{\lambda}_3 = -1.294445.$$ 

Table 1 – The data set

<table>
<thead>
<tr>
<th>No</th>
<th>Y</th>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>X_4</th>
<th>X_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.683</td>
<td>72</td>
<td>5.37</td>
<td>0.939</td>
<td>28</td>
<td>4.111</td>
</tr>
<tr>
<td>2</td>
<td>1.256</td>
<td>55</td>
<td>3.56</td>
<td>1.473</td>
<td>20</td>
<td>12.248</td>
</tr>
<tr>
<td>3</td>
<td>14.867</td>
<td>546</td>
<td>5.40</td>
<td>0.987</td>
<td>30</td>
<td>0.829</td>
</tr>
<tr>
<td>4</td>
<td>23.149</td>
<td>1033</td>
<td>4.60</td>
<td>3.206</td>
<td>31</td>
<td>22.069</td>
</tr>
<tr>
<td>5</td>
<td>28.699</td>
<td>1090</td>
<td>5.33</td>
<td>1.156</td>
<td>32</td>
<td>2.179</td>
</tr>
<tr>
<td>6</td>
<td>8.303</td>
<td>266</td>
<td>7.25</td>
<td>1.711</td>
<td>33</td>
<td>11.800</td>
</tr>
<tr>
<td>7</td>
<td>5.008</td>
<td>122</td>
<td>9.66</td>
<td>1.648</td>
<td>34</td>
<td>1.425</td>
</tr>
<tr>
<td>8</td>
<td>22.078</td>
<td>1292</td>
<td>4.71</td>
<td>0.971</td>
<td>35</td>
<td>2.230</td>
</tr>
<tr>
<td>9</td>
<td>2.010</td>
<td>41</td>
<td>7.68</td>
<td>0.851</td>
<td>36</td>
<td>1.872</td>
</tr>
<tr>
<td>10</td>
<td>11.629</td>
<td>425</td>
<td>4.95</td>
<td>1.416</td>
<td>37</td>
<td>2.079</td>
</tr>
<tr>
<td>11</td>
<td>1.728</td>
<td>69</td>
<td>6.22</td>
<td>1.116</td>
<td>38</td>
<td>5.639</td>
</tr>
<tr>
<td>12</td>
<td>1.538</td>
<td>102</td>
<td>2.17</td>
<td>0.873</td>
<td>39</td>
<td>1.933</td>
</tr>
<tr>
<td>13</td>
<td>38.101</td>
<td>1151</td>
<td>6.85</td>
<td>0.929</td>
<td>40</td>
<td>17.575</td>
</tr>
<tr>
<td>14</td>
<td>7.655</td>
<td>290</td>
<td>5.90</td>
<td>1.062</td>
<td>41</td>
<td>5.437</td>
</tr>
<tr>
<td>15</td>
<td>4.975</td>
<td>354</td>
<td>3.91</td>
<td>0.832</td>
<td>42</td>
<td>2.357</td>
</tr>
<tr>
<td>16</td>
<td>24.324</td>
<td>701</td>
<td>4.48</td>
<td>1.126</td>
<td>43</td>
<td>2.079</td>
</tr>
<tr>
<td>17</td>
<td>1.031</td>
<td>19</td>
<td>26.35</td>
<td>1.034</td>
<td>44</td>
<td>1.201</td>
</tr>
<tr>
<td>18</td>
<td>9.971</td>
<td>150</td>
<td>13.02</td>
<td>0.773</td>
<td>45</td>
<td>1.591</td>
</tr>
<tr>
<td>19</td>
<td>12.001</td>
<td>346</td>
<td>6.17</td>
<td>1.121</td>
<td>46</td>
<td>4.682</td>
</tr>
<tr>
<td>20</td>
<td>1.804</td>
<td>41</td>
<td>11.98</td>
<td>1.049</td>
<td>47</td>
<td>6.341</td>
</tr>
<tr>
<td>21</td>
<td>0.614</td>
<td>23</td>
<td>6.48</td>
<td>0.826</td>
<td>48</td>
<td>5.865</td>
</tr>
<tr>
<td>22</td>
<td>1.704</td>
<td>59</td>
<td>7.51</td>
<td>1.009</td>
<td>49</td>
<td>5.437</td>
</tr>
<tr>
<td>23</td>
<td>16.979</td>
<td>927</td>
<td>3.59</td>
<td>1.027</td>
<td>50</td>
<td>2.441</td>
</tr>
<tr>
<td>24</td>
<td>8.303</td>
<td>266</td>
<td>7.51</td>
<td>1.009</td>
<td>51</td>
<td>5.437</td>
</tr>
<tr>
<td>25</td>
<td>24.845</td>
<td>405</td>
<td>8.92</td>
<td>1.254</td>
<td>52</td>
<td>4.672</td>
</tr>
<tr>
<td>26</td>
<td>3.116</td>
<td>123</td>
<td>4.42</td>
<td>1.016</td>
<td>53</td>
<td>1.277</td>
</tr>
<tr>
<td>27</td>
<td>6.949</td>
<td>254</td>
<td>12.48</td>
<td>1.496</td>
<td>54</td>
<td>28.267</td>
</tr>
<tr>
<td>28</td>
<td>8.599</td>
<td>237</td>
<td>7.32</td>
<td>2.325</td>
<td>55</td>
<td>0.861</td>
</tr>
</tbody>
</table>

There are no multivariate outliers among 52 rows of data from Table 1 (without rows 35 and 47) since their SMD values do not exceed 14.86, which is the quantile of the Chi-Square distribution, applicable for a significance level of 0.005. That is why we go to the third step.

In the third step, we build the linear regression model (4) for 52 rows of normalized data from Table 1 (without rows 35 and 47). The estimates 1, 2, 3, and 4 equal

-6.2999, 1.8251, 0.86530, and 0.003968, respectively. The estimate 5 of a standard deviation of ε is 0.1548.
In the fourth step, we test the normality of the distribution of residuals in the linear regression model (4) for 52 rows of normalized data from Table 1 (without rows 35 and 47). To achieve this, we employ the Pearson Chi-Squared test. We accepted the null hypothesis \( H_0 \), affirming that the observed frequency distribution of the residuals in (4) closely resembles the normal distribution (indicating no significant difference between the distributions). This decision was reached because the test \( \chi^2 \) statistic, measuring 6.98 does not exceed 9.49, that is the quantile of the Chi-Square distribution, applicable for 4 degrees of freedom and a significance level of 0.05. Therefore, we go to step 5.

In the fifth step, the nonlinear regression model (6) was constructed. Then, in the sixth step, the prediction interval of nonlinear regression by (7) and determine whether one or more values of the response are outliers. The estimate \( \hat{b}_1 = 0.1596 \) value equals 2,0106 for a significance level of 0.05. Therefore, we go to step 5.

In the fifth step, the nonlinear regression model (6) was constructed. Then, in the sixth step, the prediction interval of nonlinear regression by (7) was built and it was determined whether one or more values of the response were outliers. The estimate \( \hat{b}_1 = 0.1596 \) value equals 2,0106 for a significance level of 0.05. Therefore, we go to step 5.

In this case, the inverse matrix of (8) is

\[
S_Z^{-1} = \begin{pmatrix}
0.0807 & 0.0278 & -0.0316 \\
0.0278 & 0.2101 & -0.0138 \\
-0.0316 & -0.0138 & 0.4056
\end{pmatrix}.
\tag{9}
\]

The values of averages (sample means) \( \bar{Z}_1, \bar{Z}_2, \) and \( \bar{Z}_3 \) are 3.459, 1.502, and 0.060, respectively. The \( S_{Z_1} \) value equals 0.1596. The \( t_{0.025} \) value equals 2,0106 for a significance level of 0.05 and 48 degrees of freedom.

There is no multivariate outliers among data of 51 rows from Table 1 (without rows 35, 47, and 51). The estimates \( \hat{b}_1, \hat{b}_2, \) and \( \hat{b}_3 \), which equal \(-0.161328, -0.159392, -0.239815, -1.253682 \) are inside the prediction interval. Therefore, we go to step 1 of the fourth iteration.

In step 1 of the fourth iteration, we normalize the data of 51 rows from Table 1 (without rows 35, 47, and 51). In this case, the parameter estimates of the four-variate Box-Cox transformation for the data from Table 1 (without rows 35, 47, and 51) are calculated by the maximum likelihood method and are \( \hat{\lambda}_Y = -0.161328, \hat{\lambda}_1 = -0.159392, \hat{\lambda}_2 = -0.239815, \hat{\lambda}_3 = -1.253682 \)

There are no multivariate outliers among data of 51 rows from Table 1 (without rows 35, 47, and 51) since their SMD values do not exceed 14.86, which is the quantile of the Chi-Square distribution, applicable for a significance level of 0.005. That is why we go to the third step.

In the third step, we build the linear regression model (4) for 51 rows of normalized data from Table 1 (without rows 35, 47, and 51). The estimates \( \hat{b}_0, \hat{b}_1, \hat{b}_2, \) and \( \hat{b}_3 \) equal \(-6.1084, 1.7898, 0.84037, \) and 0.04182, respectively. The estimate \( \hat{\sigma}_e \) is 0.1421.

In the fourth step, we test the normality of the distribution of residuals in the linear regression model (4) for 51 rows of normalized data from Table 1 (without rows 35, 47, and 51). To achieve this, we employ the Pearson Chi-Squared test. We accepted the null hypothesis \( H_0 \), affirming that the observed frequency distribution of the error values in (4) closely resembles the normal distribution (indicating no significant difference between the distributions). This decision was reached because the test \( \chi^2 \) statistic, measuring 5.57 does not exceed 9.49, that is the quantile of the Chi-Square distribution, applicable for 4 degrees of freedom and a significance level of 0.05. Therefore, we go to step 5.

In the fifth step, we construct the nonlinear regression model (6). Then, in the sixth step, we build the prediction interval of nonlinear regression by (7) and determine whether one or more values of the response are outliers. In our case the inverse matrix of (8) is

\[S_Z^{-1} = \begin{pmatrix}
0.0834 & 0.0284 & -0.0313 \\
0.0284 & 0.2468 & -0.0232 \\
-0.0313 & -0.0232 & 0.4084
\end{pmatrix}.
\]

The values of averages \( \bar{Z}_1, \bar{Z}_2, \) and \( \bar{Z}_3 \) are 3.441, 1.454, and 0.065, respectively. The \( S_{Z_1} \) value equals 0.1466. The \( t_{0.025} \) value equals 2,0117 for a 0.05 significance level and 47 degrees of freedom.

No outliers are present in the data for the nonlinear regression model (6) since all \( Y \) values for 51 rows of the data from Table 1 (without rows 35, 47, and 51) are inside the prediction interval. Therefore, we complete constructing the nonlinear regression model (6).

The nonlinear regression model (6) has the parameter estimates \( \hat{\lambda}_Y, \hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3, \hat{b}_0, \hat{b}_1, \hat{b}_2, \) and \( \hat{b}_3 \), which equal \(-0.161328, -0.159392, -0.239815, -1.253682, -6.1084, 1.7898, 0.84037, \) and 0.04182, respectively. The estimate \( \hat{\sigma}_e \) of a standard deviation \( \varepsilon \) is 0.1421. The nonlinear regression model (6) is limited to estimating LOC of open-source Kotlin-based apps with the following restrictions on predictors: the interval for \( X_1 \) is from 19 to 1292, the interval for \( X_2 \) is from 2.167 to 26.526, and the interval for \( X_3 \) is from 0.681 to 3.206.

To assess the predictive accuracy of the nonlinear regression model (6), we utilized standard metrics namely \( R^2, \) MMRE, and PRED(0.25). The acceptable values of MMRE and PRED(0.25) are not more than 0.25 and not less than 0.75 respectively. For model (6) with the above parameter estimates, predicated upon the four-variate Box-Cox transformation applied to the dataset of the 51 apps from Table 1 (excluding entries from rows 35, 47, and 51), the computed values for \( R^2, \) MMRE, and PRED(0.25) are 0.9235, 0.1458, and 0.8235, respectively.

These values indicate good model quality. However, the data from the table 1 is the training set. To avoid the problem of overfitting the model [32], the predictive accuracy of the model (6) should be checked on the test set, the data of which were not used to build the model. That
we do next. In addition, we compare the built model (6) with two other models that are obtained based on the univariate transformations.

4 EXPERIMENTS

The test dataset was obtained using the CodeMR tool [30] around the variables and for the training set from Table 1. Table 2 contains the test dataset.

For comparison of the model (6) with other nonlinear regression models with three predictors, two nonlinear regression models are built based on normalizing the data of 51 rows from Table 1 (without rows 35, 47, and 51) using the univariate transformation.

The nonlinear regression model based on the univariate regression model (4) for the normalized data and the decimal logarithm univariate transformation has the form

\[ Y = 10^{c+b_1X_1+b_2X_2+b_3X_3}, \]

where the estimators for parameters are: \( \hat{b}_0 = -2.11196 \), \( \hat{b}_1 = 1.02339 \), \( \hat{b}_2 = 0.652935 \), \( \hat{b}_3 = 0.047833 \). The estimate \( \hat{\sigma}_e \) is 0.08605.

\[
\begin{array}{cccc}
\text{No} & \text{App name} & Y & X_1 & X_2 \\
1 & moko-resources & 3.996 & 173 & 3.908 \\
2 & Loritta & 79.295 & 2815 & 5.304 \\
3 & kable & 1.818 & 56 & 4.393 \\
4 & binary-compatibility-validator & 1.149 & 46 & 5.377 \\
5 & contacts-android & 27.109 & 1149 & 5.148 \\
6 & strikt & 2.418 & 74 & 8.676 \\
7 & kotlin-power-assert & 1.017 & 29 & 6.172 \\
8 & kroto-plus & 10.909 & 153 & 7.943 \\
9 & Hexagon & 6.65 & 212 & 7.943 \\
10 & badge & 11.132 & 636 & 5.634 \\
11 & MoshiX & 3.35 & 154 & 8.260 \\
12 & BleGattCoroutines & 1.834 & 46 & 5.261 \\
13 & kotlin-spark-api & 13.344 & 216 & 23.685 \\
14 & data2viz & 1.573 & 51 & 6.137 \\
15 & Confetti & 14.425 & 313 & 6.042 \\
16 & locus-android & 0.842 & 25 & 5.81 \\
17 & actions-on-google-java & 4.542 & 149 & 4.694 \\
18 & aws-sdk-kotlin & 6.556 & 300 & 3.467 \\
19 & moko-kwift & 1.023 & 41 & 5.366 \\
20 & EzXHelper & 1.557 & 49 & 9.939 \\
21 & ktgbotapi & 0.539 & 24 & 3.292 \\
22 & detekt-intellij-plugin & 0.934 & 42 & 4.811 \\
23 & swiftpoint & 2.883 & 84 & 9.881 \\
24 & Pick-Storage-Access-Framework & 2.821 & 56 & 12.411 \\
25 & kowasm & 4.629 & 389 & 2.956 \\
\end{array}
\]

The prediction results \( Y \) of models (6) and (10) for values of predictors from Table 2 and values of MRE are presented in Table 3. Prediction results obtained from model (6) and the corresponding MRE values are presented in Table 3, showcasing two cases: utilizing univariate and four-variate Box-Cox transformations. The MRE values for model (6) based on the Box-Cox four-variate transformation exhibit a reduction compared to those of model (6) based on the Box-Cox univariate transformation for 15 from 25 rows of data (rows 1–6, 8, 10, 11, 17, 19–22, 25). Also, the MRE values for model (6) based on the Box-Cox univariate transformation are less than for model (10) based on the decimal logarithm univariate transformation for 16 from 25 rows of data (rows 3, 5–8, 10–12, 14–17, 20, 21, 23, 25).
Note, that a more significant advantage of the model (6) constructed by the four-variate Box-Cox transformation compared with the two above models relying on univariate transformations, is the reduced widths of the confidence and prediction intervals. These intervals are defined by data from Table 2. Table 3 contains the lower (LB) and upper (UB) bounds of prediction intervals we obtained for the data of app 2 is from Table 2.

Additionally, for 18 of the 25 data rows (excluding rows 2, 5, 9–11, 13, and 15), the confidence intervals’ widths for nonlinear regression, based on the Box-Cox four-variate transformation, are less than for nonlinear regression based on the Box-Cox univariate transformation for 20 from 25 data rows (except rows 8, 9, 10, 11, 13, and 15 with the difference up to 27%) from 25 data rows (except rows 8, 9, 10, 13, and 15 with the difference of 7.8, 5.4, 14.3, 1.4, 5.2, 35.8, and 0.8%, respectively) and less than after decimal logarithm univariate transformation for 18 from (with the difference up to 27%) from 25 data rows (except rows 2, 5, 9, 10, 11, 13, and 15 with the difference of 20.3, 1.5, 1.8, 45.8, and 0.9%, respectively).

The largest deviation between the widths of the intervals we obtained for the data of app 2 is from Table 2. That result can be explained by the fact that the value 2815 of the predictor $X_i$ exceeds the upper bound of the corresponding restriction ($X_i$ is from 19 to 1292 according to the training dataset from Table 1), for which model (6) was built, by more than two times.
6 DISCUSSION

Utilizing appropriate techniques, we employ four-variate normalizing transformations to construct the nonlinear regression model for early estimation of LOC in open-source Kotlin-based applications, as in [13]. This approach is chosen due to the non-Gaussian distribution of residuals in the linear regression model, as the chi-squared test result indicated. Moreover, the four-variate distribution of the data from Table 1 is not Gaussian what the Mardia multivariate normality test based on measures of the multivariate skewness and kurtosis indicates. We utilize the statistical technique based on the multivariate normalizing transformations and the SMD for normalized data to detect four-variate outliers in the non-Gaussian data from Table 1. Note, that we have more four-variate outliers for the data from Table 1 without applying normalization.

For a larger number of data rows, the widths of both confidence and prediction intervals in multiple nonlinear regression, utilizing the Box-Cox four-variate transformation, are smaller compared to nonlinear regressions employing univariate transformations, including both the decimal logarithm and the Box-Cox. Moreover, model (6) utilizing the Box-Cox four-variate transformation demonstrates a smaller MMRE value compared with all other nonlinear models employing univariate transformations. This may prove the Box-Cox four-variate transformation to be the best four-variate normalization transformation for non-Gaussian data from Table 1.

The advantages of the proposed model (6) include the possibility of early LOC estimation of open-source Kotlin-based apps using the values of three metrics at the app level (the total number of classes, WMC, and DIT), that can be measured from the class diagram. The disadvantages of the proposed model (6) include, first of all, the fact that the early LOC estimation can be performed only for a part of the open-source Kotlin-based apps. The proposed model (6) is limited to the early LOC estimation of open-source Kotlin-based apps for which there are the following restrictions on predictors: the interval for $X_1$ is from 19 to 1292, the interval for $X_2$ is from 2.167 to 26.526, and the interval for $X_3$ is from 0.681 to 3.206.

The obtained results indicate that a constructed model with three predictors for early LOC estimation of open-source Kotlin-based apps improves confidence in estimating the LOC metric of the above apps.

CONCLUSIONS

The task of improving confidence in early LOC estimation for open-source Kotlin-based applications has been accomplished.

The scientific novelty of the obtained results is that the three-factor nonlinear regression model for early LOC estimation of open-source Kotlin-based apps is firstly constructed based on the Box-Cox four-variate transformation. Compared to the other nonlinear regression models, this model demonstrates a smaller mean magnitude of relative error and narrower confidence and prediction intervals with three predictors for more cases.

The practical significance of the obtained results is that the computer program to implement the constructed model using sci-language for Scilab was developed. With the experimental results at hand, we are confident in recommending the developed model for practical use.

Prospects for further research may include the application of other multivariate normalizing transformations and data sets to construct multiple nonlinear regression models for early LOC estimation of open-source Kotlin-based apps for other restrictions on predictors.

ACKNOWLEDGEMENTS

This work is proactive. The research was performed within the scope of the scientific activity of the authors’ working hours according to the main positions.

REFERENCES


29. Prykhodko S., Prykhodko N., Makarova L. et al. Outlier Detection in Non-Linear Regression Analysis Based on the
НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ РАННЬОГО ОЦІНЮВАННЯ МЕТРИКИ LOC ЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА KOTLIN

Приходько С. Б. – д-р техн. наук, професор, завідувач кафедри програмного забезпечення автоматизованих систем Національного університету кораблебудування імені адмірала Макарова, Миколаїв, Україна.

Приходько Н. В. – канд. екон. наук, доцент, доцент кафедри фінансів Національного університету кораблебудування імені адмірала Макарова, Миколаїв, Україна.

Кольцов А. В. – аспірант кафедри програмного забезпечення автоматизованих систем Національного університету кораблебудування імені адмірала Макарова, Миколаїв, Україна.

АННОТАЦIЯ

Актуальність. Раннє оцінювання рядків коду (LOC) у проектах програмного забезпечення має важливе значення, оскільки це безпосередньо впливає на прогнозування зусиль з розробки програмного забезпечення для цілого спектру мов програмування, включаючи застосунки з відкритим кодом на Kotlin. Об’єктом дослідження є процес раннього оцінювання метрики LOC застосунків з відкритим кодом на Kotlin. Підстановка дослідження є відкритій реґресійній моделі для раннього оцінювання метрики LOC застосунків з відкритим кодом на Kotlin.

Мета. Метою роботи є побудова нелінійної регресійної моделі з трьома предикторами для раннього оцінювання метрики LOC застосунків з відкритим кодом на Kotlin на основі чотиризольного нормалізуючого перетворення Boks-Koksa для підвищення достовірності раннього оцінювання LOC цих застосунків.

Метод. Для раннього оцінювання LOC у застосунках із відкритим кодом на Kotlin модель, довірчі та прогнозні інтервали нелінійної регресії були побудовані за допомогою нормалізуючого перетворення Boks-Koksa з чотирма змінними та за допомогою відповідних методів. Ці методи базуються на множинному нелінійному регресійному аналізі з використанням багатовимірних нормалізуючих перетворень та враховують кореляцію між залежними та незалежними змінними у випадку негаусових даних. Як наслідок, такий підхід має тенденцію до зменшення середньої величини відносно похибки, зменшення ширини довірчих інтервалів та інтервалів прогнозування порівняно з моделями, що використовують однофакторні нормалізуючі перетворення.

Результати. Проведено порівняння побудованої моделі з моделями нелінійної реґресії з використанням десяткового логарифму та одновимірного перетворення Boks-Koksa.

КЛЮЧОВІ СЛОVA: оцінка, рядки коду, застосунок з відкритим вихідним кодом, Kotlin, нелінійна регресійна модель, перетворення Boks-Koksa, клас, змішані методи на клас, глибина дерева успадкування.

ЛІТЕРАТУРА