UDC 004.412:519.237.5

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. The prospects for further research may include the application of other data sets 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

$\hat{\mathbf{b}}$ 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;

$\mathbf{P}$ is a non-Gaussian random vector;

R^2 is a multiple coefficient of determination;

$\mathbf{S_Z}$ is a sample covariance matrix for normalized data;

SMD_Z is a squared Mahalanobis distance for normalized data;

$\mathbf{T}$ is a Gaussian random vector;

$t_{\alpha/2,\nu}$ is a quantile of the student's t-distribution with ν 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;

α is a significance level;

DOI 10.15588/1607-3274-2024-1-8

β_1 is a multivariate skewness;

β_2 is a multivariate kurtosis;

ε is a Gaussian random variable that defines residuals;

ν is a number of degrees of freedom;

σ_ε is a standard deviation of ε;

$\boldsymbol{\psi}$ 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

$$\mathbf{T} = \boldsymbol{\psi}(\mathbf{P}) \qquad (1)$$

and the inverse transformation for (1)

$$\mathbf{P} = \boldsymbol{\psi}^{-1}(\mathbf{T}). \qquad (2)$$

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, 23–28]. 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.

DOI 10.15588/1607-3274-2024-1-8

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

$$Z_j = x(\lambda_j) = \begin{cases} \left(X_j^{\lambda_j} - 1\right)\Big/\lambda_j, & \text{if} \quad \lambda_j \neq 0; \\ \ln(X_j), & \text{if} \quad \lambda_j = 0. \end{cases} \tag{3}$$

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

$$Z_Y = \hat{Z}_Y + \varepsilon = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \hat{b}_3 Z_3 + \varepsilon, \tag{4}$$

ε is a Gaussian random variable that defines residuals, $\varepsilon \sim N\left(0, \sigma_\varepsilon^2\right)$.

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

$$Y = \psi_Y^{-1}\left(\hat{Z}_Y + \varepsilon\right). \tag{5}$$

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

$$Y = \left[\hat{\lambda}_Y \left(\hat{Z}_Y + \varepsilon\right) + 1\right]^{1/\hat{\lambda}_Y}, \tag{6}$$

where ε is a Gaussian random variable, $\varepsilon \sim N\left(0, \sigma_\varepsilon^2\right)$, 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]

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\alpha/2,\nu} S_{Z_Y} \left\{1 + \frac{1}{N} + \left(\mathbf{z}_X^+\right)^T \mathbf{S}_Z^{-1} \left(\mathbf{z}_X^+\right)\right\}^{1/2}\right), \tag{7}$$

where $t_{\alpha/2,\nu}$ is a student's t-distribution quantile with $\alpha/2$ significance level and ν degrees of freedom; $\nu = N - k - 1$; k is the number of independent variables (in our case, k is 3); $\mathbf{z}_X^+$ is a vector with components $Z_{1_i} - \bar{Z}_1$, $Z_{2_i} - \bar{Z}_2$, …, $Z_{k_i} - \bar{Z}_k$ for i-row; $\bar{Z}_j = \frac{1}{N}\sum_{i=1}^{N} Z_{j_i}$, $j = 1,2,\ldots,k$; $S_{Z_Y}^2 = \frac{1}{\nu}\sum_{i=1}^{N}\left(Z_{Y_i} - \hat{Z}_{Y_i}\right)^2$, $\nu = N - k - 1$; $\mathbf{S}_Z$ is a $k \times k$ matrix

DOI 10.15588/1607-3274-2024-1-8

$$\mathbf{S}_Z = \begin{pmatrix} S_{Z_1Z_1} & S_{Z_1Z_2} & \dots & S_{Z_1Z_k} \\ S_{Z_1Z_2} & S_{Z_2Z_2} & \dots & S_{Z_2Z_k} \\ \dots & \dots & \dots & \dots \\ S_{Z_1Z_k} & S_{Z_2Z_k} & \dots & S_{Z_kZ_k} \end{pmatrix}. \quad (8)$$

In (8) $S_{Z_qZ_r} = \sum_{i=1}^{N} \left[Z_{q_i} - \overline{Z}_q\right]\left[Z_{r_i} - \overline{Z}_r\right],\ q,r = 1,2,\dots,k$.

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 CodeMR tool [30], focusing on the following variables: the actual software size measured in thousand lines of code (KLOC) Y, the total number of classes X_1, the WMC metric at the application level X_2, 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 checked the four-dimensional data from Table 1 for multivariate outliers. Before analyzing the four-dimensional data from Table 1 for multivariate outliers, we assessed the normality of the multivariate data in Table 1. This preliminary check was essential, as common statistical methods, including multivariate outlier detection based on the squared Mahalanobis distance (SMD), are designed to identify outliers assuming a Gaussian distribution. We applied a multivariate normality test proposed by Mardia and based on measures of the multivariate skewness β_1 and kurtosis β_2 [31]. According to this test, the distribution of four-dimensional data from Table I is not Gaussian since the test statistic for multivariate skewness $N\beta_1/6$ of this data exceeds 40.00, that is the quantile of the Chi-Square distribution, applicable for 20 degrees of freedom and a for significance level of 0.005.

Similarly, the test statistic for multivariate kurtosis β_2, which equals 66.74, is greater than the value of the Gaussian distribution quantile, which is 28.86 for 24 mean, 3.56 variance, and 0.005 significance level. Because, as in [17], to detect multivariate outliers in the four-dimensional non-Gaussian data from Table I, we used the statistical technique based on the multivariate normalizing transformations and the SMD for normalized data. To normalize the data from Table 1, 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 $\hat{\lambda}_Y = -0.137228$, $\hat{\lambda}_1 = -0.138740$, $\hat{\lambda}_2 = -0.220743$, $\hat{\lambda}_3 = -1.067093$.

There are two multivariate outliers in four-dimensional non-Gaussian data in Table 1 since the SMD_Z 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

No	Y	X_1	X_2	X_3	No	Y	X_1	X_2	X_3
1	1.682	73	5.37	0.959	28	4.111	157	5.82	1.408
2	1.256	55	3.56	1.473	29	12.248	331	6.12	0.894
3	14.867	546	5.40	0.987	30	0.829	31	5.71	0.871
4	23.149	1033	4.60	3.206	31	22.069	623	7.76	0.681
5	28.699	1090	5.33	1.156	32	2.179	92	5.78	1.424
6	8.303	266	7.25	1.711	33	11.800	199	11.82	1.472
7	5.508	122	9.66	1.648	34	1.425	77	4.10	1.429
8	22.078	1292	4.71	0.971	**35**	**2.220**	**221**	**3.67**	**0.561**
9	2.010	47	7.68	0.851	36	1.872	101	4.25	1.198
10	11.629	425	4.95	1.416	37	2.097	91	3.64	0.934
11	1.728	69	6.22	1.116	38	5.639	313	3.82	1.125
12	1.538	102	2.17	0.873	39	1.933	80	6.25	1.125
13	38.101	1151	6.85	0.929	40	17.575	776	4.30	0.932
14	7.655	290	5.90	1.062	41	5.437	167	7.59	1.192
15	4.975	354	3.91	0.833	42	2.357	83	3.59	1.145
16	24.324	701	4.48	1.126	43	0.762	24	6.71	1.000
17	1.031	19	26.53	1.053	44	1.201	37	8.00	0.865
18	9.971	150	13.02	0.773	45	1.591	42	14.36	1.095
19	12.001	346	6.17	1.121	46	4.682	212	4.46	1.236
20	1.804	41	11.98	1.049	**47**	**6.341**	**55**	**30.80**	**6.073**
21	0.614	23	6.48	0.826	48	5.865	207	9.44	1.048
22	1.704	59	7.51	1.000	49	5.443	179	5.44	0.849
23	16.979	927	3.59	1.027	50	2.441	136	3.71	0.794
24	25.845	405	8.92	1.254	51	4.672	66	13.58	0.879
25	3.116	123	4.42	1.016	52	1.727	74	3.41	1.838
26	9.494	254	12.48	1.496	53	28.267	595	13.54	1.183
27	8.599	237	7.32	2.325	54	0.861	44	4.75	0.773

There are no multivariate outliers among 52 rows of data from Table 1 (without rows 35 and 47) since their SMD_Z 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 $\hat{b}_0$, $\hat{b}_1$, $\hat{b}_2$, and $\hat{b}_3$ equal –6.2999, 1.8251, 0.86530, and 0.003968, respectively. The estimate $\hat{\sigma}_\varepsilon$ of a standard deviation of ε is 0.1548.

DOI 10.15588/1607-3274-2024-1-8

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 ε values in (4) closely resembles the normal distribution (indicating no significant difference between the distributions). This decision was reached because the test χ^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) was built and it was determined whether one or more values of the response (dependent random variable) were outliers.

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

$$\mathbf{S}_Z^{-1} = \begin{pmatrix} 0.0807 & 0.0278 & -0.0316 \\ 0.0278 & 0.2101 & -0.0138 \\ -0.0316 & -0.0138 & 0.4056 \end{pmatrix}.$$

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

There is one outlier in the data for the nonlinear regression model (6) since the Y value for row 51 is outside the prediction interval. Therefore, we discard row 51 and 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_Z 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 $\hat{\sigma}_\varepsilon$ 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 ε values in (4) closely resembles the normal distribution (indicating no significant difference between the distributions). This decision was reached because the test χ^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

$$\mathbf{S}_Z^{-1} = \begin{pmatrix} 0.0834 & 0.0284 & -0.0313 \\ 0.0284 & 0.2468 & -0.0232 \\ -0.0313 & -0.0232 & 0.4084 \end{pmatrix}. \tag{9}$$

The values of averages $\overline{Z}_1$, $\overline{Z}_2$, and $\overline{Z}_3$ are 3.441, 1.454, and 0.065, respectively. The S_{Z_Y} value equals 0.1466. The $t_{\alpha/2,\nu}$ 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 $\hat{\sigma}_\varepsilon$ of a standard deviation ε 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

DOI 10.15588/1607-3274-2024-1-8

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 linear regression model (4) for the normalized data and the decimal logarithm univariate transformation has the form

$$Y = 10^{\varepsilon + \hat{b}_0} X_1^{\hat{b}_1} X_2^{\hat{b}_2} X_3^{\hat{b}_3}, \qquad (10)$$

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 $\hat{\sigma}_\varepsilon$ is 0.08605.

Table 2 – The test dataset

No	App name	Y	X_1	X_2	X_3
1	moko-resources	3.996	173	3.908	1.145
2	Loritta	79.295	2815	5.304	1.497
3	kable	1.818	56	4.393	1.054
4	binary-compatibility-validator	1.149	46	5.37	1.022
5	contacts-android	27.109	1149	5.148	1.55
6	strikt	2.418	74	8.676	1.203
7	kotlin-power-assert	1.017	29	6.172	1.172
8	kroto-plus	10.909	153	17.353	1.007
9	Hexagon	6.65	212	7.943	1
10	beagle	11.132	636	5.634	1.06
11	MoshiX	8.35	154	8.26	0.721
12	BleGattCoroutines	1.834	46	5.261	1.457
13	kotlin-spark-api	13.344	216	23.685	1.167
14	data2viz	1.573	51	6.137	1.078
15	Confetti	14.425	313	6.042	1.201
16	locus-android	0.842	25	5.8	1.64
17	actions-on-google-java	4.542	149	4.604	0.872
18	aws-sdk-kotlin	6.556	300	3.467	1.107
19	moko-kswift	1.023	41	5.366	1.195
20	EzXHelper	1.557	49	9.939	1.082
21	ktgbotapi	0.539	24	3.292	1.042
22	detekt-intellij-plugin	0.934	42	4.81	1.595
23	swiftpoet	2.883	84	9.881	0.893
24	Fuck-Storage-Access-Framework	2.821	56	12.411	1.036
25	kowasm	4.629	389	2.956	0.817

The nonlinear regression model based on the Box-Cox univariate transformation is analogously (6) with the only difference being that the data for variables are normalized by the Box-Cox univariate transformation using the maximum likelihood method. The estimators for parameters of the Box-Cox univariate transformation for each from variables Y, X_1, X_2, and X_3 are $\hat{\lambda}_Y = -0.067289$, $\hat{\lambda}_1 = -0.034850$, $\hat{\lambda}_2 = -0.459658$, $\hat{\lambda}_3 = -1.289541$. The parameter estimators of the linear regression model for normalized data by the Box-Cox univariate transformation are $\hat{b}_0 = -5.33943$, $\hat{b}_1 = 1.09177$, $\hat{b}_2 = 1.38983$, $\hat{b}_3 = 0.069187$. The estimate $\hat{\sigma}_\varepsilon$ is 0.17507.

To carry out experiments, a computer program was developed to implement the constructed models (6) and (10). The program was coded using sci-language compatible with the Scilab system. Scilab, accessible at http://www.scilab.org, stands as a cost-free, open-source software option.

5 RESULTS

The values of R^2, MMRE and PRED(0.25) equal respectively 0.9138, 0.1538, and 0.8431 for the model (6) based on the Box-Cox univariate transformation, and equal respectively 0.9166, 0.1548, and 0.8039 for the model (10) for the decimal logarithm univariate transformation for the training dataset. In this case, the MMRE and R^2 values are better for the model (6) based on the Box-Cox four-variate transformation. The PRED(0.25) value is better for model (6) based on the Box-Cox univariate transformation (0.8431 against 0.8235).

For model (6) with the parameter estimates, calculated upon the four-variate Box-Cox transformation for the training dataset of the 51 apps from Table 1 (excluding entries from rows 35, 47, and 51) applied to the test dataset of the 25 apps from Table 2, the computed values for R^2, MMRE, and PRED(0.25) are 0.9818, 0.1871, and 0.7600, respectively. For model (10) applied to the test dataset of the 25 apps from Table 2, the computed values for R^2, MMRE, and PRED(0.25) are 0.9811, 0.1925, and 0.7600, respectively. For model (6) with the parameter estimates, calculated upon the univariate Box-Cox transformation for the dataset of the 51 apps from Table 1 (excluding entries from rows 35, 47, and 51) applied to the test dataset of the 25 apps from Table 2, the computed values for R^2, MMRE, and PRED(0.25) are 0.9619, 0.1943, and 0.7200, respectively. In the case of the test dataset, the MMRE and R^2 values are better for the model (6) based on the Box-Cox four-variate transformation too.

The prediction results $\hat{Y}$ 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 four-variate 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).

DOI 10.15588/1607-3274-2024-1-8

Table 3 – The prediction results and confidence intervals of nonlinear regressions

No	Y	The four-variate Box-Cox transformation				The univariate transformations: the decimal logarithm				The univariate transformations: the Box-Cox			
		$\hat{Y}$	MRE	LB	UB	$\hat{Y}$	MRE	LB	UB	$\hat{Y}$	MRE	LB	UB
1	3.996	3.631	0.0914	3.373	3.912	3.696	0.0750	3.407	4.009	3.598	0.0996	3.317	3.905
2	79.295	77.070	0.0281	62.905	95.076	79.378	0.0010	67.528	93.307	89.295	0.1261	73.915	108.140
3	1.818	1.302	0.2839	1.213	1.399	1.253	0.3109	1.139	1.378	1.296	0.2873	1.191	1.411
4	1.149	1.206	0.0493	1.127	1.291	1.166	0.0149	1.064	1.278	1.226	0.0671	1.132	1.329
5	27.109	30.183	0.1134	25.990	35.183	31.167	0.1497	27.473	35.358	32.565	0.2013	28.256	37.583
6	2.418	2.613	0.0808	2.435	2.807	2.615	0.0815	2.409	2.838	2.638	0.0911	2.431	2.864
7	1.017	0.822	0.1918	0.757	0.893	0.802	0.2117	0.718	0.895	0.864	0.1504	0.783	0.954
8	10.909	8.639	0.2081	7.505	9.976	8.574	0.2140	7.396	9.940	7.706	0.2936	6.752	8.805
9	6.65	7.412	0.1146	6.877	7.997	7.185	0.0804	6.668	7.741	7.233	0.0876	6.704	7.807
10	11.132	17.686	0.5887	15.972	19.617	17.722	0.5920	16.131	19.470	18.146	0.6301	16.450	20.030
11	8.35	5.318	0.3631	4.646	6.105	5.232	0.3734	4.664	5.869	5.163	0.3817	4.475	5.964
12	1.834	1.206	0.3425	1.105	1.317	1.170	0.3619	1.044	1.312	1.234	0.3274	1.112	1.369
13	13.344	15.695	0.1762	13.006	19.052	15.057	0.1284	12.498	18.141	12.753	0.0443	10.858	15.005
14	1.573	1.448	0.0792	1.359	1.545	1.418	0.0987	1.304	1.541	1.481	0.0586	1.373	1.598
15	14.425	9.154	0.3654	8.509	9.855	9.033	0.3738	8.443	9.663	9.164	0.3647	8.523	9.857
16	0.842	0.691	0.1794	0.622	0.769	0.672	0.2018	0.578	0.781	0.733	0.1295	0.644	0.835
17	4.542	3.486	0.2324	3.233	3.763	3.485	0.2327	3.204	3.791	3.451	0.2402	3.179	3.749
18	6.556	5.601	0.1457	5.109	6.148	5.995	0.0856	5.458	6.584	5.626	0.1419	5.096	6.215
19	1.023	1.080	0.0560	1.001	1.166	1.044	0.0204	0.944	1.154	1.107	0.0822	1.012	1.212
20	1.557	1.816	0.1666	1.680	1.966	1.865	0.1976	1.690	2.057	1.865	0.1981	1.701	2.047
21	0.539	0.483	0.1046	0.435	0.536	0.436	0.1915	0.375	0.507	0.469	0.1301	0.411	0.535
22	0.934	1.048	0.1217	0.950	1.157	1.010	0.0814	0.884	1.154	1.068	0.1435	0.950	1.202
23	2.883	3.179	0.1027	2.920	3.466	3.195	0.1083	2.901	3.520	3.147	0.0914	2.860	3.463
24	2.821	2.353	0.1658	2.148	2.581	2.466	0.1258	2.202	2.762	2.367	0.1611	2.134	2.626
25	4.629	6.135	0.3253	5.359	7.044	6.946	0.5004	6.093	7.918	6.150	0.3285	5.315	7.126

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 the confidence intervals of nonlinear regressions utilizing both univariate and four-variate transformations, with a significance level of 0.05. We defined the confidence intervals for $\hat{Y}$ using (7) with the sole distinction being the absence of 1 in the summation within curly brackets. Also, we used the inverse matrix (9) in this case. The widths of the confidence interval of 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 rows of data (except rows 8–10, 13, and 15).

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 those based on the decimal logarithm univariate transformation. Similar results are observed in the prediction intervals of nonlinear regressions using the test dataset from Table 2.

The lower (LB) and upper (UB) bounds of prediction intervals for nonlinear regressions, based on univariate and four-variate transformations respectively, are presented in Table 4 at a significance level of 0.05. It's worth noting that the width of the confidence interval for nonlinear regression for the four-variate Box-Cox transformation is less than after the univariate Box-Cox transformation for 20 (with the difference up to 23%) from 25 data rows (except rows 8, 9, 10, 13, and 15 with the difference of 20.3, 1.5, 1.8, 45.8, and 0.9%, 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 24.8, 16.6, 1.8, 4.4, 9.1, 21.1, and 7.1%, respectively). It's also worth noting that the width of the confidence interval for nonlinear regression for four-variate Box-Cox transformation is less than after the univariate Box-Cox transformation for 18 (with the difference up to 26.4%) from 25 data rows (except rows 2, 5, 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 30.7%) from 25 data rows (except rows 2, 5, 8, 9, 10, 13, and 15 with the difference of 48.1, 25.1, 3.6, 3.8, 16.9, 18.9, and 5.9%, 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_1 exceeds the upper bound of the corresponding restriction (X_1 is from 19 to 1292 according to the training dataset from Table 1), for which model (6) was built, by more than two times.

DOI 10.15588/1607-3274-2024-1-8

Table 4 – The bounds of the prediction interval

No	Y	univariate				four-variate	
		decimal logarithm		Box-Cox		Box-Cox	
		LB	UB	LB	UB	LB	UB
1	3.996	2.431	5.620	2.415	5.421	2.533	5.320
2	**79.295**	**51.031**	**123.471**	**53.201**	**152.706**	**42.332**	**149.615**
3	1.818	0.821	1.911	0.891	1.903	0.957	1.800
4	1.149	0.765	1.777	0.845	1.796	0.890	1.659
5	27.109	20.273	47.915	20.288	53.093	18.104	52.688
6	2.418	1.720	3.977	1.784	3.943	1.857	3.753
7	1.017	0.524	1.227	0.598	1.261	0.615	1.115
8	10.909	5.539	13.272	5.007	12.012	5.642	13.651
9	6.65	4.731	10.912	4.771	11.097	4.964	11.380
10	11.132	11.624	27.020	11.622	28.727	11.141	29.140
11	8.35	3.414	8.018	3.378	7.990	3.577	8.123
12	1.834	0.764	1.793	0.845	1.818	0.886	1.668
13	13.344	9.588	23.647	8.104	20.359	9.724	26.369
14	1.573	0.932	2.157	1.017	2.178	1.061	2.011
15	14.425	5.955	13.701	6.010	14.147	6.053	14.259
16	0.842	0.434	1.041	0.504	1.075	0.517	0.937
17	4.542	2.291	5.302	2.318	5.195	2.437	5.098
18	6.556	3.932	9.139	3.718	8.614	3.801	8.470
19	1.023	0.684	1.594	0.763	1.621	0.800	1.482
20	1.557	1.222	2.846	1.269	2.770	1.312	2.561
21	0.539	0.281	0.675	0.325	0.682	0.366	0.644
22	0.934	0.656	1.556	0.732	1.575	0.772	1.444
23	2.883	2.095	4.875	2.112	4.739	2.229	4.633
24	2.821	1.610	3.778	1.597	3.545	1.673	3.376
25	4.629	4.511	10.693	4.002	9.572	4.094	9.457

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 errors 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 models 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

1. Ponnala R., Reddy C. R. K. Object Oriented Dynamic Metrics in Software Development: A Literature Review, *International Journal of Applied Engineering Research*, 2019, Vol. 14, No. 22, pp. 4161–4172.
2. Boehm B. W., Abts C., Brown A. W. et al. Software cost estimation with COCOMO II. Upper Saddle River, NJ: Prentice Hall PTR, 2000, 506 p.
3. Rumiński B. Top Apps Built with Kotlin Multiplatform [2023 Update] [Electronic resource]. Access mode: https://www.netguru.com/blog/top-apps-built-with-kotlin-multiplatform
4. Kaczmarek J., Kucharski M. Size and effort estimation for applications written in Java, *Information and Software Technology,* 2004, Vol. 46, Issue 9, pp. 589–601. DOI: 10.1016/j.infsof.2003.11.001
5. Laird L. M., Brennan M. C. Software measurement and estimation. A practical approach. quantitative software engineering series. Wiley-IEEE Computer Society Press, 2006, 379 p.

DOI 10.15588/1607-3274-2024-1-8

6. Tan H. B. K., Zhao Y., Zhang H. Conceptual data model-based software size estimation for information systems, *Transactions on Software Engineering and Methodology*, 2009, Vol. 19, Issue 2, pp. 1–37. DOI: 10.1145/1571629.1571630
7. Zifen Y. An improved software size estimation method based on object-oriented approach, *Electrical & Electronics Engineering : IEEE Symposium EEESYM'12,* Kuala Lumpur, Malaysia, 24–27 June 2012, proceedings. Los Alamitos: IEEE, 2012, pp. 615–617. DOI: 10.1109/EEESym.2012.6258733.
8. Kiewkanya M., Surak S. Constructing C++ software size estimation model from class diagram, *Computer Science and Software Engineering : 13th International Joint Conference, Khon Kaen, Thailand, 13–15 July 2016 : proceedings.* – Los Alamitos: IEEE, 2016, pp. 1–6. DOI: 10.1109/JCSSE.2016.7748880
9. Sholiq S., Dewi R. S., Subriadi A. P. A comparative study of software development size estimation method: UCPabc vs Function Points, Procedia Computer Science, 2017, Vol. 124, pp. 470–477. DOI: 10.1016/j.procs.2017.12.179
10. Prykhodko N. V., Prykhodko S. B. The non-linear regression model to estimate the software size of open source Java-based systems, *Radio Electronics, Computer Science, Control*, 2018, No. 3 (46), pp. 158–166. DOI: 10.15588/1607-3274-2018-3-17
11. Prykhodko S. B., Prykhodko N. V., Smykodub T. G. Chotyrokhfaktorna neliniyna regresiyna model dlia otsiniuvannia rozmiru Java-zastosunkiv z vidkrytym kodom, *Vcheni zapysky TNU imeni V.I. Vernads'kogo. Serija: tehnichni nauk*y, 2020, Vol. 31 (70), Issue 2, pp. 157–162. DOI: https://doi.org/10.32838/2663-5941/2020.2-1/25
12. Prykhodko S. B., Shutko I. S., Prykhodko A. S. A nonlinear regression model to estimate the size of web apps created using the CakePHP framework, *Radio Electronics, Computer Science, Control,* 2021, No. 4 (59), pp. 129–139. DOI: 10.15588/1607-3274-2021-4-12
13. Manisha, Rishi, R. Early size estimation using machine learning, *Proceedings of the* 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 17–19 March 2021 : proceedings. Los Alamitos, IEEE, 2021, pp. 757–762. DOI: 10.1109/INDIACom51348.2021.00135
14. Daud M., Malik A. A. Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (ADAFs), *IEEE Access*, 2021, Vol. 9, pp. 81986–81999. DOI: 10.1109/ACCESS.2021.3085752
15. Zhang K., Wang X., Ren J. et al. Efficiency improvement of function point-based software size estimation with deep learning model. Los Alamitos, IEEE, 2021, Vol. 9, pp. 107124–107136. DOI: 10.1109/ACCESS.2020.2998581
16. Ritu, Garg Y. Comparative Analysis of Machine Learning Techniques in Effort Estimation, *Proceedings of the 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), Faridabad, India, 2022 : proceedings.* Los Alamitos, IEEE, 2022. pp. 401–405. DOI: 10.1109/COM-IT-CON54601.2022.9850592
17. Brar P., Nandal D. A Systematic Literature Review of Machine Learning Techniques for Software Effort Estimation Models, *Proceeding of the 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT).* Sonepat, India, 08–09 July 2022 – Los Alamitos, IEEE, 2022, pp. 494–499, DOI: 10.1109/CCiCT56684.2022.00093.
18. Assefa Y., Berhanu F., Tilahun A. et al. Software Effort Estimation using Machine learning Algorithm, *Proceeding of the 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 28–30 November 2022 : proceedings.* Los Alamitos, IEEE, 2022, pp. 163–168, DOI: 10.1109/ICT4DA56482.2022.9971209
19. Jadhav A., Shandilya S. K. , Izonin I. et al. Effective Software Effort Estimation Leveraging Machine Learning for Digital Transformation, *IEEE Access*, 2023, Vol. 11, pp. 83523–83536. DOI: 10.1109/ACCESS.2023.3293432
20. Kumar S., Arora M., Sakshi et al. A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques, *Proceedings of the 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 21–23 September 2022 : proceedings.* Los Alamitos, IEEE, pp. 416–422. DOI: 10.1109/ICIRCA54612.2022.9985542
21. R. K. B. N. Software Effort Estimation using ANN (Back Propagation), *Proceedings of the 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 23–25 February 2023 : proceedings.* Los Alamitos, IEEE, pp. 1–2. DOI: 10.1109/ICCMC56507.2023.10084264.
22. Sousa A. O., Veloso D. T., Gonçalves H. M. et al. Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects, *IEEE Access,* 2023, Vol. 11, pp. 89933–89946. DOI: 10.1109/ACCESS.2023.3307310
23. Nassif A. B., AbuTalib M., Capretz L. F. Software effort estimation from use case diagrams using nonlinear regression analysis, *Electrical and Computer Engineering : IEEE Canadian Conference CCECE'20.* London, ON, Canada, 30 Aug.–2 Sept., 2020 : proceedings,IEEE, 2020, pp. 1–4. DOI: 10.1109/CCECE47787.2020.9255712.
24. Prykhodko S., Prykhodko N., Knyrik K. Estimating the efforts of mobile application development in the planning phase using nonlinear regression analysis, *Applied Computer Systems,* 2020, Vol. 25, No. 2, pp. 172–179. DOI: 10.2478/acss-2020-0019
25. [Nhung H. L. T. K., Hai V. V. , Silhavy R. et al. Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression, *IEEE Access*, 2022. Vol. 10, pp. 2963–2986. DOI: 10.1109/ACCESS.2021.3139183
26. Sahoo P., Behera D. K., Mohanty J. R. et al. Effort Estimation of Software products by using UML Sequence models with Regression Analysis, *Proceedings of the 2022 OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 14–16 December 2022 : proceedings.* Los Alamitos, IEEE, pp. 97–101, DOI: 10.1109/OCIT56763.2022.00028.
27. Cibir E., Ayyildiz T. E. An Empirical Study on Software Test Effort Estimation for Defense Projects, *IEEE Access,* 2022, Vol. 10, pp. 48082–48087. DOI: 10.1109/ACCESS.2022.3172326
28. Yuan X., Su J., Yu C. and Ye S. Power Grid Software Cost Estimation Based on Improved COCOMO Model, *Proceeding of the 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 26–28 May 2023 : proceedings.* Los Alamitos, IEEE, pp. 1265–1269. DOI: 10.1109/ICETCI57876.2023.10176686
29. Prykhodko S., Prykhodko N., Makarova L. et al. Outlier Detection in Non-Linear Regression Analysis Based on the

DOI 10.15588/1607-3274-2024-1-8

Normalizing Transformations, *Proceedings of the 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), IEEE, Lviv-Slavske, 25–29 February 2020 : proceeding.* Los Alamitos, IEEE, pp. 407–410. DOI: 10.1109/TCSET49122.2020.235464
30. CodeMR [Electronic resource]. Access mode: https://plugins.jetbrains.com/plugin/10811-codemr
31. Mardia K. V. Measures of multivariate skewness and kurtosis with applications, *Biometrika,* 1970, Vol. 57, pp. 519–530. DOI: 10.1093/biomet/57.3.519
32. Frost J. Overfitting Regression Models: Problems, Detection, and Avoidance [Electronic resource]. Access mode: https://statisticsbyjim.com/regression/overfitting-regression-models/

Received 05.01.2024.
Accepted 16.02.2024.

УДК 004.412:519.237.5

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.

АНОТАЦІЯ

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. Предметом дослідження є нелінійні регресійні 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 чотирьохвимірного нормалізуючого перетворення Бокса-Кокса 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 Бокса-Кокса 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 змешненя середньої величини відносної похибки, зменшення ширини довірчих інтрвалів та інтервалів прогнозування порівняно з моделями, що використовують однофакторні нормалізуючі перетворення.

Результати. Проведено порівняння побудованої моделі з моделями нелінійної регресії з використанням десяткового логарифму та одновимірного перетворення Бокса-Кокса.

Висновки. Модель нелінійної регресії з трьома предикторами для ранньої оцінки метрики LOC застосунків із відкритим вихідним кодом на Kotlin побудовано на основі перетворення чотирьох змінних Бокса-Кокса. Порівняно з іншими моделями нелінійної регресії, ця модель демонструє більший множинний коефіцієнт детермінації, менше значення середньої величини відносної похибки та менші ширини довірчих інтервалів та інтервалів прогнозування. Перспективи подальших досліджень можуть включати застосування інших багатовимірних нормалізуючих перетворень і наборів даних для побудови моделі нелінійної регресії для ранньої оцінки метрики LOC застосунків із відкритим вихідним кодом на Kotlin для інших обмежень на предиктори.

КЛЮЧОВІ СЛОВА: оцінка, рядки коду, застосунок з відкритим вихідним кодом, Kotlin, нелінійна регресійна модель, перетворення Бокса-Кокса, клас, зважені методи на клас, глибина дерева успадкування.

ЛІТЕРАТУРА

1. Ponnala R. Object Oriented Dynamic Metrics in Software Development: A Literature Review / R. Ponnala, C. R. K. Reddy // International Journal of Applied Engineering Research. – 2019. – Vol. 14, No. 22. – P. 4161–4172.
2. Software cost estimation with COCOMO II / [B. W. Boehm, C. Abts, A. W. Brown et al.]. – Upper Saddle River, NJ: Prentice Hall PTR, 2000. – 506 p.
3. Rumiński B. Top Apps Built with Kotlin Multiplatform [2023 Update] [Electronic resource] / B. Rumiński. – Access mode: https://www.netguru.com/blog/top-apps-built-with-kotlin-multiplatform
4. Kaczmarek J. Size and effort estimation for applications written in Java / J. Kaczmarek, M. Kucharski // Information and Software Technology. – 2004. – Vol. 46, Issue 9. – P. 589–601. DOI: 10.1016/j.infsof.2003.11.001
5. Laird L. M. Software measurement and estimation. A practical approach. quantitative software engineering series / L. M. Laird, M. C. Brennan. – Wiley-IEEE Computer Society Press, 2006. – 379 p.
6. Tan H. B. K. Conceptual data model-based software size estimation for information systems / H. B. K. Tan, Y. Zhao, H. Zhang // Transactions on Software Engineering and Methodology. – 2009. – Vol. 19, Issue 2. – P. 1–37. DOI: 10.1145/1571629.1571630
7. Zifen Y. An improved software size estimation method based on object-oriented approach / Y. Zifen // Electrical & Electronics Engineering : IEEE Symposium EEESYM’12, Kuala Lumpur, Malaysia, 24–27 June 2012 : proceedings. – Los Alamitos:

DOI 10.15588/1607-3274-2024-1-8

IEEE, 2012. – P. 615–617. DOI: 10.1109/EEESym.2012.6258733.

8. Kiewkanya M. Constructing C++ software size estimation model from class diagram / M. Kiewkanya, S. Surak // Computer Science and Software Engineering : 13th International Joint Conference, Khon Kaen, Thailand, 13–15 July 2016 : proceedings. – Los Alamitos: IEEE, 2016. – P. 1–6. DOI: 10.1109/JCSSE.2016.7748880
9. Sholiq S. A comparative study of software development size estimation method: UCPabc vs Function Points / S. Sholiq, R. S. Dewi, A. P. Subriadi // Procedia Computer Science, – 2017. – Vol. 124, – P. 470–477. DOI: 10.1016/j.procs.2017.12.179
10. Prykhodko N. V. The non-linear regression model to estimate the software size of open source Java-based systems / N. V. Prykhodko, S. B. Prykhodko // Radio Electronics, Computer Science, Control. – 2018. – No. 3 (46). – P. 158–166. DOI: 10.15588/1607-3274-2018-3-17
11. Приходько С.Б. Чотирьохфакторна нелінійна регресійна модель для оцінювання розміру Java-застосунків з відкритим кодом / С. Б. Приходько, Н. В. Приходько, Т. Г. Смикодуб // Науковий журнал «Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки». 2020 – Том 31 (70) № 2. – С. 157–162. DOI: https://doi.org/10.32838/2663-5941/2020.2-1/25
12. Prykhodko S.B. A nonlinear regression model to estimate the size of web apps created using the CakePHP framework / S. B. Prykhodko, I. S. Shutko, A. S. Prykhodko // Radio Electronics, Computer Science, Control. – 2021. – No. 4 (59). – P. 129–139. DOI: 10.15588/1607-3274-2021-4-12
13. Manisha Early size estimation using machine learning / Manisha, R. Rishi // Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 17–19 March 2021 : proceedings. – Los Alamitos: IEEE, 2021. – P. 757–762. DOI: 10.1109/INDIACom51348.2021.00135
14. Daud, M. Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (ADAFs) / M. Daud, A. A. Malik // IEEE Access. – 2021. – Vol. 9. – P. 81986–81999. DOI: 10.1109/ACCESS.2021.3085752
15. Efficiency improvement of function point-based software size estimation with deep learning model / [K. Zhang, X. Wang, J. Ren et al.] // Los Alamitos: IEEE. – 2021. – Vol. 9. – P. 107124–107136. DOI: 10.1109/ACCESS.2020.2998581
16. Ritu. Comparative Analysis of Machine Learning Techniques in Effort Estimation / Ritu, Y. Garg // Proceedings of the 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), Faridabad, India, 2022 : proceedings. – Los Alamitos: IEEE, 2022. – P. 401–405, DOI: 10.1109/COM-IT-CON54601.2022.9850592
17. Brar P. A Systematic Literature Review of Machine Learning Techniques for Software Effort Estimation Models / P. Brar, D. Nandal // Proceeding of the 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 08–09 July 2022 – Los Alamitos: IEEE, 2022. – P. 494–499, DOI: 10.1109/CCiCT56684.2022.00093
18. Software Effort Estimation using Machine learning Algorithm / [Y. Assefa, F. Berhanu, A. Tilahun et al.] // Proceeding of the 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 28–30 November 2022 : proceedings. – Los Alamitos : IEEE, 2022. – P. 163–168, DOI: 10.1109/ICT4DA56482.2022.9971209
19. Effective Software Effort Estimation Leveraging Machine Learning for Digital Transformation / [A. Jadhav, S. K. Shandilya, I. Izonin et al] // IEEE Access. – 2023. – Vol. 11. – P. 83523–83536. DOI: 10.1109/ACCESS.2023.3293432
20. A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques / [S. Kumar, M. Arora, Sakshi et al.] // Proceedings of the 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 21–23 September 2022 : proceedings. – Los Alamitos: IEEE. – P. 416–422. DOI: 10.1109/ICIRCA54612.2022.9985542
21. R. K. B. N. Software Effort Estimation using ANN (Back Propagation) / R. K. B. N., Y. Suresh // Proceedings of the 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 23–25 February 2023 : proceedings. – Los Alamitos : IEEE. – P. 1–2, DOI: 10.1109/ICCMC56507.2023.10084264.
22. Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects / [A. O. Sousa, D. T. Veloso, H. M. Gonçalves et al.] // IEEE Access. – 2023. – Vol. 11. – P. 89933–89946. DOI: 10.1109/ACCESS.2023.3307310
23. Nassif A.B. Software effort estimation from use case diagrams using nonlinear regression analysis / A. B. Nassif, M. AbuTalib, L. F. Capretz // Electrical and Computer Engineering : IEEE Canadian Conference CCECE'20, London, ON, Canada, 30 Aug.–2 Sept., 2020 : proceedings. – IEEE, 2020. – P. 1–4. DOI: 10.1109/CCECE47787.2020.9255712.
24. Prykhodko S. Estimating the efforts of mobile application development in the planning phase using nonlinear regression analysis / S. Prykhodko, N. Prykhodko, K. Knyrik // Applied Computer Systems. – 2020. – Vol. 25, No. 2. – P. 172–179. DOI: 10.2478/acss-2020-0019
25. Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression / [H. L. T. K. Nhung, V. V. Hai, R. Silhavy et al.] // IEEE Access. – 2022. – Vol. 10. – P. 2963–2986. DOI: 10.1109/ACCESS.2021.3139183
26. Effort Estimation of Software products by using UML Sequence models with Regression Analysis / [P. Sahoo, D. K. Behera, J. R. Mohanty et al.] // Proceedings of the 2022 OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 14–16 December 2022 : proceedings. – Los Alamitos : IEEE. – P. 97–101. DOI: 10.1109/OCIT56763.2022.00028.
27. Cibir E. An Empirical Study on Software Test Effort Estimation for Defense Projects / E. Cibir, T. E. Ayyildiz // IEEE Access. – 2022. – Vol. 10. – P. 48082–48087. DOI: 10.1109/ACCESS.2022.3172326
28. Power Grid Software Cost Estimation Based on Improved COCOMO Model / [X. Yuan, J. Su, C. Yu and S. Ye] // Proceeding of the 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 26–28 May 2023 : proceedings. – Los Alamitos : IEEE. – P. 1265–1269, DOI: 10.1109/ICETCI57876.2023.10176686
29. Outlier Detection in Non-Linear Regression Analysis Based on the Normalizing Transformations / [S. Prykhodko, N. Prykhodko, L. Makarova et al.] // Proceedings of the 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), IEEE, Lviv-Slavske, 25–29 February 2020 : proceeding. – Los Alamitos : IEEE. – P. 407–410. DOI: 10.1109/TCSET49122.2020.235464
30. CodeMR [Electronic resource]. – Access mode: https://plugins.jetbrains.com/plugin/10811-codemr
31. Mardia K. V. Measures of multivariate skewness and kurtosis with applications / K. V. Mardia // Biometrika. – 1970. – Vol. 57. – P. 519–530. DOI: 10.1093/biomet/57.3.519
32. Frost J. Overfitting Regression Models: Problems, Detection, and Avoidance [Electronic resource]. – Access mode: https://statisticsbyjim.com/regression/overfitting-regression-models/

DOI 10.15588/1607-3274-2024-1-8