mathematics

MDPI

Article

An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort

Anca-Elena Iordan

Department of Computer Science, Technical University of Cluj-Napoca, 400027 Cluj-Napoca, Romania; anca.iordan@cs.utcluj.ro

Abstract: Software effort estimation has constituted a significant research theme in recent years. The more important provocation for project managers concerns reaching their targets within the fixed time boundary. Machine learning strategies can lead software management to an entire novel stage. The purpose of this research work is to compare an optimized long short-term memory neural network, based on particle swarm optimization, with six machine learning methods used to predict software development effort: K-nearest neighbours, decision tree, random forest, gradient boosted tree, multilayer perceptron, and long short-term memory. The process of effort estimation uses five datasets: China and Desharnais, for which outputs are expressed in person-hours; and Albrecht, Kemerer, and Cocomo81, for which outputs are measured in person-months. To compare the accuracy of these intelligent methods four metrics were used: mean absolute error, median absolute error, root mean square error, and coefficient of determination. For all five datasets, based on metric values, it was concluded that the proposed optimized long short-term memory intelligent method predicts more accurately the effort required to develop a software product. Python 3.8.12 programming language was used in conjunction with the TensorFlow 2.10.0, Keras 2.10.0, and SKlearn 1.0.1 to implement these machine learning methods.

Keywords: software effort estimation; K-nearest neighbours; decision tree; random forest; gradient boosted tree; multilayer perceptron; long short-term memory; particle swarm optimization

MSC: 68T07

Citation: Iordan, A.-E. An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort. *Mathematics* **2024**, *12*, 200. https://doi.org/10.3390/math12020200

Academic Editors: Dawei Cheng, Zhibin Niu and Yiyi Zhang

Received: 24 November 2023
Revised: 1 January 2024
Accepted: 3 January 2024
Published: 8 January 2024

1. Introduction

In the process of developing a software product, the stage of estimating the entire effort required to obtain the product according to the initial specifications, is a very complex task for the project manager. To facilitate the work of the project manager, multiple techniques specific to artificial intelligence [1] have been used to predict as accurately as possible the effort needed for software development. These techniques are rarely regarded as compelling for uncertainty administration, and the results show their improbable prediction abilities for effort estimation at underlying phases of the software lifecycle [2].

The concept of machine learning [3] is a subdomain of computer science that gives computers the ability to act without explicit programming. Machine learning [4] deals with the study and construction of algorithms that can learn certain patterns from a set of training data, then make predictions and make decisions with a completely new dataset as input. The provocation of this study is to reveal which of the six used methods—K-nearest neighbours, decision tree, random forest, gradient boosted tree, multilayer perceptron, and long short-term memory—is more efficient for the domain of software project management.

The best results obtained by the six used intelligent methods, implemented based on the parameter tuning process, are compared with the results of other studies. In order to obtain a more accurate estimate of the effort than the existing ones, the LSTM method is improved by using particle swarm optimization which aims to optimize the weights of the

LSTM neural network. The scientific contributions of this research work mainly consist of the following two aspects:

- An improved LSTM model based on particle swarm optimization is proposed, and its superiority is proved by comparing not only with the results obtained in this study, but also with the results obtained in the analysed scientific works.
- The optimized LSTM neural network using particle swarm optimization is innovatively used in software development effort estimation.

The direct results of this study based on the six methods previously specified can be used to simplify the tasks of project managers and increase the efficiency of the development process. To understand this research work as a whole, this article is structured as follows:

- In the introduction is presented the reason for choosing the theme.
- The second section presents a brief description of the current stage in the effort estimation process.
- The third section contains details about the used approach for software development effort estimation.
- The next section includes an analysis of the results provided by the six used intelligent methods, following the parameter tuning process and comparison with previous results.
- The fifth section presents the improved version of the LSTM method based on particle swarm optimization, highlighting the superior results it provides.
- The last section presents all the conclusions reached after the implementation of used methods.

2. Literature Survey

Over the years, software effort approximation has used approaches based on fuzzy logic, evolutionary methods, and artificial neural networks.

In paper [5], two machine learning methods were used (linear regression, K-nearest neighbours) and three versions of the Cocomo dataset. To determine which of the two analysed methods is better, the following five metrics were considered: root mean square error, relative absolute error, mean absolute error, and correlation coefficient. The model proposed in the aforementioned work consisted in identifying the problem domain, scanning data, and partition data in testing and training, using the WEKA tool. The reached outputs unveil that the linear regression method is a superior estimator by contraposition with KNN.

Another version for approximation of software effort is explored in [6] using the Case-Based Reasoning method optimized by the Genetic Algorithm on seven datasets: Albrecht, Maxwell, NASA, Telecom, Kemerer, China, and Desharnais (Table 1). The main goals of the authors were to investigate the combination of the GA algorithm with CBR to find the best combination of CBR parameters in order to improve the accuracy of software effort prediction. The research methodology consisted of processing the dataset, splitting the processed data for training and testing, and computing the CBR-GA model. Based on the values obtained for the three metrics used—mean absolute error, mean balanced relative error, and mean inverted balanced error—the proposed model provides more accurate estimates, especially in the case of larger datasets.

A gradient boosting regressor model proposed in paper [7] was applied on two datasets: Cocomo81 and China. The performance of the proposed model was reported to seven models: stochastic gradient descent, K-nearest neighbours, decision tree, bagging regressor, random forest regressor, Ada-boost regressor, and gradient boosting regressor starting from four metrics: mean absolute error, mean square error, root mean square error, and coefficient of determination. The gradient boosting algorithm is improved by adding the summation of the predicted results of the previous tree, and this iteration continues until the estimated accuracy is achieved. The research procedure consisted of data collection, data preprocessing, splitting the data for training and testing in a ratio of 80:20, and implementing the gradient boosting model. The study results proved that the

gradient boosting regressor performance is outstanding regarding the two datasets for all used metrics, obtaining values such as 0.98 (Cocomo81) and 0.93 (China) for coefficient of determination and 153 (Cocomo81) and 676.6 (China) for mean absolute error.

In study [8], three methods—linear regression, random forest, and multilayer perceptron—were used to obtain an accurate estimation of software effort. These methods were used on the Desharnais dataset, and the implementation was achieved through the WEKA toolkit. The research methodology steps cover a preprocessing technique used to eliminate irrelevant and excessive attributes, splitting the data for training and testing and for developing the three chosen models. The conclusion obtained by comparing these three methods was that linear regression determines a more accurate estimate than the other two methods based on five metrics: mean absolute error, root mean square error, root relative absolute error, relative squared error, and correlation coefficient.

In research presented in paper [9], more variants of the Cocomo dataset were used with a different number of selected attributes to estimate effort based on four machine learning algorithms: linear regression, support vector machine, regression tree, and random forest. The experiment procedure phases include data collection, data preprocessing, data analysis, data splitting, and prediction models development. The effects of the experiments revealed that the support vector machine and random forest methods categorically provide consistent results in the case of using only five important selected attributes compared to the case of using all the attributes.

A comparison between the multiple linear regression method and the expert judgement method applied on a real-time dataset obtained from a medium-sized multinational organization was made in [10]. Multiple linear regression generates better results based on the values of the used evaluation metrics.

In paper [11], the random forest method was compared with the regression tree method based on three datasets: ISBSG R8, Tukutuku, and Cocomo. The improved random forest method analyses the dependency by the number of used attributes from a dataset, observing whether the accuracy was sensitive of considered parameters. The evaluation metrics—magnitude of relative error, mean magnitude of relative error, and median magnitude of relative error—show that the random forest technique surpasses the regression tree technique.

In study [12], decision tree methods were used to estimate development effort and the cost of software when agile principles are fulfilled. The training process was based on 10-fold cross-validation, and the combining of three learning methods (decision tree, random forest and Ada-boost regressor) led to the improvement of prediction accuracy.

Another research [13] proposed the Neural Coherent Clustered Ensemble Classifier model combined with the Optimized Satin Bowerbird model applied to seven datasets: Cocomo81, CocomoNasa93, CocomoNASA60, Desharnais, AlbrechtFPA, ChinaFPA, and Cocomo_sdr (Table 1). The Neural Gas Coherent Clustering method was used with the scope to group the datasets in a coherent approach using an index of the nearest characteristic vector as the definitive parameter. The ECOPB proposed model was evaluated using two metrics: clustering accuracy and mean magnitude of relative error.

An improved analogy-based effort estimation method was introduced in [14] through the standard deviation technique. Validation of achieving improvement was based on a magnitude of relative error metric applied to a dataset, which included information about 21 projects developed using Scrum-based agile software development collected from six different software houses in Pakistan.

In the software development domain, the most accurate approximation of the software effort represents one of the most controvertible problems. The accurate approximation of the software effort needed for software project development is fundamental for performing software administration. For this consideration, correct approximation of software effort is a difficult research labour.

Table 1. Software effort estimation—related works synthesis.

Existing Works	Datasets	Methods	Metrics
Marapelli [5]	Cocomo	Linear regression K-nearest neighbours	Root mean square error Relative absolute error Mean absolute error Correlation coefficient
Hameed et al. [6]	Albrecht Maxwell NASA Telecom Kemerer China Desharnais	Genetic algorithm	Mean absolute error Mean balanced relative error Mean inverted balanced error
Kumar et al. [7]	Cocomo81 China	Stochastic gradient descent K-nearest neighbours Decision tree Bagging regressor Random forest regressor Ada-boost regressor Gradient boosting regressor	Mean absolute error Mean square error Root mean square error Coefficient of determination
Singh et al. [8]	Desharnais	Linear regression Random forest Multilayer perceptron	Mean absolute error Root mean square error Root relative absolute error Relative squared error Correlation coefficient
Zakaria et al. [9]	Cocomo	Linear regression Support vector machine Regression tree Random forest	Mean square error Root mean square error Mean absolute error Mean absolute percentage error
Abdelali et al. [11]	ISBSG R8 Tukutuku Cocomo	Random forest Regression tree	Magnitude of relative error Mean magnitude of relative error Median magnitude of relative error
Sanchez et al. [12]	Projects set from Pakistan	Decision tree Random forest Ada-boost regressor	Mean square error Mean relative error Coefficient of determination Mean magnitude of relative error
Resmi et al. [13]	Cocomo81 CocomoNasa93 CocomoNASA60 Desharnais AlbrechtFPA ChinaFPA Cocomo_sdr	ECOPB	Clustering accuracy Mean magnitude of relative error
Muhammad et al. [14]	21 projects from Pakistan	Analogy-based effort	Magnitude of relative error

3. Research Approach

3.1. Data Preparation

To establish the effort needed to develop a software product, there is a large number of data collections, such as: Albrecht, Cocomo81, China, Desharnais, ISBSG, Kemerer, Kitchenham, Maxwell, Miyazaki, NASA, and Tukutuku. In the procedure to approximate the software effort presented in this study, five datasets were used: Albrecht, Kemerer, Cocomo81, China, and Desharnais.

The number of attributes and the number of analysed projects for these five datasets are presented in Table 2. The Albrecht dataset [15] is characterized by eight attributes obtained from 24 IBM software projects, the Kemerer dataset [16] is characterized by seven attributes obtained from 15 analysed projects, and the Cocomo81 dataset [17] is characterized by 17 attributes obtained from 63 NASA software projects. The outputs of all these three datasets, representing the actual effort required to develop the software project, were in the unit person-months. The China dataset [18] was represented by 16 attributes

extracted from 499 analysed projects, and the Desharnais dataset [19] was represented by 12 attributes extracted from 81 software projects accomplished in Canada. The outputs of these last two mentioned datasets, representing the actual effort required to develop the software project, were in the unit person-hours.

Table 2. Datasets dimensions.

Datasets	Projects	Input Attributes	Output Attribute	Output Unit
Albrecht	24	7	1	Person-months
Kemerer	15	6	1	Person-months
Cocomo81	63	16	1	Person-months
China	499	15	1	Person-hours
Desharnais	81	11	1	Person-hours

From the collections of attributes associated with all datasets, six attributes were used; information about them is presented in Table 3. The selection of the used input attributes was performed arbitrarily, influencing the intelligent models implemented in this study. The second column in Table 3 contains the names of the used attributes, and the third column the meaning of each attribute. The last four columns of Table 3 contain information about each of the used attribute values as follows: the fifth column specifies the minimum value of the attribute, the sixth column specifies the maximum value of the attribute, the seventh column specifies the average value of the attribute, and the eighth column specifies the standard deviation of the attribute.

Table 3. Lists with chosen attributes for all five used datasets.

Datasets	Used Attributes	Attributes Description	Min	Max	Mean	Std
Albrecht	InputNumeric	Input functions number	7	193	40.25	36.913
	OutputNumeric	Output functions number	12	150	47.25	35.169
	FileNumeric	Count of file processing	3	60	17.37	15.522
	RawFPcouns	Raw function points	189.52	1902	638.54	452.654
	AdjfpNumeric	Adjusted function points	199	1902	647.62	487.995
	InquiryNumeric	Count of query functions	0	75	16.87	19.337
Kemerer	Language	Programming language	1	3	1.2	0.561
	RawFP	Unadjusted function points	97	2284	993.86	597.426
	Duration	Duration of the project	5	31	14.26	7.544
	AdjFP	Adjusted function points	99.9	2306.8	999.14	589.592
	Hardware	Hardware resources	1	6	2.33	1.676
	KSLOC	Kilo lines of code	39	450	186.57	136.817
Cocomo81	Data	Database size	0.94	1.16	1.04	0.073
	Time	Time constraint	1	1.66	1.11	0.161
	Cplx	Complexity of product	0.7	1.65	1.09	0.202
	Stor	Storage constraint	1	1.56	1.14	0.179
	Tool	Software tools use	0.83	1.24	1.01	0.085
	Loc	Lines of code	1.98	1150	77.21	168.51
China	Input	Function points of input	0	9404	167.12	486.301
	Output	Function points of external output	0	2455	113.61	221.299
	File	Function points of internal logical files	0	2955	91.23	210.289
	Added	Function points of added functions	0	13,580	360.41	829.797
	Resource	Team type	1	4	1.45	0.823
	AFP	Adjusted function points	9	17,518	486.91	1059.008
Desharnais	TeamExp	Team experience	−1	4	2.18	1.415
	ManagerExp	Manager experience	−1	7	2.53	1.643
	Length	Length of the project	1	39	11.67	7.424
	Entities	Number of entities	7	387	122.33	84.882
	Language	Programming language	1	3	1.55	0.707
	Adjustment	Adjusted factor	5	52	27.63	10.592

3.2. Used Metrics

Accurate assessment of the performance of artificial intelligence strategies [20] is very complicated due to unbalanced data collections. The following four metrics were used to achieve the previously stated scope: mean absolute error, root mean square error, median absolute error, and coefficient of determination.

Mean absolute error [21], denoted by MAE, signifies the average sum of absolute errors. The formula by which the mean absolute error is determined is established by Equation (1).

$$\mathrm{MAE} = \frac{1}{m} \cdot \sum_{k=1}^{m} \left| x_k - x_k'' \right| \tag{1}$$

In this formula (as well as in the following three), m signifies the number of all data points, x_k signifies the value to be estimated, and x_k'' is the estimated value. Median absolute error [22], denoted by MdAE, calculates the median of all absolute differences between the real effort and the estimated effort, defined by the formula represented in Equation (2).

$$\mathrm{MdAE} = \mathrm{median}\left(\left\{ \left| x_k - x_k'' \right| \right\}_{k=1}^{m} \right) \tag{2}$$

Root mean square error [23], denoted by RMSE, evaluates the standard deviation of the estimated value. The mathematical relation by which the root mean square error is evaluated is represented by Equation (3).

$$\mathrm{RMSE} = \sqrt{\frac{1}{m} \cdot \sum_{k=1}^{m} \left(x_k - x_k'' \right)^2} \tag{3}$$

Coefficient of determination [21], denoted by CD, is defined by dividing the sum of squared residual by the sum of all squares; the relation is given in Equation (4).

$$\mathrm{CD} = 1 - \frac{\sum_{k=1}^{m} \left(x_k - x_k'' \right)^2}{\sum_{k=1}^{m} \left(x_k - x_k' \right)^2} \tag{4}$$

where:

$$x' = \frac{1}{p} \cdot \sum_{k=1}^{m} x_k. \tag{5}$$

The range associated with the coefficient of determination is given by real numbers between 0 and 1. In the best situation, the predicted values fit exactly the real values, resulting in a value of 1 for coefficient of determination.

If the obtained value for the coefficient is negative, then a correlation does not exist between the data and the used model. These four selected metrics represent a crucial performance statistic in the case of regression models because are easy to understand, interpretable, and reliable elements for evaluation of the prediction accuracy. The lower the values associated with the first three metrics, the more accurately the model predicts. The determination of characteristic values of these four metrics was realized through the sklearn.metrics library belonging to the Scikit-learn [24] tool.

3.3. Selected Machine Learning Methods

For effort estimation, six machine learning methods were chosen: K-nearest neighbours (KNN), decision tree (DT), random forest (RF), gradient boosted tree (GBT), multilayer perceptron (MLP), and long short-term memory (LSTM).

The K-nearest neighbours method [25] is the easiest machine learning method, and the fact that it fetches the right results in a shorter time has led to its great use for both classification and regression problems. This method is based on a feature similarity, which represents that the similarity level of the values' features to those of the training set determines how a new input will be predicted. An output value will be predicted for the

new input based on the features of the input neighbours, with the predicted value being determined by the majority of its neighbours.

A decision tree [26] consists of internal nodes, branches, and leaf nodes, where each internal node represents the test for a certain attribute, each branch expresses the result of the test, and the leaf nodes represent the classes. The decision tree classifier includes two stages: building the tree, and applying it as a solution model to the addressed problem. To evaluate the utility of an attribute, the notion of information gain, defined in terms of entropy, is used. Thus, the greater the information gain, the lower the entropy.

The random forest [27] strategy consists of a collection of decision trees in which each tree is built by applying an algorithm and a random vector to the training dataset. The prediction given by this method is obtained by a majority of votes over the predictions given by each individual tree.

The gradient boosted tree [28] is one of the most powerful strategies for building predictive models, involving three elements: a loss function to be optimized, a weak learner to make the predictions, and an additive model to add weak learners to minimize the loss function. The loss function depends on the type of problem to be solved. Decision trees are used as weak learners in this strategy, being built in a greedy way in which the best branching points are chosen based on the purity of the scores.

Artificial neural networks are computational systems inspired by the way biological nervous systems process information. The main element of this paradigm is the structure of the information processing system, which is composed of large number of highly interconnected processing elements that work together to solve a certain task. The learning process of biological systems involves adjustments of the synaptic connections that exist between neurons, which also happens in the case of artificial networks that learn and adjust by example. From the numerous types of artificial neural networks, the multilayer perceptron and the long short-term memory neural networks were used in this study.

MLP includes more layers, every layer being connected to the following one. The layers' nodes are neurons characterized by activation functions, less so for the nodes from the input layer. Between the input and the output layers there are one or more hidden layers. Backpropagation is the learning strategy that allows multilayer perceptron to iteratively adapt the weights in the network layers in order to optimize the cost function.

LSTM [29] is a category of artificial recurrent neural network [30] used in the deep learning area. Compared with standard feed-forward neural networks, LSTM is characterized by feedback connections. Moreover, LSTM is very sensitive to the number of nodes in hidden layers, the number of training epochs, the initial learning rate, the momentum, and the dropout probability. These parameters will have a large impact on the software effort estimation performance.

The Python 3.8.12 programming language [31] together with four libraries—Keras 2.10.0 [32], Tensorflow 2.10.0, MathPlotLib [33], and SKlearn 1.0.1—were chosen to implement and evaluate these six selected machine learning methods.

3.4. Development of Effort Estimation Software

To successfully achieve the proposed objectives, an intelligent software was developed whose functionalities are included in the UML use case diagram [34] shown in Figure 1. The use case diagram contains an actor (the user of the intelligent software), eleven use cases, and relationships between them.

Thereby, the software functionalities consist of:

- Selecting a dataset from the five analysed sets used in model evaluation;
- Parameter tuning, training, testing, and evaluating an intelligent method selected from the six intelligent methods used in this study;
- Improving the LSTM method through particle swarm optimization;
- Parameter tuning, training, testing, and evaluation of the optimized LSTM method;
- Comparing the performance of the improved LSTM method with the results generated by the six intelligent chosen methods.

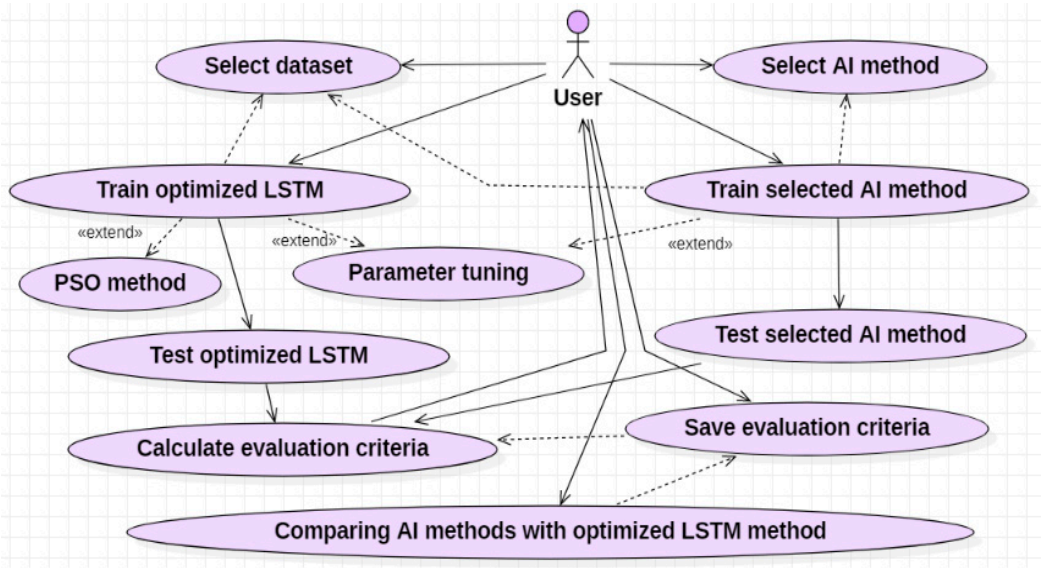

Figure 1. UML use case diagram.

4. Analysis of the Six Selected Classical Intelligent Methods

In most methods of machine learning [35], the parameters represent variables used by methods to learn the data characteristics and to adjust the learning from the dataset, with the purpose of obtaining the best performance. The parameter tuning [36] concept involves obtaining the suitable parameters for every learning method, such that the predicted results are optimal.

After the parameter tuning process, it is necessary to split the datasets used for training and testing. For every used dataset, in order to detect the suitable percentage to train and test the data, a fitting strategy was used and a value of approximately 80% was chosen for the training set and the remainder for the test set. Table 4 shows information about the number of effort values (columns 2 and 7), the minimum effort value (columns 3 and 8), the maximum effort value (columns 4 and 9), the average effort value (columns 5 and 10) and the standard deviation (columns 6 and 11) used both in the training stage of the intelligent methods and in their testing stage.

Table 4. Real effort values.

Datasets	Training					Testing				
	Number	Minim	Maxim	Mean	Std	Number	Minim	Maxim	Mean	Std
Albrecht	18	0.5	105.2	18.41	25.609	6	2.9	102.4	32.26	36.225
Kemerer	11	23.2	1107.31	222.91	306.831	4	72.0	287.0	209.15	94.446
Cocomo81	47	5.9	11,400.0	819.91	2075.392	16	6.0	2040.0	282.06	525.316
China	374	26.0	54,620.0	4218.06	6679.07	125	89.0	49,034.0	3417.62	5826.511
Desharnais	60	651.0	23,940.0	5241.55	4714.124	21	546.0	14,987.0	4488.47	3478.961

4.1. K-Nearest Neighbours

The K-nearest neighbours method, having the six inputs presented in Table 3 and an output (estimated effort), was set with the following features: Minkowski metric for distance computation, leaf size equal with 30, and uniform weights. To implement this method, the KNeighborsRegressor function from the SKlearn 1.0.1 library was used. For the K-nearest neighbours method, in the parameter tuning process, the following two parameters were used to determine the performance of this method:

- k—signifies the number of neighbours used to determine the prediction for new instances. In this study, the eight used values of this parameter vary between 3 and 10.

- p—signifies the power parameter from the Minkowski distance [37], characterized by the following formula:

$$d_{Minkowski} = \left(\sum_{k=1}^{m}\left(\left|x_k - x_k''\right|\right)^p\right)^{\frac{1}{p}} \quad (6)$$

The usefulness of the Minkowski distance within the KNN method consists in determining which neighbours will be analysed to compare their characteristics with those of a new instance for which a new prediction is determined. Thereby, distance metrics are used calculate which are the neighbours with the most appropriate features and choose the first K neighbours to obtain the new prediction. For this second parameter, three values between 1 and 3 were used by the KNN method to predict the effort. If the parameter p is equal to the value 1, the Minkowski distance is reduced to the Manhattan distance [38] given by the following formula:

$$d_{Manhattan} = \sum_{k=1}^{m}\left|x_k - x_k''\right| \quad (7)$$

In the case when the parameter p is equal with value 2, the Minkowski distance is transformed into the Euclidean distance [39], represented by the next formula:

$$d_{Euclidean} = \sqrt{\sum_{k=1}^{m}\left(x_k - x_k''\right)^2} \quad (8)$$

Following the used values in the parameter tuning process (eight values for parameter k and three values for parameter p), 24 variants of the KNN method were trained and tested. In Table 5, columns 3 to 5 show the values obtained for the four used metrics by these 24 variants of the KNN method: the third column shows the minimum value, the fourth column shows the maximum value, and the fifth column shows the average value for each metric. Columns 6 and 7 present the values of the parameters for which the minimum values of the four metrics were obtained.

Table 5. Test results for KNN method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	p	k	Minim	Maxim	Mean
MAE	Albrecht	11.228	18.557	15.914	1	3	6.333	59.467	22.872
	Kemerer	75.519	99.728	87.201	1	4	86.675	210.325	140.968
	Cocomo81	221.619	257.051	235.505	2	3	14.001	476.333	170.152
	China	1747.048	2154.672	1983.695	1	3	272.041	24,896.841	12,753.127
	Desharnais	1422.878	1660.392	1562.577	1	7	1831.011	8977.015	4149.707
MdAE	Albrecht	4.556	6.657	5.671	2	8	6.251	30.187	16.237
	Kemerer	57.933	105.039	89.929	1	3	63.233	237.001	130.167
	Cocomo81	35.783	138.357	90.451	1	3	13.998	477.336	169.568
	China	611.602	1228.203	963.552	1	5	211.713	21,100.605	11,009.203
	Desharnais	921.667	1541.417	1270.797	1	3	1547.02	10,061.333	3580.746
RMSE	Albrecht	18.202	31.746	27.054	1	3	6.333	59.467	22.872
	Kemerer	90.933	109.744	101.412	1	4	86.671	210.325	140.968
	Cocomo81	446.987	483.493	466.399	1	7	17.742	448.714	203.692
	China	3922.755	4436.163	4161.753	1	7	237.135	22,966.109	10,303.723
	Desharnais	1847.561	2839.927	2230.060	1	6	1886.501	9945.833	4193.794
CD	Albrecht	0.389	0.437	0.416	1	3	6.333	59.467	22.872
	Kemerer	0.303	0.351	0.332	1	4	86.675	210.325	140.968
	Cocomo81	0.738	0.793	0.762	2	3	14.001	476.333	170.152
	China	0.668	0.762	0.713	1	3	272.041	24,896.841	12,753.127
	Desharnais	0.591	0.658	0.624	1	7	1831.011	8977.015	4149.707

The last three columns show information about the estimated effort by the KNN model for which optimum values of the metrics were obtained. For all five datasets, by comparing

the real effort values from Table 4 with the estimated values from Table 5, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort.

4.2. Decision Tree

Characterized by the six inputs presented in Table 3 and an output (estimated effort), the DT method was designed as follows: the minimum number of samples required to split an internal node is equal with 2 and the strategy to split each node is the best split. This method was implemented through the DecisionTreeRegressor function from the SKlearn 1.0.1 library. To determine the performance of the Decision Tree [40] method applied to the five datasets, in the parameter tuning process, the following two parameters were tuned:

- d—represents the maximum depth of the tree. In this research study, six values were used for this parameter varying between 5 and 10.
- n—represents the maximum leaf nodes. In this research study, 10 values were used for this parameter varying between 11 and 20.

Following the used values in the parameter tuning process (six values for parameter d and 10 values for parameter n), 60 variants of the DT method were trained and tested. Table 6 shows the values provided by the 60 variants of the DT method, the meanings of the columns from Table 6 being the same as those from Table 5. For all five datasets, by comparing the real effort values from Table 3 with the estimated values from Table 6, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort.

Table 6. Test results for DT method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	d	n	Minim	Maxim	Mean
MAE	Albrecht	6.975	23.439	12.378	5	12	7.751	101.202	68.357
	Kemerer	95.853	123.987	105.829	6	12	73.202	159.006	124.358
	Cocomo81	275.979	305.639	287.251	6	17	24.016	1069.077	453.055
	China	1575.633	2556.485	2098.901	5	16	895.606	33,291.007	18,912.274
	Desharnais	1953.361	2693.082	2087.125	7	17	2593.501	9100.246	4507.015
MdAE	Albrecht	3.825	9.868	6.738	5	12	7.751	101.202	68.357
	Kemerer	81.802	123.525	99.104	5	18	81.121	162.015	132.831
	Cocomo81	48.617	154.181	77.217	6	17	24.016	1069.077	453.055
	China	755.642	1845.505	1398.297	6	19	895.603	33,291.075	17,588.015
	Desharnais	1453.083	2108.705	1701.025	6	11	1414.447	14,973.075	7982.055
RMSE	Albrecht	10.213	32.872	17.083	5	12	7.751	101.202	68.357
	Kemerer	106.226	137.477	119.253	6	12	73.202	159.006	124.358
	Cocomo81	514.792	585.492	541.035	8	18	98.764	902.033	345.159
	China	2848.557	4441.981	3471.155	5	16	895.606	33,291.007	18,912.274
	Desharnais	2526.071	3704.809	3306.172	5	18	2507.501	9217.258	6988.345
CD	Albrecht	0.497	0.561	0.536	5	12	7.751	101.202	68.357
	Kemerer	0.275	0.309	0.289	6	12	73.202	159.006	124.358
	Cocomo81	0.691	0.774	0.733	6	17	24.016	1069.077	453.055
	China	0.685	0.793	0.723	5	16	895.606	33,291.007	18,912.274
	Desharnais	0.479	0.563	0.515	7	17	2593.501	9100.246	4507.015

4.3. Random Forest

Random forest [41] is an estimator that fits a number of decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. At the implementation of RF method, the RandomForestRegressor function from the SKlearn 1.0.1 library was used. The RF method, having the six inputs presented in Table 3 and an output (estimated effort), was set with the following features: squared error as a function to measure the quality of a split, and the value of 2 for the minimum number of samples

required to split an internal node and also for the minimum number of samples required to be at a leaf node.

To determine the performance of the random forest method applied to the five datasets, in the parameter tuning process, the following two parameters were tuned:

- d—represents the maximum depth of the tree. In this research study, six values were used for this parameter varying between 5 and 10.
- t—represents the number of trees in the forest. In this research study, six values were used for this parameter belonging to the following set: {50, 100, 150, 200, 250, 300}.

Following the used values in the parameter tuning process (six values for parameter d and six values for parameter t), 36 variants of the RF method were trained and tested. Table 7 shows the values provided by the 36 variants of the RF method, the meanings of the columns from Table 7 being the same as those from Table 5.

Table 7. Test results for RF method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	d	t	Minim	Maxim	Mean
MAE	Albrecht	6.015	20.914	10.817	7	150	9.374	98.207	67.264
	Kemerer	82.739	135.792	110.145	9	200	77.025	204.876	132.329
	Cocomo81	235.857	299.066	267.743	8	250	33.099	1329.045	567.987
	China	1409.324	2441.075	1970.845	6	300	706.125	35,987.755	20,075.177
	Desharnais	1792.837	2479.175	1995.075	8	200	2275.045	11,975.357	5705.075
MdAE	Albrecht	3.579	8.528	4.381	8	100	12.842	86.925	46.152
	Kemerer	69.174	117.026	98.713	10	150	82.038	236.109	157.235
	Cocomo81	43.925	138.271	69.925	9	100	78.808	1099.125	458.794
	China	621.782	1753.655	1203.555	10	250	819.255	36,606.177	19,877.095
	Desharnais	1284.947	2095.175	1685.255	9	300	1155.038	13,475.105	6795.755
RMSE	Albrecht	7.726	30.714	24.003	7	150	9.374	98.207	67.264
	Kemerer	83.726	128.057	102.113	9	200	77.025	204.876	132.329
	Cocomo81	487.283	557.883	521.151	9	100	78.808	1099.125	458.794
	China	2792.372	4337.077	3507.808	6	300	706.125	35,987.755	20,075.177
	Desharnais	2379.063	3685.915	2985.175	9	300	1155.038	13,475.105	6795.755
CD	Albrecht	0.519	0.593	0.553	7	150	9.374	98.207	67.264
	Kemerer	0.284	0.327	0.301	9	200	77.025	204.876	132.329
	Cocomo81	0.658	0.732	0.693	8	250	33.099	1329.045	567.987
	China	0.709	0.821	0.765	6	300	706.125	35,987.755	20,075.177
	Desharnais	0.491	0.581	0.543	8	200	2275.045	11,975.357	5705.075

For all five datasets, by comparing the real effort values from Table 4 with the estimated values from Table 7, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort.

4.4. Gradient Boosted Tree

The gradient boosted tree [42] estimator makes an additive model in a forward stepwise manner, allowing the optimization of arbitrary differentiable loss functions. In each step, a regression tree is fitted on the negative gradient of the given loss function. For the implementation of the GBT method, the GradientBoostingRegressor function from the SKlearn 1.0.1 library was used. The GBT method, characterized by the six inputs presented in Table 3 and an output (estimated effort), has been designed with the following features: the squared error to optimize the loss function, the friedman_mse function to measure the quality of a split, the value of 3 as the minimum number of samples required to split an internal node, and the value of 200 for the number of boosting stages to perform.

To determine the performance of the gradient boosted tree method applied to the five datasets, in the parameter tuning process, the following two parameters were tuned:

- d—represents the maximum depth of the individual regression estimators. In this research study, five values were used for this parameter varying between 1 and 5.
- l—represents the learning rate which shrinks the contribution of each tree. In this research study, five values were used for this parameter belonging to the following set: {0.05, 0.1, 0.15, 0.2, 0.25}.

Following the used values in the parameter tuning process (five values for parameter d and five values for parameter l), 25 variants of the GBT method were trained and tested. Table 8 shows the values provided by the 25 variants of the GBT method, the meanings of the columns from Table 8 being the same as those from Table 5. For all five datasets, by comparing the real effort values from Table 3 with the estimated values from Table 8, it is observed that all estimated intervals are included in the real intervals.

Table 8. Test results for GBT method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	d	l	Minim	Maxim	Mean
MAE	Albrecht	5.892	27.674	18.936	5	0.15	8.294	89.491	69.174
	Kemerer	60.197	120.345	82.392	4	0.25	74.147	223.736	148.884
	Cocomo81	209.198	279.885	251.092	5	0.2	31.725	1492.221	671.257
	China	1287.382	2835.135	1487.655	4	0.25	95.125	39,875.755	15,978.045
	Desharnais	1549.792	2278.755	1956.705	5	0.15	908.755	11,309.875	5995.255
MdAE	Albrecht	3.092	17.538	11.941	4	0.2	12.743	78.926	54.729
	Kemerer	58.682	106.827	80.724	3	0.15	89.387	213.357	176.086
	Cocomo81	39.826	127.864	81.023	3	0.25	81.216	1283.117	687.832
	China	504.672	2095.185	1675.085	2	0.15	709.148	38,075.275	17,899.345
	Desharnais	901.925	2399.075	1108.125	3	0.2	1093.445	8796.095	5750.174
RMSE	Albrecht	8.094	38.757	23.931	5	0.25	8.839	88.937	68.147
	Kemerer	64.045	134.981	98.927	4	0.25	74.147	223.736	148.884
	Cocomo81	463.782	521.236	482.226	3	0.25	81.216	1283.117	687.832
	China	2563.824	6235.975	4387.187	4	0.25	95.125	39,875.755	15,978.045
	Desharnais	2095.827	3190.725	2601.007	3	0.2	1093.445	8796.095	5750.174
CD	Albrecht	0.527	0.609	0.563	5	0.15	8.294	89.491	69.174
	Kemerer	0.347	0.392	0.367	4	0.25	74.147	223.736	148.884
	Cocomo81	0.759	0.812	0.774	5	0.2	31.725	1492.221	671.257
	China	0.773	0.865	0.815	4	0.25	95.125	39,875.755	15,978.045
	Desharnais	0.585	0.639	0.607	5	0.15	908.755	11,309.875	5995.255

4.5. Multilayer Perceptron

At the implementation of MLP method, the MLPRegressor function from the SKlearn 1.0.1 library was used. The MLP method, characterized by the six inputs presented in Table 3 and an output (estimated effort), has been designed with the following features: three hidden layers each with 100 neurons, the relu activation function for the hidden layers, and an Adam solver for weight optimization. To determine the performance of the multilayer perceptron [43] method applied to the five datasets, in the parameter tuning process, the following two parameters were tuned:

- t—represents the maximum number of iterations. The values of this parameter are represented by the elements of the set {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}.
- l—represents the initial learning rate. In this research study, the values of this parameter are represented by the elements of the set {0.002, 0.003, 0.004, 0.005, 0.006}.

Following the used values in the parameter tuning process (10 values for parameter t and five values for parameter l), 50 variants of the MLP method were trained and tested. Table 9 shows the values provided by the 50 variants of the MLP method, the meanings of the columns from Table 9 being the same as those from Table 5.

Table 9. Test results for MLP method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	t	l	Minim	Maxim	Mean
MAE	Albrecht	7.555	32.306	17.046	800	0.003	3.726	92.538	56.623
	Kemerer	48.145	523.383	275.035	400	0.003	85.215	239.455	135.764
	Cocomo81	199.513	224.522	211.092	400	0.004	39.785	945.767	435.843
	China	1383.713	2764.524	1309.175	500	0.004	100.262	36,214.905	17,088.325
	Desharnais	1587.619	2434.246	1997.077	200	0.004	1189.909	8186.266	4378.075
MdAE	Albrecht	3.256	25.655	12.104	400	0.003	2.598	41.759	28.729
	Kemerer	47.427	582.999	321.428	400	0.003	85.215	239.455	135.764
	Cocomo81	57.833	98.139	69.924	1000	0.003	48.505	957.248	675.912
	China	517.898	1929.029	1417.075	500	0.004	100.262	36,214.905	17,088.325
	Desharnais	905.841	2521.222	1203.755	100	0.006	1114.505	7368.811	4877.075
RMSE	Albrecht	8.208	42.338	24.753	800	0.003	3.726	92.538	56.623
	Kemerer	58.217	570.662	298.519	400	0.003	85.215	239.455	135.764
	Cocomo81	355.245	371.664	362.214	500	0.002	60.725	964.043	542.912
	China	2892.196	5864.361	3708.185	500	0.004	100.262	36,214.905	17,088.325
	Desharnais	2176.591	3252.083	2709.122	200	0.004	1189.909	8186.266	4378.075
CD	Albrecht	0.503	0.547	0.522	800	0.003	3.726	92.538	56.623
	Kemerer	0.382	0.438	0.413	400	0.003	85.215	239.455	135.764
	Cocomo81	0.762	0.826	0.793	400	0.004	39.785	945.767	435.843
	China	0.792	0.843	0.825	500	0.004	100.262	36,214.905	17,088.325
	Desharnais	0.576	0.624	0.599	200	0.004	1189.909	8186.266	4378.075

For all five datasets, by comparing the real effort values from Table 4 with the estimated values from Table 9, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort.

4.6. Long Short-Term Memory

In addition to recurrent neural networks, to which category it belongs, long short-term memory [44] adds a gate structure to its architecture. Compared to traditional neural networks, which are characterized by only one input, LSTM has two input sets: the current information and the output vector provided by the previous unit, while the complex processes associated with the LSTM cell are performed by the unit state. The essence of LSTM lies in the hidden layer, which instead of having nodes contains blocks of memory that contain components which make them smarter than nodes, called memory cells, consisting of three separate gates to regulate the flow and modification of information. The LSTM unit state consists of a forget gate, an input gate, and an output gate. The purpose of the forget gate is to choose to retain or discard some information. The input gate has the role of determining which information is retained internally and of ensuring that critical information can be saved. The output gate's role is to ascertain the output value and to control the current LSTM state which must be pass to the enable function.

In order to implement the LSTM method, an instance of the LSTM class defined in Keras.layers was used. The LSTM method, characterized by the six inputs presented in Table 3 and an output (estimated effort), has been designed with the following features: hyperbolic tangent as activation function, sigmoid as recurrent activation function, and a value of 0.5 for dropout probability. To determine the performance of the LSTM method applied to five datasets using an Adam optimizer, in the parameter tuning process the following two parameters were tuned:

- e—represents the number of training epochs. The values of this parameter are represented by the elements of the set {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}.
- n—represents the number of neurons in the LSTM hidden layer. The values of this parameter belong to the set {25, 50, 75, 100}.

Following the used values in the parameter tuning process (10 values for parameter e and four values for parameter n), 40 variants of the LSTM method were trained and tested. Table 10 shows the values provided by the 40 variants of the LSTM method, the meanings of the columns from Table 10 being the same as those from Table 5. For all five datasets, by comparing the real effort values from Table 3 with the estimated values from Table 10, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort.

Table 10. Test results for LSTM method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	e	n	Minim	Maxim	Mean
MAE	Albrecht	4.870	7.428	5.341	700	50	7.218	102.357	62.851
	Kemerer	42.094	499.132	233.437	500	25	72.912	253.776	148.866
	Cocomo81	178.051	433.352	256.467	300	25	93.918	1845.402	1001.563
	China	865.122	2837.991	1899.123	1000	25	544.762	39,938.265	28,055.125
	Desharnais	1404.571	2297.282	1709.153	600	100	610.162	9234.885	3545.995
MdAE	Albrecht	2.911	6.995	4.235	500	100	6.286	100.391	59.973
	Kemerer	26.206	580.956	311.468	500	25	72.912	253.776	148.866
	Cocomo81	30.642	332.799	187.395	900	50	21.553	500.38	245.728
	China	272.353	2188.822	1550.135	900	50	224.834	26,239.266	15,066.025
	Desharnais	880.121	2130.503	1406.345	1000	75	657.802	7847.106	5008.755
RMSE	Albrecht	6.560	8.209	7.349	100	75	7.637	99.044	55.862
	Kemerer	57.560	544.419	279.828	500	25	72.912	253.776	148.866
	Cocomo81	245.153	574.658	352.714	300	25	93.918	1845.402	1001.563
	China	2034.574	4864.522	2897.095	1000	25	544.762	39,938.265	28,055.125
	Desharnais	1960.007	3178.601	2499.065	600	100	610.162	9234.885	3545.995
CD	Albrecht	0.594	0.631	0.612	700	50	7.218	102.357	62.851
	Kemerer	0.431	0.493	0.464	500	25	72.912	253.776	148.866
	Cocomo81	0.816	0.897	0.845	300	25	93.918	1845.402	1001.563
	China	0.784	0.902	0.845	1000	25	544.762	39,938.265	28,055.125
	Desharnais	0.608	0.662	0.631	600	100	610.162	9234.885	3545.995

4.7. Comparative Analysis with Previous Works

In the case of the Albrecht dataset, the minimum value obtained for the MAE metric is 4.870, the minimum value obtained for the MdAE metric is 2.911, and the minimum value obtained for the RMSE metric is 6.560, all these three values being obtained using the LSTM method.

Thus, this method provides the most efficient estimate for the Albrecht dataset among the six analysed. Comparing the results obtained for the Albrecht dataset with the value of 7.742 (Table 11) for the MAE metric presented in the paper [6], it is observed that the optimal variants of the DT, RF, RBT, MLP, and LSTM methods provide lower values, and therefore is a better estimate of the effort. Only in the case of the KNN method was a value greater than 7.742 obtained.

For the Kemerer dataset, the minimum value obtained for the MAE metric is 42.094, the minimum value obtained for the MdAE metric is 26.206, the minimum value obtained for the RMSE metric is 57.560, and the maximum value for the CD metric is 0.493, all these values also being obtained using the LSTM method. Therefore, the LSTM method provides the most efficient estimate for the Kemerer dataset among the six analysed methods. Comparing the results obtained for the Kemerer dataset with the value 138.911 (Table 11) for the MAE metric presented in paper [6], it is observed that the optimal variants of all six analysed methods provide lower values for the MAE metric, and is thus a better estimation of effort.

Table 11. Comparison with the results of other studies.

Datasets	Metrics	Metric Values	
		Other Studies	LSTM—This Study
Albrecht	MAE	7.742 [6]	4.870
Kemerer	MAE	138.911 [6]	42.094
Cocomo81	MAE	928.3318 [9] 255.2615 [5] 153 [7]	178.051
	RMSE	2278.87 [9] 533.4206 [5] 228.7 [7]	245.153
	CD	0.98 [7]	0.897
China	MAE	926.182 [6] 676.6 [7]	865.122
	RMSE	1803.3 [7]	2034.574
	CD	0.93 [7]	0.902
Desharnais	MAE	2244.675 [6] 2013.7987 [8]	1404.571
	RMSE	2824.57 [8]	1847.560

In the case of the Cocomo81 dataset, the minimum value obtained for the MAE metric is 178.051, the minimum value obtained for the MdAE metric is 30.642, the minimum value obtained for the RMSE metric is 245.153, and the maximum value for the CD metric is 0.897, all these values being obtained thanks to the LSTM method. Thus, this method provides the most efficient estimate for the Cocomo81 dataset among the six analysed methods. Comparing the results obtained for the Cocomo81 dataset with the value 928.3318 (Table 11) for the MAE metric and with the value 2278.87 for the RMSE metric presented in the paper [9], it is observed that the optimal variants of all six analysed methods provide lower values, and so it is a better estimate of the effort. Comparing the results obtained for Cocomo81 dataset with the value 255.2615 (Table 11) for the MAE metric presented in the paper [5], it is observed that the optimal variants of KNN, RF, GBT, MLP, and LSTM methods provide lower values, and is thus a better estimate of the effort. Only in the case of the DT method was a value greater than 255.2615 obtained. Comparing the results obtained for the Cocomo81 dataset with the value of 533.4206 for the RMSE metric presented in the paper [5], it is observed that the optimal variants of all the six analysed methods provide lower values, and is thus a better estimation of the effort. Comparing the minimum value 178.051 obtained for the MAE metric through the LSTM method with the value 153 (Table 11) presented in the paper [7], the minimum value 245.153 obtained for the RMSE metric through LSTM method with the value 228.7 presented in the paper [7], and the maximum value 0.897 obtained for the CD metric through the LSTM method with the value 0.98 presented in the same paper, it can be concluded that the model presented in [7] is more efficient than the LSTM method used in this research work.

For the China dataset, the minimum value obtained for the MAE metric is 865.122, the minimum value obtained for the MdAE metric is 272.353, the minimum value obtained for the RMSE metric is 2034.574, and the maximum value for the CD metric is 0.902, all these values being obtained using the LSTM method. Thus, this method provides the most efficient estimate for China dataset among the six analysed methods. Comparing the results obtained for the China dataset with the value 926.182 (Table 11) for the MAE metric presented in paper [6], it is observed that only the LSTM method provides lower values, and is thus a better estimate of the effort. In the case of the KNN, DT, RF, GBT, and MLP methods, values higher than 926.182 were obtained, and so they are more ineffective methods. Comparing the minimum value 865.122 obtained for the MAE metric by means of the LSTM method with the value 676.6 (Table 11) presented in the paper [7], the minimum value 2034.574 obtained for the RMSE metric by means of the LSTM method with the

value 1803.3 presented in paper [7], and the maximum value 0.902 obtained for the CD metric through the LSTM method with the value 0.93 presented in the same paper, it can be concluded that the model presented in [7] is more efficient than the LSTM method used in this research paper for the China dataset.

In the case of the Desharnais dataset, the minimum value obtained for the MAE metric is 1404.571, the minimum value obtained for the MdAE metric is 880.121, and the minimum value obtained for the RMSE metric is 1847.561, the first two values being obtained thanks to the LSTM method, and the last one through the KNN method. For the CD metric, the maximum value 0.662 was obtained using the LSTM method. Comparing the results obtained for the Desharnais dataset with the value 2244.675 (Table 11) for the MAE metric presented in the paper [6], it is observed that the optimal variants of all six analysed methods provide lower values, therefore it is a better estimate of the effort. Moreover, comparing the results obtained for the Desharnais dataset with the value 2013.79 for the MAE metric presented in the work [8] and with value 2824.57 for the RMSE metric presented in the same work, it is observed that the optimal variants of all six analysed methods provide lower values, so it is a better estimation of the effort.

As can be seen from the second paragraph, the research methodologies in the case of the five used works [5–9] in the comparison of the results are similar to the methodological process used in this study, but with different percentages used when dividing the data for training and testing. After comparing the results obtained with the values presented in the research works selected for comparison, it can be observed that for the Albrecht, Kemerer, and Desharnais datasets, the LSTM method provides better estimates of the effort. Because, for Cocomo81 and China datasets, the architecture of the LSTM method presented in this paper does not provide satisfactory results in comparison with the results obtained by the model presented in the paper [7], further research should be carried out to improve the LSTM method.

5. Optimized LSTM Based on Particle Swarm Optimization

Particle swarm optimization (PSO) [45] is an optimization method belonging to the computational intelligence field, being derived from the study of bird predation behaviour. Every particle in the PSO method fits to a possible solution of the problem, being characterized by three metrics: velocity, position, and fitness. PSO calculates the fitness value of the particles through a process of continuously updating the position and velocity during the iterative process to reach the global optimum. The PSO method is characterized by a fast search speed, easy convergence, and great efficiency. Starting from this aspect, the PSO method combined with LSTM has been applied in many fields, such as: stock forecast [46], smart agriculture [47], financial risk [48], and teaching quality [49].

In the case of the standard LSTM method, the values of some hyperparameters, such as the initial learning rate, the dropout probability, and the momentum, must be set manually. The choice of suitable values for these parameters is based on the experience of the researchers. The weight of the LSTM hidden layer represents the input of the particle swarm. The initial output error of the LSTM is used as the fitness of the particle swarm, then the particle performance is analysed according to condition. The random initial particle swarm updates its own parameter according to the individual extremum and global extremum.

For a better initialization of these three hyperparameters, in this research paper the advantages of the PSO method are combined with the LSTM method. The activities performed within the optimized LSTM method based on PSO are shown in the UML activity diagram drawn in Figure 2. It can be observed that after the activity of establishing the training set and before the training of the LSTM method, the PSO method is used to establish the values of the previously mentioned hyperparameters. The initial values of the PSO method parameters are 10 for the population size, 50 for the number of iterations, and 1.5 for the two acceleration factors, and the range of optimized LSTM hyperparameters is shown in Table 12.

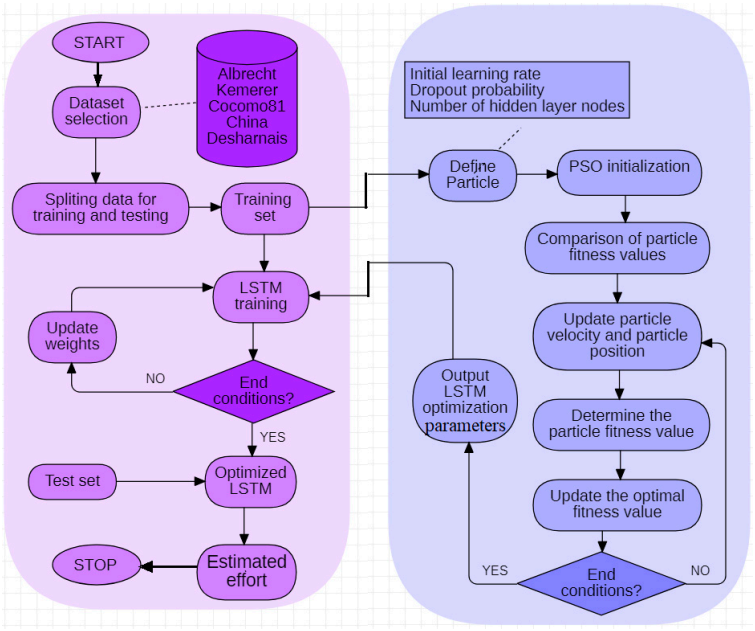

Figure 2. UML activity diagram.

Table 12. Range of optimized LSTM parameters.

Parameters	Lower Value	Upper Value
Initial learning rate	0.0001	0.1
Momentum	0.5	0.9
Dropout probability	0	1

To estimate the soft effort using the optimized LSTM method, the same two parameters (number of training epochs, and number of neurons in the LSTM hidden layer) are used in the tuning process as in the case of the standard LSTM method, their values belonging to the same sets. Table 13 shows the values provided by the 40 variants of the improved LSTM method, the meanings of columns from Table 13 being the same as those from Table 5.

Table 13. Test results for improved LSTM method.

Metrics	Datasets	Metric Values			Parameters		Estimated Effort		
		Minim	Maxim	Mean	e	n	Minim	Maxim	Mean
MAE	Albrecht	3.604	20.026	12.046	1000	100	7.879	102.378	62.199
	Kemerer	35.294	81.210	51.715	300	25	72.475	281.969	151.872
	Cocomo81	133.096	498.216	243.865	700	100	42.699	1996.398	875.823
	China	331.089	686.181	498.155	600	75	136.281	40,989.973	24,775.855
	Desharnais	1253.781	1857.354	1437.557	100	25	1369.367	14,461.862	7868.247
MdAE	Albrecht	2.902	11.568	5.092	1000	100	7.879	102.378	62.199
	Kemerer	13.747	83.125	47.836	1000	100	79.445	280.277	149.346
	Cocomo81	30.142	363.576	215.732	500	50	34.991	1239.811	676.557
	China	111.213	464.951	277.955	600	75	136.281	40,989.973	24,775.855
	Desharnais	849.929	1393.136	1007.793	400	25	667.482	7990.067	5908.635
RMSE	Albrecht	4.717	32.982	19.187	800	75	7.808	100.381	59.035
	Kemerer	41.412	133.247	78.945	100	75	75.541	274.121	147.326
	Cocomo81	217.794	790.626	373.584	700	100	42.699	1996.398	875.823
	China	873.102	1386.098	1175.075	500	75	116.507	42,016.782	29,775.095
	Desharnais	1535.852	2949.732	2211.123	100	25	1369.367	14,461.862	7868.247
CD	Albrecht	0.615	0.697	0.648	1000	100	7.879	102.378	62.199
	Kemerer	0.528	0.579	0.541	300	25	72.475	281.969	151.872
	Cocomo81	0.917	0.986	0.953	700	100	42.699	1996.398	875.823
	China	0.893	0.951	0.937	600	75	136.281	40,989.973	24,775.855
	Desharnais	0.627	0.683	0.645	100	25	1369.367	14,461.862	7868.247

The real effort values provided by the Albrecht test dataset are between 2.9 and 102.4 (Table 4). The range of effort values estimated by the model provided by the minimum values of MAE and MdAE metrics and by the maximum value of CD metric is determined by the values 7.879 and 102.378 (Table 13). The range of effort values estimated by the model provided by the minimum values of the RMSE metric is determined by the values 7.808 and 100.381.

The real effort values provided by the Kemerer test dataset are between 72 and 287 (Table 4). The range of effort values estimated by the model provided by the minimum values of the MAE metric and by the maximum value of the CD metric is determined by the values 72.475 and 281.969 (Table 13). The range of effort values estimated by the model provided by the minimum values of the MdAE metric is determined by the values 79.445 and 280.277. The range of effort values estimated by the model provided by the minimum values of the RMSE metric is determined by the values 75.541 and 274.121.

Figure 3 shows a graphical representation of the predicted values for the software effort by optimized LSTM method in the case of the minimum values of the metrics relative to the real effort specific to each of the five datasets.

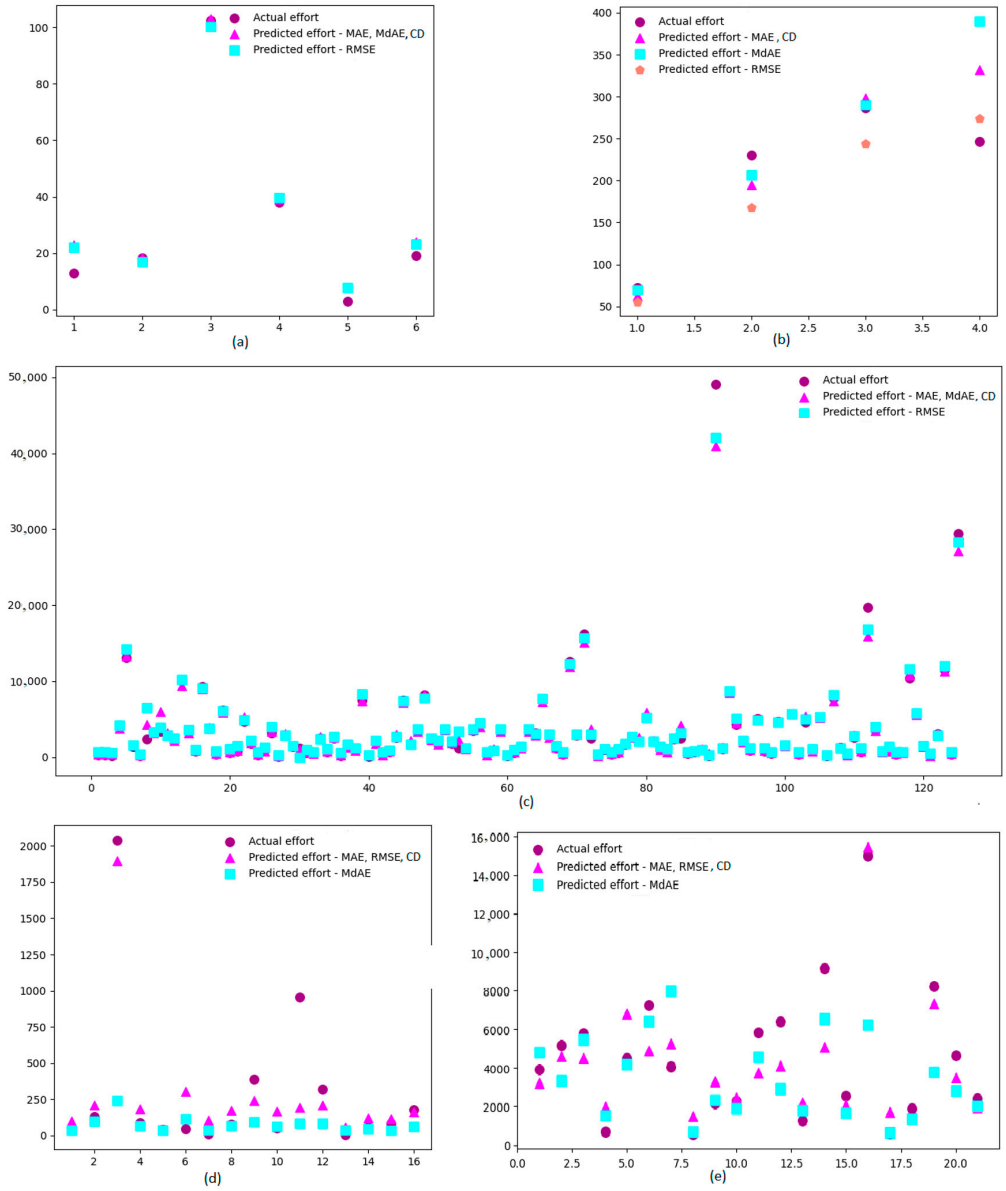

Figure 3. Effort estimated by optimized LSTM method compared to the real effort. (**a**) Albrecht dataset; (**b**) Kemerer dataset; (**c**) China dataset; (**d**) Cococmo81 dataset; (**e**) Desharnais dataset.

Comparing the minimum value 133.096 (Table 13) obtained for the MAE metric by means of the optimized LSTM method with the value 153 (Table 11) presented in the paper [7], the minimum value 217.794 obtained for the RMSE metric by means of the optimized LSTM method with the value 228.7 presented in the paper [7], and the maximum value 0.986 obtained for the CD metric by means of the optimized LSTM method with the value 0.98 presented in the same paper, it can be concluded that the optimized LSTM method presented in this study is more efficient than the model presented in [7] for the Cocomo81 dataset. The real effort values provided by the Cocomo81 test dataset are between 6 and 2040 (Table 4). The range of effort values estimated by the model provided by the minimum values of the MAE and RMSE metrics and by the maximum value of the CD metric is determined by the values 42.699 and 1996.398 (Table 13). The range of effort values estimated by the model provided by the minimum values of the MdAE metric is determined by the values 34.991 and 1239.811.

Comparing the minimum value 331.089 (Table 13) obtained for the MAE metric by means of the optimized LSTM method with the value 676.6 (Table 11) presented in the paper [7], the minimum value 873.102 obtained for the RMSE metric by means of the optimized LSTM method with the value 1803.3 presented in the paper [7], and the maximum value 0.951 obtained for the CD metric by means of the optimized LSTM method with the value 0.93 presented in the same paper, it can be concluded that the optimized LSTM method presented in this study is more efficient than the model presented in [7] for the China dataset. The real effort values provided by the China test dataset are between 89 and 49,034 (Table 4). The range of effort values estimated by the model provided by the minimum values of the MAE and MdAE metrics and by the maximum value of the CD metric is determined by the values 136.281 and 40,989.973 (Table 13).

The range of effort values estimated by the model provided by the minimum values of the RMSE metric is determined by the values 116.507 and 42,016.782. The real effort values provided by the Desharnais test dataset are between 546 and 14,987 (Table 4). The range of effort values estimated by the model provided by the minimum values of the MAE and RMSE metrics and by the maximum value of the CD metric is determined by the values 1369.367 and 14,461.862 (Table 13). The range of effort values estimated by the model provided by the minimum values of the MdAE metric is determined by the values 667.482 and 7990.067. For all five datasets, by comparing the real effort values from Table 4 with the estimated values from Table 13, it is observed that all estimated intervals are included in the intervals corresponding to the real values of the effort. Table 14 presents a synthesis of the optimal values obtained for all four used metrics in the case of all the intelligent methods used in this study.

Table 14. Optimal metrics values.

Datasets	Methods	MAE	MdAE	RMSE	CD
Albrecht	KNN	11.228	4.556	18.202	0.437
	DT	6.975	3.825	10.213	0.561
	RF	6.015	3.579	7.726	0.593
	GBT	5.892	3.092	7.094	0.609
	MLP	7.555	3.256	8.208	0.547
	LSTM	4.870	2.911	6.560	0.631
	Optimized LSTM	3.604	2.902	4.717	0.697
Kemerer	KNN	75.519	57.933	90.933	0.351
	DT	95.853	81.802	106.226	0.309
	RF	82.739	69.174	83.726	0.327
	GBT	60.197	58.682	64.045	0.392
	MLP	48.145	47.427	58.217	0.438
	LSTM	42.094	26.206	57.560	0.493
	Optimized LSTM	35.294	13.747	41.412	0.579

Table 14. *Cont.*

Datasets	Methods	MAE	MdAE	RMSE	CD
Cocomo81	KNN	221.619	35.783	446.987	0.793
	DT	275.979	48.617	514.792	0.774
	RF	235.857	43.925	487.283	0.732
	GBT	209.198	39.826	463.782	0.812
	MLP	199.513	57.833	355.245	0.826
	LSTM	178.051	30.642	245.153	0.897
	Optimized LSTM	133.096	30.142	217.794	0.986
China	KNN	1747.048	611.602	3922.755	0.762
	DT	1575.633	755.642	2848.557	0.793
	RF	1409.324	621.782	2792.372	0.821
	GBT	1287.382	504.672	2563.824	0.865
	MLP	1383.713	517.898	2892.196	0.843
	LSTM	865.122	272.353	2034.574	0.902
	Optimized LSTM	331.089	111.213	873.102	0.951
Desharnais	KNN	1422.878	921.667	1847.561	0.658
	DT	1953.361	1453.083	2526.071	0.563
	RF	1792.837	1284.947	2379.063	0.581
	GBT	1549.792	901.925	2095.827	0.639
	MLP	1587.619	905.841	2176.591	0.624
	LSTM	1404.571	880.121	1960.007	0.662
	Optimized LSTM	1253.781	849.929	1535.852	0.683

From the comparison of the values of the metrics obtained (Tables 11 and 14), it is concluded that the optimized LSTM method provides a more efficient option for estimating the software effort.

6. Conclusions

Estimating the effort required to develop software products is one of the most vexing problems for software project managers because it affects the status of the projects in terms of success or failure. This research paper proposes an optimized LSTM neural network method which use PSO methods to optimize three hyperparameters of LSTM with the aim of obtaining the most accurate estimate of the effort required to develop a software product. The results obtained by the optimized LSTM method were compared with the results provided by six other prediction methods: KNN, DT, RF, GBT, MLP, and LSTM, by applying to the values of five datasets: Albrecht, Kemerer, Cococmo81, China, and Desharnais. The superiority of the optimized LSTM method compared to the other six prediction methods resulted from obtaining lower values for the three metrics used in the evaluation: MAE, MdAE, RMSE, and CD.

There were some limitations, especially for small datasets such as Albrecht and Kemerer, which have a small number of observations and low dimensionality. Future work will be needed to investigate the reason for this discrepancy and what optimization method can search in small datasets. In addition, as a further direction, optimization of the LSTM method should be attempted using other techniques with the purpose to obtain a better estimation of effort estimation with the existing datasets.

From this scientific work, software developers will be able to benefit in selecting the best models for predicting the development effort of software products before they are developed.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in Albrecht dataset at [10.1109/TSE.1983.235271] [15]. Kemerer dataset https://zenodo.org/record/268464 [16]. Cocomo81 dataset https://doi.org/10.1109/TSE.1984.5010193 [17]. China dataset https://zenodo.org/record/268446 [18]. Desharnais dataset [19].

Conflicts of Interest: The author declares no conflict of interest.

References

1. Panoiu, M.; Panoiu, C.; Mezinescu, S.; Militaru, G.; Baciu, I. Machines Learning Techniques Applied to the Harmonic Analysis of Railway Power Supply. *Mathematics* **2023**, *11*, 1381. [CrossRef]
2. Walter, B.; Jolevski, I.; Garnizov, I.; Arsovic, A. Supporting Product Management Lifecycle with Common Best Practices. In *Systems, Software and Services Process Improvement*; Springer: Grenoble, France, 2023; pp. 207–215.
3. Muscalagiu, I.; Popa, H.E.; Negru, V. Improving the Performances of Asynchronous Search Algorithms in Scale-Free Networks using the Nogoood Processor Technique. *Comput. Inform.* **2015**, *34*, 254–274.
4. Iordan, A.E. Supervised learning use to acquire knowledge from 2D analytic geometry problems. In *Recent Challenges in Intelligent Information and Database Systems*; Springer: Singapore, 2022; pp. 189–200.
5. Marapelli, B. Software Development Effort and Cost Estimation using Linear Regression and K-Nearest Neighbours Machine Learning Algorithms. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *9*, 2278–3075.
6. Hameed, S.; Elsheikh, Y.; Azzeh, M. An Optimized Case-Based Software Project Effort Estimation Using Genetic Algorithm. *Inf. Softw. Technol.* **2023**, *153*, 107088. [CrossRef]
7. Kumar, P.S.; Behera, H.; Nayak, J.; Naik, B. A Pragmatic Ensemble Learning Approach for Effective Software Effort Estimation. *Innov. Syst. Softw. Eng.* **2022**, *18*, 283–299. [CrossRef]
8. Singh, A.J.; Kumar, M. Comparative Analysis on Prediction of Software Effort Estimation using Machine Learning Techniques. In Proceedings of the International Conference on Intelligent Communication and Computational Research, Punjab, India, 25 January 2020.
9. Zakaria, N.A.; Ismail, A.R.; Ali, A.Y.; Khalid, N.H.; Abidin, N.Z. Software Project Estimation with Machine Learning. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 726–734. [CrossRef]
10. Fedotova, O.; Teixeira, L.; Alvelos, H. Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. *J. Inf. Sci. Eng.* **2018**, *29*, 925–945.
11. Abdelali, Z.; Mustapha, H.; Abdelwahed, N. Investigating the Use of Random Forest in Software Effort Estimation. *Procedia Comput. Sci.* **2019**, *148*, 343–352. [CrossRef]
12. Sanchez, E.R.; Santacruz, E.F.V.; Maceda, H.C. Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development. *Mathematics* **2023**, *11*, 1477. [CrossRef]
13. Resmi, V.; Anitha, K.L.; Narasimha Murthy, G.K. Optimized Satin Bowerbird for Software Project Effort Estimation. *Eur. Chem. Bull.* **2023**, *12*, 410–423.
14. Muhammad, A.L.; Khalid, M.K.; Hani, U. Using Standard Deviation with Analogy-Based Estimation for Improved Software Effort Prediction. *KSII Trans. Internet Inf. Syst.* **2023**, *17*, 1356–1375.
15. Albrecht, A.J.; Gaffney, J.E. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Trans. Softw. Eng.* **1993**, *9*, 639–648. [CrossRef]
16. Zenodo. Kemerer. Available online: https://zenodo.org/record/268464 (accessed on 11 July 2023).
17. Boehm, B.W. Software Engineering Economics. *IEEE Trans. Softw. Eng.* **1984**, *10*, 4–21. [CrossRef]
18. Zenodo. China: Effort Estimation Dataset. Available online: https://zenodo.org/record/268446 (accessed on 15 July 2023).
19. Desharnais, J.M. Analyse Statistique de la Productivitie des Projets Informatique a Partie de la Technique des Point des Function. Master's Thesis, University of Montreal, Montréal, QC, Canada, 1999.
20. Panoiu, M.; Panoiu, C.; Iordan, A.; Ghiormez, L. Artificial Neural Networks in Predicting Current in Electric Arc Furnaces. *IOP Conf. Ser. Mater. Sci. Eng.* **2014**, *57*, 012011. [CrossRef]
21. Handelman, G.S.; Kok, H.K.; Chandra, R.; Razavi, A.; Huang, S.; Brooks, M.; Lee, M.; Asadi, H. Peering into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods. *Am. J. Roentgenol.* **2019**, *212*, 38–43. [CrossRef]
22. Botchkarev, A. Performance Metrics in Machine Learning Regression, Forecasting and Prognostics: Properties and Topology. *Interdiscip. J. Inf. Knowl. Manag.* **2019**, *14*, 45–79.
23. Hossin, M.; Sulaiman, M.N. A Review on Evaluation Metrics for Data Classification Evaluations. *Int. J. Data Min. Knowl. Manag. Process* **2015**, *5*, 1–12.
24. Gavin Hackeling, G. *Mastering Machine Learning with Scikit-Learn*; Packt Publishing Ltd.: Birmingham, UK, 2018.
25. Covaciu, F.; Pisla, A.; Iordan, A.E. Development of a Virtual Reality Simulator for an Intelligent Robotic System Used in Ankle Rehabilitation. *Sensors* **2021**, *21*, 1537. [CrossRef]
26. Patel, H.; Prajapati, P. Study and Analysis of Decision Tree Based Classification Algorithms. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 74–78. [CrossRef]
27. Spoon, K.; Beemer, J.; Whitmer, J.; Fan, J.; Frazee, J.; Stronach, J.; Bohonak, A.; Levine, R. Random Forests for Evaluating Pedagogy and Informing Personalized Learning. *J. Educ. Data Min.* **2016**, *8*, 20–50.
28. Castro-Martín, L.; Mar Rueda, M.; Ferri-García, R.; Hernando-Tamayo, C. On the Use of Gradient Boosting Methods to Improve the Estimation with Data Obtained with Self-Selection Procedures. *Mathematics* **2021**, *9*, 2991. [CrossRef]
29. Iordan, A.E. Usage of Stacked Long Short-Term Memory for Recognition of 3D Analytic Geometry Elements. In Proceedings of the International Conference on Agents and Artificial Intelligence, Lisbon, Portugal, 3–5 February 2022.

30. Alamia, A.; Gauducheau, V.; Paisios, D.; VanRullen, R. Comparing Feedforward and Recurrent Neural Network Architectures with Human Behavior in Artificial Grammar Learning. *Sci. Rep.* **2020**, *10*, 22172. [CrossRef] [PubMed]
31. Awar, N.; Zhu, S.; Biros, G.; Gligoric, M. A performance portability framework for Python. In Proceedings of the ACM International Conference on Supercomputing, New York, NY, USA, 14–18 June 2021.
32. Ullo, S.L.; Del Rosso, M.P.; Sebastianelli, A.; Puglisi, E.; Bernardi, M.L.; Cimitile, M. How to develop your network with Python and Keras. *Artif. Intell. Appl. Satell.-Based Remote Sens. Data Earth Obs.* **2021**, *98*, 131–158.
33. Hunt, J. Introduction to Matplotlib. In *Advanced Guide to Python 3 Programming*; Springer: Cham, Switzerland, 2019; Volume 5, pp. 35–42.
34. Iordan, A.E.; Covaciu, F. Improving design of a triangle geometry computer application using a creational pattern. *Acta Tech. Napoc. Appl. Math. Mech. Eng.* **2020**, *63*, 73–78.
35. Covaciu, F.; Crisan, N.; Vaida, C.; Andras, I.; Pusca, A.; Gherman, B.; Radu, C.; Tucan, P.; Hajjar, N.A.; Pisla, D. Integration of Virtual Reality in the Control System of an Innovative Medical Robot for Single-Incision Laparoscopic Surgery. *Sensors* **2023**, *23*, 5400. [CrossRef] [PubMed]
36. Mabayoje, A.; Balogun, A.; Hajarah, H.; Atoyebi, J.; Mojeed, H.; Adeyemo, V. Parameter tuning in KNN for software defect prediction: An empirical analysis. *J. Teknol. Sist. Komput.* **2019**, *7*, 121–126. [CrossRef]
37. Kumbure, M.; Luukka, P. A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance. *Granul. Comput.* **2022**, *7*, 657–671. [CrossRef]
38. Uyanik, B.; Orman, G.K. A Manhattan distance based hybrid recommendation system. *Int. J. Appl. Math. Electron. Comput.* **2023**, *11*, 20–29. [CrossRef]
39. Iordan, A.E. Optimal Solution of the Guarini Puzzle Extension using Tripartite Graphs. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *477*, 012046. [CrossRef]
40. Roshanski, I.; Kalech, M.; Rokach, L. Automatic Feature Engineering for Learning Compact Decision Trees. *Expert Syst. Appl.* **2023**, *229*, 120470. [CrossRef]
41. Yu, Y.; Wang, L.; Huang, H.; Yang, W. An Improved Random Forest Algorithm. *J. Phys. Conf. Ser.* **2020**, *1646*, 012070. [CrossRef]
42. Xia, Y.; Chen, J. Traffic Flow Forecasting Method based on Gradient Boosting Decision Tree. *Adv. Eng. Res.* **2017**, *130*, 413–416.
43. Han, Y.; Zhang, Z.; Kobe, F. The Hybrid of Multilayer Perceptrons: A New Geostatistical Tool to Generate High-Resolution Climate Maps in Developing Countries. *Mathematics* **2023**, *11*, 1239. [CrossRef]
44. Hsieh, S.C. Tourism demand forecasting based on an LSTM network and its variants. *Algorithms* **2021**, *14*, 243. [CrossRef]
45. Higashitani, M.; Ishigame, A.; Yasuda, K. Particle swarm optimization considering the concept of predator-prey behavior. In Proceedings of the IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 434–437.
46. Lv, L.; Kong, W.; Qi, J.; Zhang, J. An improved long short-term memory neural network for stock forecast. *MATEC Web Conf.* **2018**, *232*, 01024. [CrossRef]
47. Zheng, C.; Li, H. The Prediction of Collective Economic Development based on the PSO-LSTM Model in Smart Agriculture. *PeerJ Comput. Sci.* **2023**, *9*, 1304. [CrossRef]
48. Chen, X.; Long, Z. E-Commerce Enterprises Financial Risk Prediction Based on FA-PSO-LSTM Neural Network Deep Learning Model. *Sustainability* **2023**, *15*, 5882. [CrossRef]
49. Qu, Z.; Yin, J. Optimized LSTM Networks with Improved PSO for the Teaching Quality Evaluation Model of Physical Education. *Int. Trans. Electr. Energy Syst.* **2022**, *2022*, 8743694. [CrossRef]