ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse119922020.pdf
https://doi.org/10.30534/ijatcse/2020/119922020

Software Effort Classification with Multilayer Perceptron Neural Networks

Ahmed Fawzi Otoom[1], Alaa Alzaben[2], Razan Tlailan [3], Maen Hammad [4]
[1]Department of Software Engineering, The Hashemite University, Jordan, aotoom@hu.edu.jo
[2] Department of Software Engineering, The Hashemite University, Jordan, Alaaalzaben1995@itc.hu.edu.jo
[3] Department of Software Engineering, The Hashemite University, Jordan, Razan94@itc.hu.edu.jo
[4] Department of Software Engineering, The Hashemite University, Jordan, mhammad@hu.edu.jo

ABSTRACT

We target the problem of software effort estimation from a classification perspective. Our main goal is to build a classifier that can predict the required effort for a new project and assign it into one of the four classes: small, small-medium, medium-large, and large. A criterion is proposed for the aforementioned categorization and based on the amount of required effort. We study data sets that are based on three different categories: function points-based data sets, COCOMO-based data sets, and project characteristics-based data sets. Feature sets are prepared and fed to the multilayer perceptron (MLP) neural network algorithm. A hold-out test is implemented and ROC curve is used as a measure of the performance of the algorithm. In addition, we identify the important features for building the classification models across various data sets. Generally, MLP shows good performance across the six data sets. Moreover, there is a stability of performance within each category of data sets or across different categories with no dramatic differences in results. However, the performance of MLP seems to decrease with data sets that are based on project characteristics.

Key words: Software effort estimation, multilayer perceptron, neural network, function points, COCOMO

1. INTRODUCTION

A successful software project is a project that meets its objectives, desired budget, and scheduled deadline. According to the Standish Group International, only 29% of projects in the year 2015 were considered successful with the rest of projects being failed or cancelled [1]. In a study of Oxford's university, it was shown that 66% of large software projects overrun scheduled budget and 33% of projects are over schedule [2-3].

A key to success is proper estimation of the required effort, resources, time, and the cost for carrying out the project. Among the aforementioned, accurate effort estimation is crucial as it helps in resources allocation and good management of quality and budget. Overestimating the required effort and hence, the duration and cost of the project, may result on a company losing its contracts or wasting its resources [4], whereas underestimation may result on understaffing problem, having a low quality product, setting a tight schedule, and losing money [5].

A variety of software estimation techniques have been proposed in the literature for supporting project managers in estimating effort. Generally, estimation techniques analyze a new project's data and compare it to historical data sets of previous projects containing measurements of relevant metrics and the related effort [6]. Overall, software estimation techniques are based on three main approaches: 1) expert judgment by analogy, 2) formal models such as function points analysis and Constructive Cost Model (COCOMO), and 3) machine learning approaches.

Among the aforementioned approaches, machine learning approaches attracted large attention. They include tree-based models [7-11], ensemble methods [4], [9], and nonlinear methods such as neural networks [4],[9],[12]. The main aim is to predict the exact value of the required effort for a new software project, a regression problem. Machine learning algorithms proved to be useful for this problem.

In this paper, we target this problem but from a classification perspective rather than a regression perspective. The aim is to assign a new project's required effort into one of the four classes: small, small-medium, medium-large, and large, depending on the amount of the required effort for this project. For this purpose, we examine the performance of multilayer perceptron (MLP) classification algorithm on three different categories: function points-based data sets, COCOMO-based data sets, and project characteristics-based data sets. The MLP is chosen as it has proved to be successful within this domain.

We believe that this classification is useful in the initial planning stage of the project to give a general indication of the amount of effort and hence, the duration and resources required of the new project. If the project is approved and within the constraints of the stakeholder, it is taken to another

stage of detailed planning and detailed estimation. The research contributions of this paper can be summarized as follows:

- Analyzing six software effort data sets of three different categories: function points-based data sets, COCOMO-based data sets, and project characteristics-based data sets.
- Categorization of software effort into four predefined classes: small, small-medium, medium-large, and large.
- Building a classifier with multilayer perceptron (MLP) neural network algorithm and analyzing its classification results on three different categories of data sets.
- Identification of the important features in building the classification models and for each category of the data sets.

The rest of this paper is organized as follows: in section 2, we discuss the related work. Section 3 presents the approach. In section 4, we discuss the results. Finally, section 5 concludes our work and suggests future work.

2. RELATED WORK

The area of software effort estimation has gained attention from researchers' worldwide because of its importance in the planning stage of the software project management lifecycle.

The first approach for estimating software development is expert judgment by analogy approach. In this approach, an expert in the domain of the project applies a prior experience for estimating the required effort of a new project. A Delphi technique can also applied where the effort estimation is decided by a set of independent experts and the median of these estimates can be used as the final effort [6].

In recent years, there has been a focus on other techniques for effort estimation such as Function Points' Analysis [13] and Constructive Cost Models [14-15]. In function points' analysis, the size of a new project is calculated in function points (FP) and as a multiplication between two terms: UFP and TCF. UFP refers to unadjusted function points and is calculated based on metric assigns complexity to five components of the product (inputs, outputs, inquires, master files, and interfaces). The other term, TCF refers to technical complexity factor and it assigns complexities to fourteen factors that are related to the performance, maintainability and other issues of the system. Once the size in FPs of the new product is calculated, the estimation of the effort is carried out and based on historical data sets [16].

Another widely applied method for software effort estimation is Intermediate Constructive Cost Model (COCOMO). Effort is calculated based on the development mode of the project and a set of effort multipliers that are assessed on a specific metric [16]. The COCOMO model is flexible and applicable to different types of project [17].

More recently, there has been a focus on the application of machine learning techniques for the purpose of predicting the effort of a new project. These techniques include tree-based models [7-11], ensemble methods [4],[9], and nonlinear methods such as neural networks [4],[9],[12].

Tree-based models have attracted attention from researchers for estimating software effort. For example, the authors in [7] compared the effort estimation performance of decision trees, or Classification And Regression Trees (CART), and the performance of decision tree forest, or random forest. The two algorithms are examined on ISBSG and Desharnais data sets, with random forest outperforming the performance of CART. A similar approach was implemented by the authors in [11] with an examination on higher number of data sets that include: ISBSG, COCOMO, Tukutuku, Desharnais, and Albrecht. Two techniques are used: random forests and regression trees. The two techniques seem to perform well on the data sets with random forests performance outperforming that of regression trees on the five data sets.

Within this context, there has been a focus on another type of trees, M5P trees. M5P implements both regression and model trees. For example, the authors in [9] studied the performance of M5P trees on two data sets NASA and Desharnais. It performs well with comparable results with other regression methods. However, when the algorithm is integrated within a bagging algorithm, it provided the highest performance results on the NASA data set. In the work of [10], M5P was among the best two algorithms when tested on COCOMO'81 data set.

The aforementioned bagging algorithm, in [9], is a type of ensemble classification algorithms which have been implemented widely for software effort estimation. For example, the authors in [4] compared the performance of single learners (MLP, RBF and regression tress) and ensembles learners that include bagging and random trees. Bagging with MLP seems to be among the best classifiers when tested on a number of data sets from PROMISE and the ISBSG data set. In [10], a voting approach for combination of learners is applied on three data sets with COCOMO related data. The finding indicates no enhancement of multiple learners' performance compared to that of single learners' performance. On other hand, neural networks models have got a lot of popularity within this field. For example, the authors in [12] tested the performance of multilayer perceptron (MLP) algorithm on Desharnais data set and compared it with other algorithms that include linear regression model, k-nearest neighbor, and support vector machine for the purpose of software effort estimation. The dimensionality of the Desharnais data set is reduced based on Pearson correlation approach. The reduced data is then fed to the classifiers. MLP provided the best performance results. In the works of [4] and [9], MLP was among the best algorithms when tested on different software estimation data sets.

Moreover, in the work of [18], MLP proved successful when experimented on the COCOMO'81 data set with low relative error when compared to the actual effort that is calculated with COCOMO II. Also, MLP has proved successful of the same aforementioned data set in the work of [19].

3. THE APPROACH

The proposed approach consists of three main stages: data collection, data preparation, and classification.

3.1 Date Collection

We use six well-known software effort estimation data sets. Three of these data sets are from PROMISE software engineering repository [20]: COCOMO'81, COCOMO NASA 2, and Desharnais. The other three data sets are: Albrecht [21], China [21] and ISBSG data set. We divide the data sets into three categories depending on the measurements of different metrics: function points-based data sets, COCOMO-based data sets, and project characteristics-based data sets. Albrecht and China data sets are function points-based data sets. They describe projects based on function points' related information. Albrecht data set presents data related to 24 projects with 7 features describing numerical numbers in relation to number of files, inputs, outputs, inquiries and other information. China data set expresses 499 projects with similar features to that of Albrecht but with addition of other features that are related to resources, duration, and numbers of changes, added and modified files, ending up in 18 features.

COCOMO'81 and COCOMO NASA 2 data sets are COCOMO-based data sets that characterize projects based on COCOMO multipliers. COCOMO'81 data set presents data related to 63 projects with effort measured in calendar months of 152 hours. The data set is represented with 16 features that reflect 15 COCOMO multipliers and the lines of code (LOC) feature. The COCOMO NASA 2 presents information for 93 NASA projects from different centers with effort measured in calendar months of 152 hours. The data set is represented with 23 features that reflect 15 COCOMO multipliers and other features related to development mode, category of application, year of development, and other information.

Desharnais and ISBSG data sets are project characteristics-based data sets that depend on different characteristics of the projects. Desharnais data set describes 81 projects with 11 features that are related to team and manger experience, language used, length of the project and other information. The ISBSG data set is one of the popular data sets in this domain. We use the ISBSG data set Release 12. It presents variety of information in relation to organization type, application type, development platform, productivity, schedule features, effort information and others with a total of 69 features.

3.2 Date Preparation

After analyzing the data sets, data preparation is carried out as follows:

- Albrecht data set: the feature RawFPCounts is omitted as there is another feature AdjFP which reflects very similar information, to end up with 6 features.
- China data set: the following features are omitted: project ID, development type as it has only one value, and N-effort as it reflects very similar information to effort, to end up with 15 features.
- COCOMO'81 data set: no modification to the original data set.
- COCOMO NASA 2 data set: the following features are omitted: project ID, project name, and year of development, to end up with 20 features.
- Desharnais data set: the project ID is deleted to end up with 10 features.
- ISBSG: we studied this data set and consulted software engineering experts in regard to the important features that have influence on the effort, to end up with the following features: data quality rating, industry sector, organization type, application type, development type, language type, functional size, FP standard, and the used methodology. For the projects with too many missing values, they are excluded from the data set, to end up with 2294 projects and a dimensional space of 10 dimensions (features).

Table 1 shows a summary of the data sets before and after preparation. Moreover, tables (2-7) show a short description of the features in each data set.

Table 1: Data sets before and after preparation

Category	Data set	# Features		# Instances	
		before	after	before	after
Function-points based data sets	Albrecht	7	6	24	24
	China	18	15	499	499
COCOMO –based data sets	COCOMO'81	16	16	63	63
	COCOMO NASA 2	23	20	93	93
Project characteristics-based data sets	Desharnais	11	10	81	81
	ISBSG	69	10	6009	2294

Table 2: Albrecht data set

Feature	Description
Input	Function points of input
Output	Function points of output
Inquiry	Function points of external enquiry
File	Function points of internal logical files
FPAdj	Normalized Adjusted function points
AdjFP	Adjusted function points

Table 3: China data set

Feature	Description
AFP	Adjusted Function points (FP)
Input, output, inquiry, file	Description in Table 2
Interface	FPs for external interface added
Added	FPs of added function
Changed	FPs of changed function
Deleted	FPs of deleted function
PDR_AFP	Productivity delivery rate – AFP
PDR_UFP	Productivity delivery rate – unadjusted FP
NPDR_AFP	Normalized Productivity delivery rate – AFP
NPDU_UFP	Normalized PDR – unadjusted FP
Resource	Team type
Duration	Duration of the project

Table 4: COCOMO'81 data set

Feature	Description
rely	Required software reliability multiplier
data	Database size multiplier
cplx	Process complexity multiplier
time	Time constraint for CPU multiplier
stor	Main memory constraint multiplier
virt	Machine volatility multiplier
turn	Turnaround time multiplier
acap	Analysts capability multiplier
aexp	Application experience multiplier
pcap	Programmer capability multiplier
vexp	Virtual machine experience multiplier
lexp	Language experience multiplier
modp	Modern programming practices multiplier
tool	Use of software tools multiplier
sced	Schedule constraint multiplier
loc	Lines of code

Table 5: COCOMO NASA2 data set

Feature	Description
cagetory_of_application (category)	Application type
flight_or_ground_system (system)	A flight or a ground system
which_nasa_center (center)	The NASA center
developmen_mode (dev_mode)	Mode: organic, embedded or semidetached
The COCOMO 15 multipliers	Description in Table 4
equivphyskloc_real (Kloc)	Kilo Lines of code

Table 6: Desharnais data set

Feature	Description
TeamExp	Team experience in years
ManagerExp	Manager's experience in years
YearEnd	Year of completion
Length	Length of project
Transactions	Count of basic logical transactions
Entities	The number of entities in the systems data model
PointsAdjust	Size of project measured in adjusted function points
Envergure	Function points complexity adjustment factor
PointsNonAjust	Unadjusted FP
Langage	The used programming language

Table 7: ISBSG data set

Feature	Description
rating	Rating of the quality of data
Sector	The sector for the application
Organisation_Type (Org_type)	The type of the organization that submitted the project
Application_Type (App_type)	The type of the application being addressed by the projects
Development_Type (Dev_type)	Primary development platform
Development_Platform (Dev_plat)	The number of entities in the systems data model
Language_Type (lang_type)	The language type
Functional_Size (size)	Unadjusted FP
Standard	The used function size metric
Methodology	The used methodology

For each data set, we assign class labels as follows: first, we calculate the minimum, maximum, and median values of the effort across all projects, we refer to it as *MED*. Then, we calculate the median between the minimum value of the effort and *MED*, we refer to it as *M1*. After that, we calculate the median between *MED* and the maximum value of the effort, we refer to as *M2*. The effort classes are assigned as follows:

$$Effort = \begin{cases} small, & min \le effort < M1 \\ sml-med, & M1 \le effort < MED \\ med-lrg, & MED \le effort < M2 \\ large, & M2 \le effort \le max \end{cases} \quad (1)$$

Where *sml-med* refers to small-medium category, and *med-lrg* refers to medium-large category. The number of instances of each class and across the six data sets is illustrated in table 8.

Table 8: Number of instances across classes

Data set	small	sml-med	med-lrg	large	Total
Albrecht	7	5	7	5	24
China	126	124	124	125	499
COCOMO'81	17	15	16	15	63
COCOMO NASA 2	24	23	23	23	93
Desharnais	21	26	14	20	81
isbsg	668	351	335	940	2294

3.3 Classification

Once the feature sets are prepared, they are fed into an MLP neural network for supervised learning and classification. MLP is chosen as it proved to be successful with this domain [4],[9],[12], [18], [19] and other research areas [22-23]. An MLP is a class of feedforward artificial neural network (ANN) that consists of a network of neurons arranged in three layers: input layer, a hidden layer(s), and an output layer. Each neuron processes its inputs and generates one output value using a transfer function that is transmitted to the neurons in the subsequent layer [6],[24].

The output of hidden neuron *i* is calculated by processing the weighted inputs and its bias term $b_i^{(1)}$ as follows

$$h_i = f^{(1)}\left(b_i^{(1)} + \sum_{j=1}^{n} W_{ij}x_j\right) \quad (2)$$

WhereW_{ij} is the weight connecting input j to hidden unit i. For the output layer, it is calculated as follows

$$z = f^{(2)}\left(b^{(2)} + \sum_{j=1}^{n_h} v_j h_j\right) \quad (3)$$

Where n_h is the number of hidden neurons and v_j is the weight connecting hidden unit j to the output layer [6],[24].

Figure 1 shows an MLP neural network example for one of the studied data sets, Desharnais. The data feeds forward from the input layer through the hidden layer to the output layer. From this figure, we can notice that the input layer consists of ten unites, excluding the bias unit. There is also one hidden layer with two units. Moreover, there is a dependent variable, effort, in the output layer, with four units corresponding to the four classes. The hidden layer transfer function in this example, and for other data sets, is Hyperbolic Tangent function. It links the weighted sums of units in the hidden layer to the values of units in the output layer. For the output layer, the transfer function is Softmax.

For validation, we run holdout test, where almost 2/3 of the data is used for training the classifier and building the classification model. Then, the rest of the data is tested where the model predicted samples are compared with the original ones and the accuracy is calculated as follows:

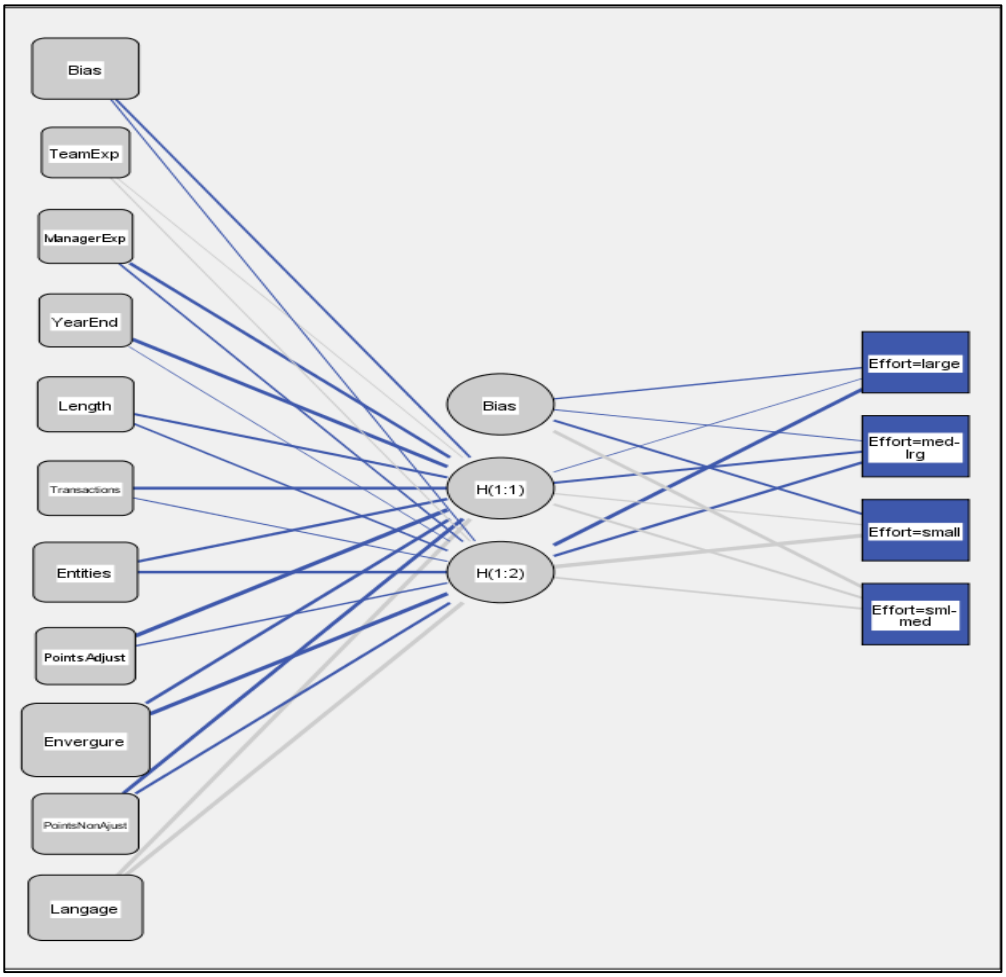


Figure 1: MLP neural network example

$$Accuracy = \frac{\#\ correctly\ classified\ smaples}{Total\ number\ of\ tested\ samples} \quad (4)$$

For further analysis of the data, we present the receiver operating characteristics (ROC) curves across the six data sets. ROC curves feature true positive rate (sensitivity) on the y-axis and false positive rate (1-specifity) on the x-axis. Moreover, we illustrate the confusion matrices associated with the MLP performance on each data set. In these matrices, the rows represent the ground truth classes and the columns represent the predicted classes. The cells blue-levels visually encode the performance percentages from 0%=pale blue to 100%=dark blue.

4. RESULTS AND ANALYSIS

Experiments are conducted to evaluate the performance of MLP on the six data sets. We present the experiments' results in terms of accuracy, ROC curve, and confusion matrix. We discuss results based on the three categories of data sets. Table 9 presents the accuracy results across the three categories. Moreover, figure 2 illustrates the ROC curves for various data sets.

Table 9: Accuracy results across different data sets

Category	Data set	Accuracy(%)
Function-points based data sets	Albrecht	83.3
	China	89.7
COCOMO-based data sets	COCOMO'81	76.5
	COCOMO NASA 2	85.7
Project characteristics-based data sets	Desharnais	70.0
	ISBSG	61.4

4.1 Function-points based data sets

As it is clear from Table 9, MLP classification algorithm provides very good accuracy results on both data sets where the classification performance on China data set outperforms that on Albrecht data set. The resources and duration features are expressed in the China data set, in addition to the other function points –based features. This seems to provide a value in classification performance.

For further analysis of the data and for better understanding for the performance of MLP on both data sets, we provide in figure 2 (a-b), the receiver operating characteristics (ROC) curves across both data sets.

The ROC curve for the Albrecht seems to be ideal with the curves being plotted on the upper left corner. In this data set, the testing data set is small with only six instances. As clear from the confusion matrix in figure 3(a), three classes are classified with 100% accuracy: small, med-lrg, and large. The sml-med class has one instance in the testing data which is classified wrongly. It is misclassified as small instead of sml-med.

In regard to the China data set, the ROC curve looks excellent with high values of true positive rates and low values of false positive rate and this clear in the confusion matrix as shown in figure 3 (b) with excellent percentages across the diagonal of the matrix.

4.2 COCOMO-based data sets

As it is clear from Table 9, MLP performs well on both data sets with the performance on COCOMO NASA 2 outperforming that on COCOMO'81. In the former data set, COCOMO NASA 2, there exists additional information such as the category of the application and the development mode. This information seems to be important for building a more accurate classifier. In figure 2 (c-d), the ROC curves on both data sets are illustrated.

As seen from figure 2 (c) and for the COCOMO'81 data set, MLP performance is excellent for the large and small classes and seems to perform well for the classification of the med-lrg class. However, the classification of the class sml-med is not so good as clear from figure 3 (c). It is noted that all the samples of this class are misclassified with half of them being classified wrongly as small class and the other half as med-lrg class.

For the COCOMO NASA 2 data set, MLP performance seems to perform well on the all classes and as clear in figure 2 (d). However, by looking at the confusion matrix in figure 3 (d), we notice that the large classes has been excluded from the testing sample as one or more cases in the testing sample have variable values that do not occur in the training sample.

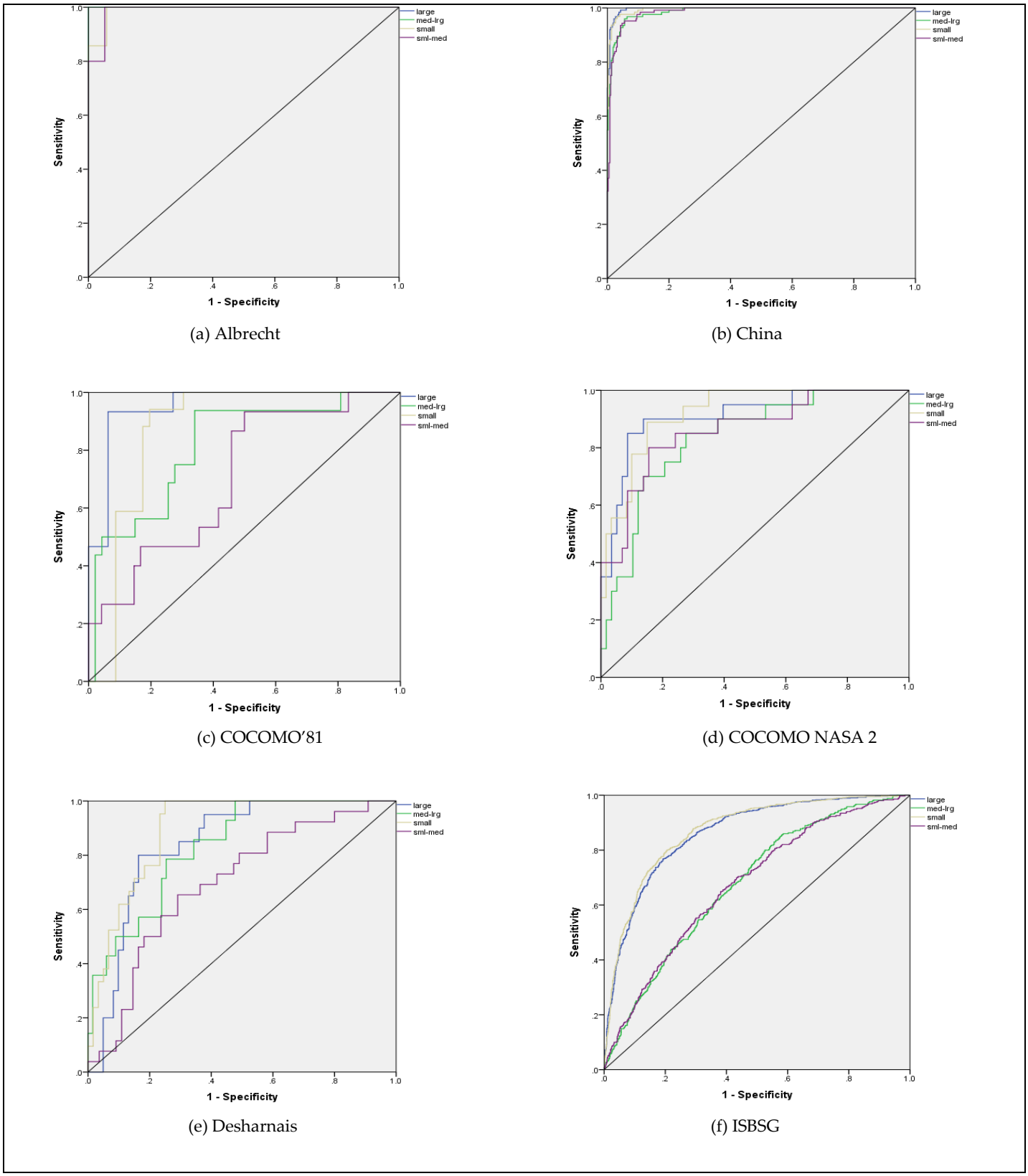
4.3 Project characteristics-based data sets

As it is clear from Table 9, MLP provides acceptable results on both data sets. On ISBSG data set, with the increasing size of the testing data set, the accuracy results decreases slightly. In figure 2 (e-f), the ROC curves on both data sets are illustrated.

For the Desharnais data set, and as seen from Figure 2 (e), MLP performs well for the small and large classes. As clear from figure 3 (e), majority of the tested samples in the small class are classified correctly and all in the large class are classified correctly. However, all the samples of the med-lrg class are classified wrongly as sml-med or large class.

For the ISBSG data set, MLP seems to perform well on the large and small data sets only as clear in figure 2 (f) and figure 3 (f). The number of instances in these two classes is much higher than that of the other two classes. This seems to be helpful of better training of these two data sets.

Overall, MLP shows good performance across the six data sets. Moreover, there is a stability of performance across

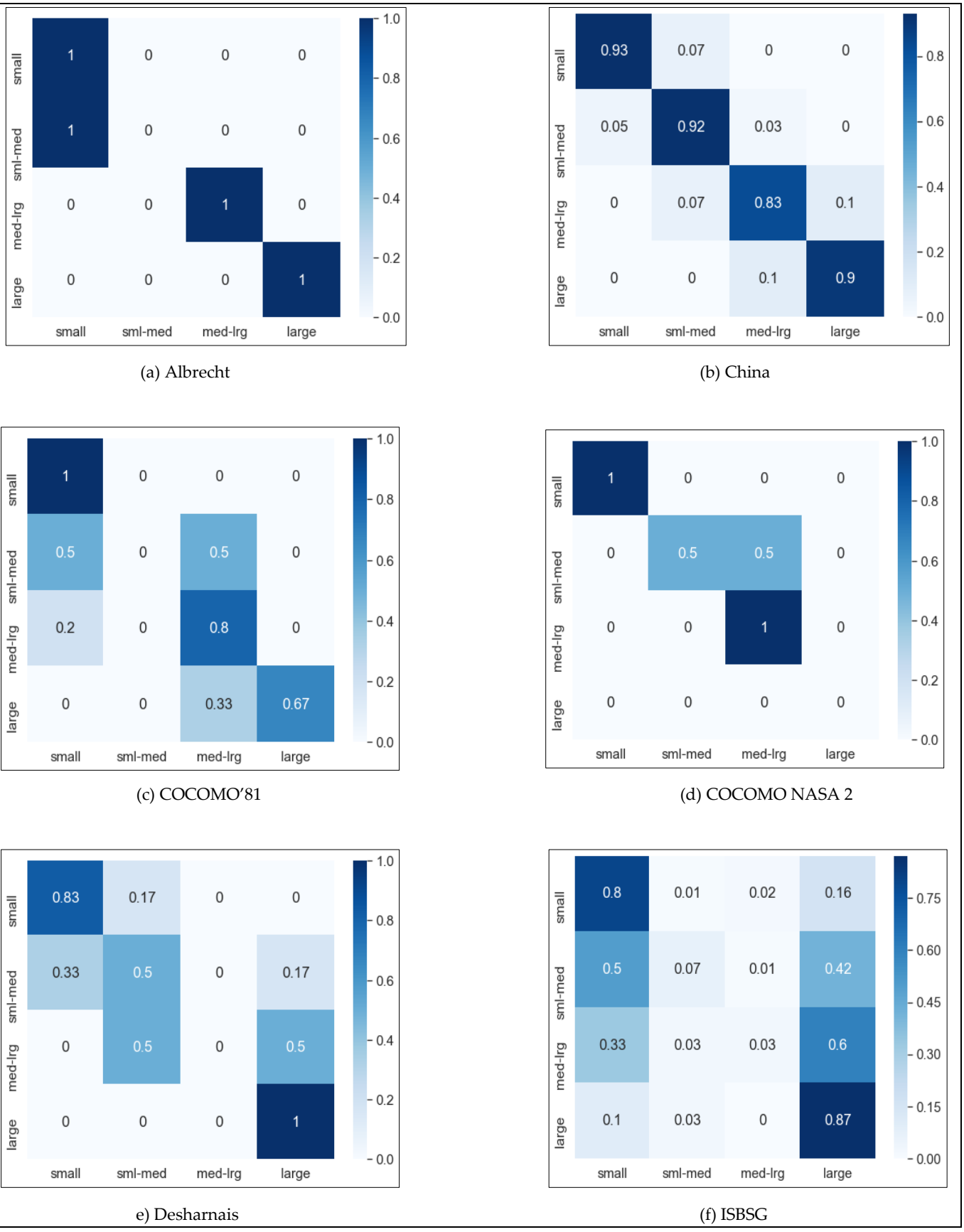


(a) Albrecht (b) China

(c) COCOMO'81 (d) COCOMO NASA 2

(e) Desharnais (f) ISBSG

Figure 2: ROC curves for the six data sets



(a) Albrecht

(b) China

(c) COCOMO'81

(d) COCOMO NASA 2

e) Desharnais

(f) ISBSG

Figure 3: Confusion matrices for the six data sets

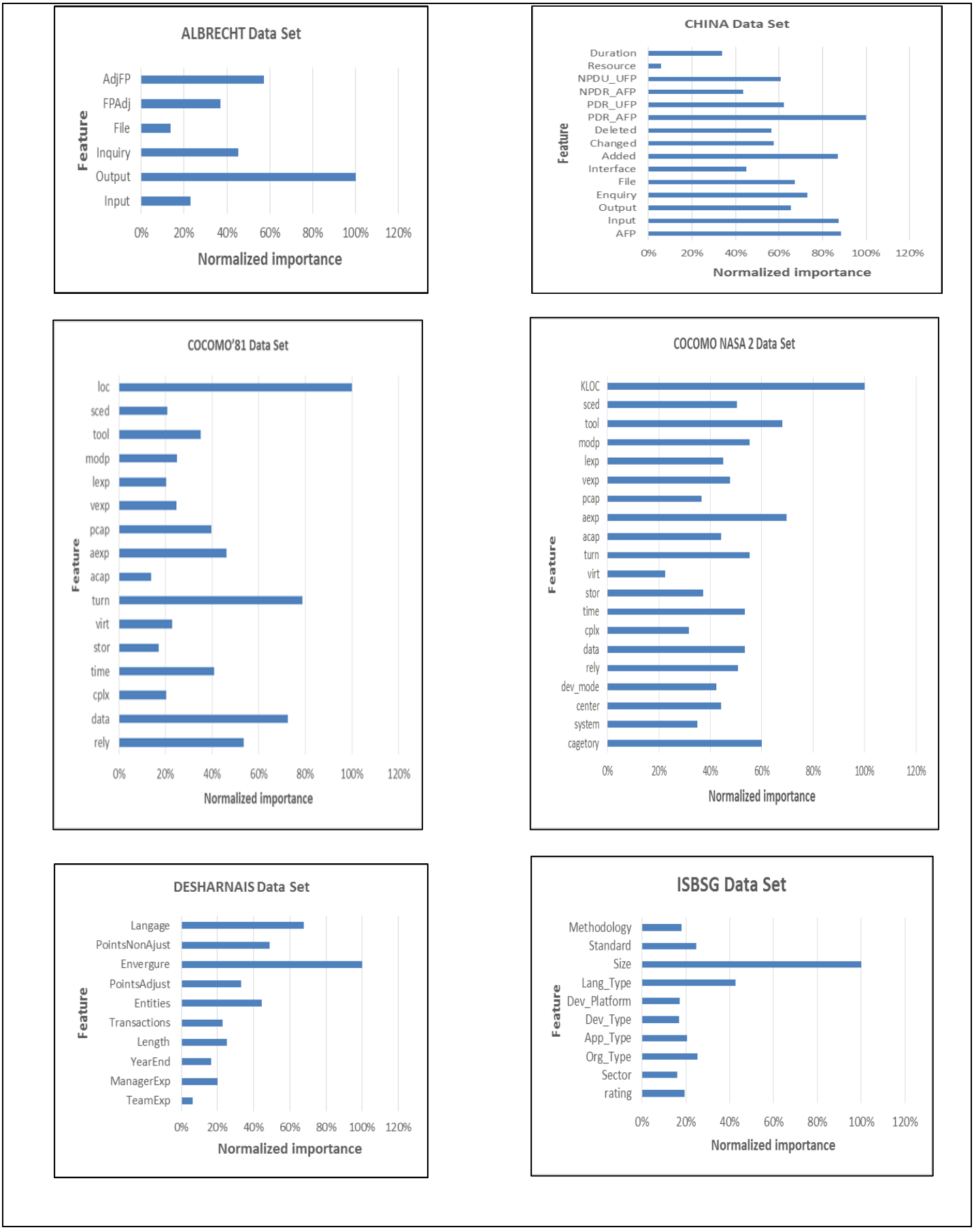


Figure 4: Feature importance across the six data sets

different types of data sets with no dramatic differences in results. However, the performance of MLP seems to decrease with data sets that are based on project characteristics, Desharnais and ISBSG.

4.4 Feature importance

For further analysis and to check the importance of features in building the classification model, we explain in figure 4 the features' importance and normalized importance across the six data sets.

In this figure, and for each data set, the normalized importance is calculated as follows: first, the importance of each feature is calculated as the relative importance of each feature in determining the classification model, with the sum of values across all the features is equal to (1.0).

Then, each feature importance value is divided on the maximum importance value across all features and the result is multiplied by 100% resulting by the normalized importance value.

In each data set, if we will consider the top two features in normalized importance as the most important features; it can be noted from figure 4 that managers should give more attention to the function points of output, the adjusted functions points, and the productivity delivery rate when estimating the required effort for the project that is based on function points.

From this figure, it can be noted that features like number of lines of code, and certain multipliers like, the turnaround time and application experience are important for projects that use COCOMO for its estimation.

Moreover, it can be noted that features like functional size, envergure, and language type are very important to be estimated right for a more accurate classification of a new projects' effort and based on project's characteristics.

5. CONCLUSION AND FUTURE WORK

From a managerial perspective, developing a software project requires three main stages: planning, implementation, and control. In the planning stage, the manager develops a project plan that identifies a roadmap for the development of successful project. Implementation and control stages are run in parallel to ensure that the software product is being built as it is planned to.

Proper planning is crucial for a successful project. A good plan requires accurate estimation of software effort. In the literature, there has been large number of approaches for estimating effort that are mainly based on 1) expert judgment, 2) function points and COCOMO models, and 3) machine learning approaches.

In the first approach, an expert's knowledge is applied to estimate the required effort for a new project and based on the expert's past experience with similar projects. In the function points' approach, a metric is applied for calculating the size of a new project in terms of function points and then the effort is calculated based on the knowledge of past projects with similar size. The COCOMO model calculates effort based on the development mode of the project and a set of multipliers. On the other hand, machine learning approaches attracted large attention in recent years for the purpose of effort estimation. Existing software projects' effort data is fed to a machine learning algorithm for the purpose of prediction of a new project software effort.

In this work, we investigated the problem of effort estimation from a classification perspective where the effort of a new project is assigned into one of four classes: small, small-medium, medium-large, and large. This is helpful for initial estimation of required time, resources and cost. This initial estimation gives project managers an indication of the required effort for the new project where a decision can be made to go further with detailed planning or not.

For this purpose, we examined six well-known data sets with three main categories; function points based data sets, COCOMO-based data sets, and project characteristics-based data sets. Three of these data sets are from PROMISE software engineering repository: COCOMO'81, COCOMO NASA 2, and Desharnais. The other three data sets are: Albrecht, China and ISBSG data set.

The MLP neural networks classification algorithm is implemented. MLP is implemented as it proved to be successful within this domain. The performance of the classifier is evaluated with hold-out test and using the accuracy measure where the number of correctly classified samples is divided by the total number of tested sample. For further analysis of the data, ROC curves and confusion matrices are used.

The accuracy results are encouraging with the highest reported accuracies are as follows; for the function points– based data sets, MLP provided accuracy close to 90.0% on China data set. For the COCOMO– based data sets, MLP provided an accuracy of 85.7% on the COCOMO NASA 2 data set. For the project characteristics -based data sets, MLP scored an accuracy of 70.0% on Desharnais data set. The ROC curves across the six data sets are generally good.

Overall, MLP performs well on different types of data sets with no dramatic differences in accuracy results. However, the performance of MLP seems to decrease with data sets that are based on project characteristics, Desharnais and ISBSG.

For further analysis of the data, we studied the importance of features in building the classification model and identified the important features that can contribute to better classification performance. It is noticed that features such as: the function points of output, the adjusted functions points, and the productivity delivery are important when estimating projects based on function points. On the other hand, it is noticed that features like number of lines of code, and certain multipliers like, the turnaround time and application experience are important for projects that use COCOMO for its estimation.
For the project's characteristics-based data sets, it can be noted that features like functional size, envergure, and language type are very important to be estimated right for a more accurate classification of a new project.

In the future, we plan to experiment with other neural networks types and compare its performance with that of MLP. Moreover, we plan to integrate an MLP with an ensemble learning algorithm for a possible enhanced performance.

REFERENCES

1. **Chaos Report**., 2015. The Standish Group International Inc.
2. Chandrasekaran, S., Gudlavalleti, S. and Kaniyar, S., 2014. **Achieving success in large complex software projects**. McKinsey & Company, pp.1-5.
3. Altuntas, T.U. and Alptekin, S.E., 2017. **Software Development Effort Estimation by Using Neural Networks-A Case Study**. International Journal of Computers, 2.
4. Minku, L.L. and Yao, X., 2011, September. **A principled evaluation of ensembles of learning machines for software effort estimation.** In Proceedings of the 7th International Conference on Predictive Models in Software Engineering (pp. 1-10). https://doi.org/10.1145/2020390.2020399
5. Mustapha, H. and Abdelwahed, N., 2019. **Investigating the use of random forest in software effort estimation**. Procedia computer science, 148, pp.343-352. https://doi.org/10.1016/j.procs.2019.01.042
6. Dejaeger, K., Verbeke, W., Martens, D. and Baesens, B., 2011. **Data mining techniques for software effort estimation: a comparative study**. IEEE transactions on software engineering, 38(2), pp.375-397. https://doi.org/10.1109/TSE.2011.55
7. Nassif, A.B., Azzeh, M., Capretz, L.F. and Ho, D., 2013, June. **A comparison between decision trees and decision tree forest models for software development effort estimation**. In 2013 Third International Conference on Communications and Information Technology (ICCIT) (pp. 220-224). IEEE.
8. Baskeles, B., Turhan, B. and Bener, A., 2007, November. **Software effort estimation using machine learning methods**. In 2007 22nd international symposium on computer and information sciences (pp. 1-6). IEEE. https://doi.org/10.1109/ISCIS.2007.4456863
9. Braga, P.L., Oliveira, A.L. and Meira, S.R., 2007, September. **Software effort estimation using machine learning techniques with robust confidence intervals**. In 7th international conference on hybrid intelligent systems (HIS 2007) (pp. 352-357). IEEE. https://doi.org/10.1109/ICHIS.2007.4344078
10. Kocaguneli, E., Kultur, Y. and Bener, A., 2009, November. **Combining multiple learners induced on multiple data sets for software effort prediction**. In 20th international symposium on software reliability engineering (ISSRE).
11. Zakrani, A., Hain, M. and Namir, A., 2018. **Software Development Effort Estimation Using Random Forests: An Empirical Study and Evaluation**. International Journal of Intelligent Engineering and Systems, 11(6), pp.300-311. https://doi.org/10.22266/ijies2018.1231.30
12. Shukla, S. and Kumar, S., 2019, July. **Applicability of Neural Network Based Models for Software Effort Estimation**. In 2019 IEEE World Congress on Services (SERVICES) (Vol. 2642, pp. 339-342). IEEE. https://doi.org/10.1109/SERVICES.2019.00094
13. Albrecht, A.J. and Gaffney, J.E., 1983. **Software function, source lines of code, and development effort prediction: a software science validation**. IEEE transactions on software engineering, (6), pp.639-648. https://doi.org/10.1109/TSE.1983.235271
14. Bohem, B., 1981. **Software Engineering Economics** 1981 Prentice Hall. Inc., Englewood Cliffs, New Jersey, 7632.
15. Boehm, B.W., Madachy, R. and Steece, B., 2000. **Software cost estimation with Cocomo II with Cdrom**. Prentice Hall PTR.
16. Schach, S.R., 2007. **Object-oriented and classical software engineering** (Vol. 6). New York: McGraw-Hill.
17. Al Asheeri, M.M. and Hammad, M., 2019, September. **Machine Learning Models for Software Cost Estimation.** In 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 1-6). IEEE. https://doi.org/10.1109/3ICT.2019.8910327
18. Rijwani, P. and Jain, S., 2016. **Enhanced software effort estimation using multi layered feed forward artificial neural network technique**. Procedia Computer Science, 89, pp.307-312.
19. Reddy, C.S. and Raju, K.V.S.V.N., 2009. **A concise neural network model for estimating software effort**. International Journal of Recent Trends in Engineering, 1(1), p.188.
20. Boetticher, G., Menzies, T., and Ostrand, T., 2007. **PROMISE repository of empirical software engineering data**.
21. Yun, F. H., 2010. **Effort estimation data sets**, http://doi.org/10.5281/zenodo.268446 .
22. Almaiah, M. A., 2020, **Multilayer neural network based on MIMO and channel estimation for impulsive noise environments in mobile wireless networks**, International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 315-321.

https://doi.org/10.30534/ijatcse/2020/48912020

23. Mercaral, J. D., Delima, A. P., Vilchez, R., 2020, **Prediction of Employees' lateness Determinants using Machine Learning Algorithms**, International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 779-783. https://doi.org/10.30534/ijatcse/2020/111912020
24. Nassif, A.B., Azzeh, M., Capretz, L.F. and Ho, D., 2016. **Neural network models for software development effort estimation: a comparative study**. Neural Computing and Applications, 27(8), pp.2369-2381. https://doi.org/10.1007/s00521-015-2127-1