USING STORY POINTS AND DECISION TREE TECHNIQUES TO ESTIMATE WORK AND COST IN AGILE SOFTWARE DEVELOPMENT

[1]MOHAMMED RAFI, [2]Dr. B.D.K. PATRO

[1]Research Scholar, Maharishi University of Information Technology, Lucknow, U.P. 226013, INDIA.
[2]Research Supervisor, Maharishi University of Information Technology, Lucknow, U.P. 226013, INDIA

KEYWORDS
estimation; software; effort; machine learning; time; cost; decision tree.

ABSTRACT
For an Information Technology (IT) project to be resource-efficiently planned, early effort estimate is crucial. On the other hand, not much study has been done on the subject of artificial intelligence-based effort estimate in software development that is agile. With the use of learning-oriented methodologies, this research project expands the use of hybrid models made up of algorithmic models as a project-level effort estimating methodology in agile frameworks. The story point technique is used in agile approaches like Scrum to estimate effort, which is an arithmetic calculation of the manpower needed to finish a system release. This study used labelled historical data to predict the project's total cost in Pakistani rupees (PKR) and its completion time in days. using AdaBoost, random forests, and decision trees to increase prediction accuracy. Ten-fold cross-validation was used to train the models, and the relative error was employed to compare the outcomes with those found in the literature. The best accuracy is obtained by the three approaches combined bootstrap aggregation (bagging) ensemble, together with project categorization, improves the results even further.

1 INTRODUCTION

A prominent career in software engineering is developing software systems by implementing policies and standards in place that try to guarantee that projects are completed within the allotted budget, time, scope, and quality. For a considerable amount of time, the most widely used approach for completing projects was the waterfall model, which is a sequential, linear model that begins with analysis and proceeds through planning, execution, evaluation, and deployment.[1]. Sequential lifecycle development is not the best approach since it requires upfront definition of all requirements, delays the delivery of functional software to users, and makes it difficult to adapt to needs changes. A group of software professionals developed an Agile Manifesto in 2001, which included a set of guidelines for new software development [2]. Scrum is one of the methodologies that emerged from agile development [3]. Scrum is an iterative method to project management that emphasises early and quick delivery of functional software as well as flexibility in response to requirements changes that arise throughout system development.

In such a case, it would be ideal to be able to provide a precise early estimate of the work and cost related to a product backlog connected to the release project. Using early effort estimates, the project manager can simulate many scenarios to determine if there are sufficient resources to complete the release project on time and under budget. Utilising tried-and-true methods for estimating work allows the team to make choices with a firm basis that have an immediate effect on the company[5]. But if neither an estimating model nor appropriate predictions are applied, this might lead to incorrect deadlines and subpar output.[6]

A size measure called story points is used to calculate the approximate amount of work needed to finish user stories. Before starting the process of work estimate at the team determines the narrative points required to complete each user story at the user story level. It is possible to predict the project delivery date and expenses using the total narrative points in a product backlog. Moreover, we can determine the team's velocity by adding up the story points for each user story that is allocated to a certain sprint. In software engineering, there are many methods for estimation. Expert judgement, analogy approaches, algorithmic models, and machine learning models are the categories into which these methods may be divided [7]. Planning poker is an estimating technique that Scrum usually encourages teams to utilise [8]. Each user story is estimated by the team using expert opinion. Further estimating rounds are carried out until convergence is achieved if the team members' estimations for a given user narrative do not converge. Although organising poker is a common strategy, it takes a lot of effort and is prone to biases and team dynamics. Therefore, it is preferable to carry out estimate utilising other methods that are not constrained by planning poker[9,10].

In this study, we looked at a method that combines a linear regression model—which forms the basis of the product backlog—with user stories, resulting in a dataset that contains every project built with this model[11]. Furthermore, each entry was given a label indicating the amount of time, effort, and money it required using a discretization of continuous features and supervised algorithms for machine learning that utilize the data to forecast the whole cost and time of completion. The proposal's main concept is to discretize data and combine several decision tree algorithms to produce an ensemble of multiple trained models that can provide an estimate and lower the averaged estimate's overall error[12]. The technique is hampered by the fact that we rely on the availability of high-quality data and have limited data. In addition to being useless in a changing environment, historical statistics reveal nothing about the experience and skill of the team[13].

The present study highlights the need for strong machine learning models and data discretization in order to provide reliable estimations and reduce prediction variation. The team gets a better notion of where the true values should be by using this to assist the model select modest intervals.

2 Related Work

Expert estimating methods like the Delphi method, the wideband Delphi methodology, Work Breakdown Structure (WBS), rule-based systems, and the top-down and bottom-up approach dominated traditional effort estimation.[14]. 1950 [15] saw the first of proposals using regression models. The 1970s and 1980s saw the introduction

of many industry-wide techniques, including[16]. These models have given academics access to invaluable datasets throughout time, enabling them to develop various machine learning-based strategies that provide more accurate predictions. While machine learning and analogy-based strategies rely on dependable databases and are readily applicable throughout the project's first stages, algorithmic techniques adhere to mathematical formulas [17-20].

Software engineering uses artificial intelligence (AI) technologies, such as machine learning, language processing, knowledge management, and machine vision, which are grouped based on their functionalities.. These categories are based on empirical evidence and are established in software engineering [21]. Machine learning was proposed as an additional category for effort estimating approaches in 2012 by Wen J. and colleagues [22].

With public datasets, machine learning algorithms, artificial neural networks, and soft computing, machine learning models are being proposed as an alternative to generalised models in a number of papers [11–17]. This category has been expanding. Mathematicians have done a great deal of work on decision tree analysis over the last 50 years, beginning with Frank P. Ramsey in 1931 [23]. Software engineering is one of the many disciplines in which it finds application. One of the primary factors contributing to Decision Trees' (DTs') appeal is its interpretability [24].

The emergence of agile methodologies gave birth to novel approaches and measurements, like narrative points and planning poker, which were introduced by Grenning in 2002 and are now often used to gauge the amount of work that has to be completed [24]. In their comprehensive literature analysis on agile software development, Fernandez-Diego et al. [25] said that when identifying a project's requirements, narrative points and an expert-based estimate known as planning poker is essential. Furthermore, the analysis demonstrates that neural networks, random forests, SVM, closest neighbours, and stochastic gradient boosting are the most often used data-based techniques[26].

Table 1. Condense the research that the algorithms are trained on.

Reference	Input	Output	Technique
[17]	The quantity of user stories, team speed, and sprint size Number of working days per month, team pay, and degree of estimation confidence	Effort, Time, Cost, Deceleration, Velocity, Friction Factors, and Dynamic Forces	Generalized Regression Neural Networks
[18]	Velocity_Effort	Time of Completion	Cascade-Correlation Neural Network (CCNN)
[19]	Velocity_Effort	Time of Completion, Total-Cost	Linear-based regression model

[20]	Velocity_Effort	Time of Completion	Decision Tree (DT)
[21]	Workdays, beginning frictional and dynamic variables, effort, and velocity	Time of Completion	Linear and multi variable models

2.1 Background

Depending on the unique needs and characteristics of the data we are dealing with, a broad range of methods from artificial intelligence and machine learning may be used to regression difficulties. The most often used techniques include deep learning approaches, DT, SVM, and many types of neural networks ridge and lasso regression, Bayesian regression, ensemble techniques, K-nearest neighbour, and Gaussian processes, among others.

DT is the method from the prior list that enables us to see the outcomes using a simple framework. Due to its capacity to manage nonlinear correlations between the goal variables and the input attributes, DT is often used in regression situations. Classification and regression tree research and applications are expanding quickly because to their easy interpretation, high prediction accuracy, and quick computing. Because they only need a few hyperparameters to be configured when implemented, trees are versatile and easy-to-use algorithms that perform well in scenarios when data is lacking[27].

2.2.1 Decision Tree

A Decision Tree, a hierarchical framework for supervised learning, is used to find the local region via a sequence of recursion splits in fewer steps. A decision tree's leaves represent the output values, while the central decision nodes evaluate a test function. This straightforward framework enables us to develop more reliable algorithms that raise estimate accuracy. As a result, when putting a DT into practice, a single configuration may be utilised to generate several techniques like random forests and AdaBoost.

2.2.2 Ada Boost

AdaBoost

AdaBoost [28] is a technique that combines many models of a weak learning algorithm to increase its accuracy. This approach creates a model in which each data item is assigned a weight; the incorrectly estimated data points are given larger weights, making them more significant when the subsequent weak base model is run. This is carried out up till the mistake is reduced.

2.2.3 Ensemble Learning

A simple definition of an ensemble is a collection of objects or people seen as a whole. An ensemble in machine learning is a collection of methods working together to increase accuracy [29-31]. By merging the outputs of the several algorithms that make up an ensemble, the results are improved [32]. Mixing combinations, mixing models, and mixing training data are the three categories of ensemble approaches. The first

kind trains different models on each subset after splitting the training data into various parts. Another name for this is bagging.

A variety of models, some of which are poor learners, are combined in the second kind of ensemble. The technique, which is often referred to as "boosting," emphasises poor performances more. Stacking is the process of performing machine learning models on top of each other's output. In this case, the input for the next levels is the predictions from the earlier layers. To get a better performance, the last kind the ensemble averages or votes the results together. It does this by applying many machine-learning techniques to the same dataset. Because it integrates the best aspects of many learning strategies into one output, it is a successful tactic[33].

Any ensemble may have some drawbacks, such as the ensemble's poor generalisation due to overfitting if the basis models have a tendency to overfit. Furthermore, as the ensemble becomes larger, the computational cost will go up as well since the model has to train every member of the ensemble before it can provide a single prediction. Because The data that is used to apply ensemble and algorithms for machine learning is always a determining factor., mixing data from different contexts will prevent the techniques from producing accurate estimations because the information will not remain consistent across software project changes or the application of different agile methodologies[34].

2.2.4 Random Forest

Through the combination of multiple models and their predictions, ensemble learning is a highly helpful strategy that lowers bias. The combination of n-base models reduces intended outcome, even if it demands more processing resources. One of the most popular methods is Random Forest (RF). A form of decision tree bagging, the random forest model's objective is to choose a random sample of features and see which ones provide the best results.

Table 2. Principal characteristics of tree-structured algorithms.

Algorithm	Features	Advantages
Decision Tree	creates a model that resembles a tree and divides the feature space into discontinuous areas.	Easy to understand, requires little computing power, manages nonlinear connections, can deal with missing data, and can deal with both both categorical and continuous variables
Random Forest	group of decision trees, with each tree constructed using a random selection of characteristics	estimates feature significance, manages high-dimensional data, is capable of handling noisy data, and use multiple tree averaging to reduce overfitting and variance.
Ensemble Learning	Many different model types' predictions are combined via ensemble learning to produce a	combines the advantages of many different models to reduce overfitting and boost generalisation. It

	single, more accurate forecast.	may be used in situations with limited data. Divide and conquer is the tactic used.
AdaBoost	fits the residuals of the previous learner iteratively to weak learners and merges them to create a powerful prediction	increases accuracy compared to decision trees or random forests by concentrating on misclassified data, managing noisy data, managing high-dimensional data, determining the importance of features, and being less susceptible to overfitting.

3 Materials and Methods

Three models are used to calculate effort and cost, and a mixture of many models is used to increase overall performance. AdaBoost, random forests, and decision trees served as the foundational learners. To get a final estimate, we averaged the predictions made by each model after it had been trained.

There were three steps to the procedure. The first section only involves prepping and, loading and labelling the original dataset using the discretization approach, taking only pertinent columns, and dividing all records into training and testing sets. All of the data are normalised after each subset has been formed. In order for training to take place, the second section configures the cross-validation and machine learning algorithms. Finally, evaluation measures are produced to determine the models' performance once they have been trained. Instead of training a single model, the multi-model technique uses the average of all the models that have been trained.

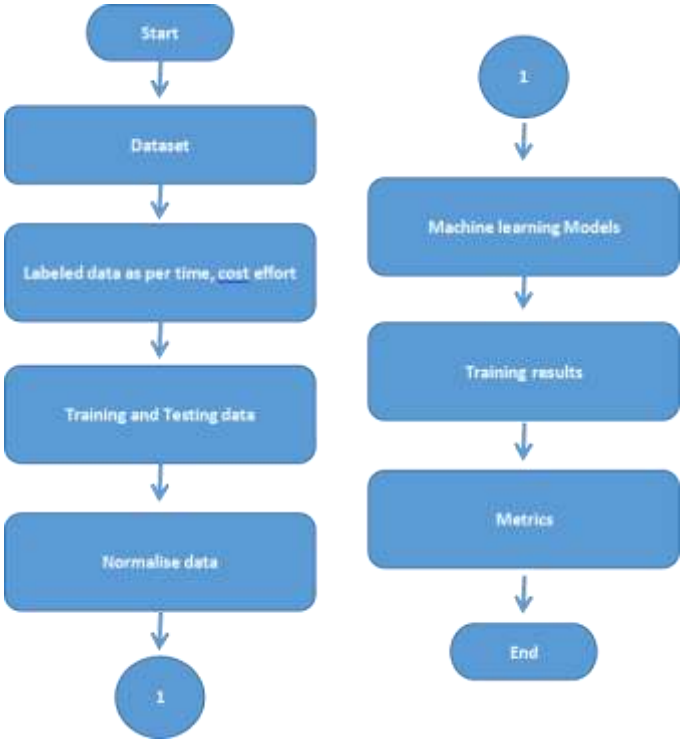

Figure 1. This project strategy's flowchart.

3.1. Proposed Approach

The three primary factors that need to be taken into account to guarantee the development's quality [35]. Since the historical statistics are derived from completed projects, when one of them is modified, the other two may also change. Nevertheless, overall effort remains unchanged in this scenario. Time and cost are the two dependent factors; they are reliant on both team velocity and overall effort.

The scope of the triangle represents the project's size and may be compared to the backlog of products. Once the group selects a specified scope, the project's progress forces us to travel from the top to the bottom of the triangle, lengthening the time and cost lines. The triangle's one side could change, lengthening or shortening its length, depending on the addition or subtraction of resources, modifications to the plans, and increase in project hours. The project will be of guaranteed quality if both parties are in balance.

The work required to finish The project features consist of the project using the narrative points metric, which indicates the scope. The complexity and size scales go from one to five, where one denotes a modest size and five denotes the largest value. The units of effort accomplished in a sprint determine the team velocity. Factors such as friction, forces, and deceleration feature might cause the project to slow down. Productivity and project velocity are decreased by friction and dynamic factors such changes in the makeup of the team, modifications to the procedures, introduction of new tools, personal problems, vague and evolving requirements, etc.

When the most important details are considered, the effort of a single tale is produced by the narrative's size and intricacy. There are five possible values for size and complexity: 1 is the smallest and 5 is a vast narrative with extreme complexity. The whole effort of all the individual tales makes up the effort of the entire endeavour.

One aspect that was assessed as the project progressed was velocity. But in order for the team leader to know the amount of points the team completes in a given sprint, the project has to be underway By examining past performance, velocity not only helps the team plan its next capacity more precisely, but it also indicates the amount of work that is really finished in each iteration [36-39]. Iteration and forecasting approaches, as well as historical data, may be used to estimate velocity [40].

In this instance, our estimate was derived from data pertaining to statistical support of previously completed software development projects. This method limits the exact use of machine learning models to the first sprint planning meetings; that is, when the project is already underway and the velocity is known based on the information available for it at this point in its development.

Table 3. A smaller dataset with category size labels, in which the input data is included in the remaining columns ActualCost are the columns that need to be forecast.

No	Effort	Vi	Actual-Time	Cost	Effort	Time
1	157	4.3	67	1300000	Large	Large
2	208	3.8	93	1700000	Medium	Medium
3	174	4.1	58	1250000	Small	Small
4	332	4.6	90	2500000	Medium	Medium
5	125	5.0	36	8520000	Large	Large
6	340	4.2	95	6352100	Large	Large
7	98	4.3	38	5420000	Medium	Medium
8	258	3.9	94	8500000	Large	Large
9	85	4.0	38	2512000	Small	Small

However, in the context of Scrum, story points serve as a meaningful indicator of effort. To estimate the labour required to create experience and access to historical data.[24]. Remember that every story point is an average of the amount of time the team will need to finish implementing a task. This makes it clear that the topic of discussion is a range of mathematical probability. A single narrative point, for instance, may correspond to four to twelve hours, two story points to ten to twenty hours, and so on.

3.1.1. Data Augmentation

Information scarcity is one issue with software work estimate. Companies often have few training examples since data collecting is costly or time-consuming. Synthetic project creation, based on finished projects, is a beneficial approach that should be big enough to be useful and generate new entries, however not so tiny as to compromise the project's core principles and essential qualities [41]. By incorporating a small quantity of Gaussian noise into training example properties, synthetic project creation is a practical and affordable method of enhancing prediction performance [42].

In order to prevent memorization of the tiny set and minimise overfitting, data augmentation is a regularisation technique that works well for training machine learning models. Along with regression and small data issues[83–87], the methodology directly modifies the amount of training samples and their distribution. Data augmentation may be used to expand the training dataset in order to overcome the problem of insufficient data.

We expanded the dataset by taking each project, adding two additional recordings, and increasing the effort and velocity using random noise values. We kept With a few tweaks, a tiny project would be little and a medium-sized project would be medium. We added noise in sufficient amounts to create new records while keeping the data intact.. The noise was produced by using the following equation:

$$Em = Ex(100+R)/100$$

where R is a random number between (−6, 6) excluding R = 0, Em is the project's original total effort, while Em is the modified effort. A similar equation was used to velocity, adding a random value up to 5% of the original characteristic. We are unable to ensure that a synthetic dataset will consistently be the same since data production is unpredictable. After the dataset was expanded, we were left with 63 project registers overall: 42 synthetic projects were put aside 70% of the projects were used for training, while 30% of the remaining 21 projects' actual values were used for testing.

The most accuracy based on an empirical design. making certain that the simplest and shortest architecture was used. Following a fold for each model instance, an iterative 10-fold cross-validation was used to individually train each method. Each algorithm generated 21 software project projections based on the test data. Because of this, the cross-validation procedure offers unique metrics for evaluation in both the training and validation phases. Based on the 21 predictions, we assessed the model's performance using test data. The final estimates regarding completion time and overall cost were obtained by adding together all of the assumptions.

3.1.2. Evaluation Criteria

The most widely used accuracy metrics are those that rely on the Mean Relative Error (MRE), Pred(x), the mean squared error (RMSE), the mean absolute error (MAE), and the accuracy percentage [25]. The sum of squared errors is how we evaluate the model's quality of fit when using Ordinary Least Squares (OLSs) [38]. The Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are two function mistakes that matter since our model is regression-based.

$$MSE = (y, \bar{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2$$

where y is the actual data, n is the number of samples, and yˆ is the forecast. Prediction accuracy is calculated using the Mean Relative Error (MRE), the bigger the better.

$$MRE = \frac{|y_i - \bar{y}_i|}{y_i}$$

$$Accuracy(\%) = (1 - MRE) \times 100$$

where yˆi = forecast and yi = reality. The coefficient of determination, which is ideally near to one, is:

$$R^2(y, \bar{y}) = 1 - \frac{\sum_i^n (y_i - \bar{y}_i)^2}{\sum_i^n (y_i - \bar{y}_i)^{2'}} where\{\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y^i$$

$$\exp lained - \text{var}\, iance(y, \bar{y}) = 1 - \frac{Var\{y - \bar{y}\}}{Var\{y\}}$$

$$MMRE = \frac{1}{n}\sum_{i-0}^{n-1} MRE_i$$

the number of observations is represented by n. Many studies in the literature compare the algorithm performance using the acceptable value of 25% for the MMRE; however, Not every article uses this as a standard measurement. The percentage of MRE in all projects which is less than or equal to l is known as Pred(l)

3.2. Dataset Discretization

The research carried out two examples that demonstrated how labelling data prior to training increases accuracy. As a result, the first scenario only used two parameters from the dataset: the team's velocity and the overall project effort expressed in narrative points. The second example combined these two elements includes three project sizes (small, medium, and big) and three budgetary, temporal, and effort categories. By doing this, projects are discretized into various sets based on their sizes, and once they are labelled, the columns are coded with numbers.

Project labelling is crucial since it gives the management the ability to establish the project's foundation. In terms of cost, the same can be said: three instances may be created for a single project in order to improve the model's depiction of the project's potential growth in terms of both time and money. Data may also be labelled by the kind of danger that might influence the development if sufficient information is given.

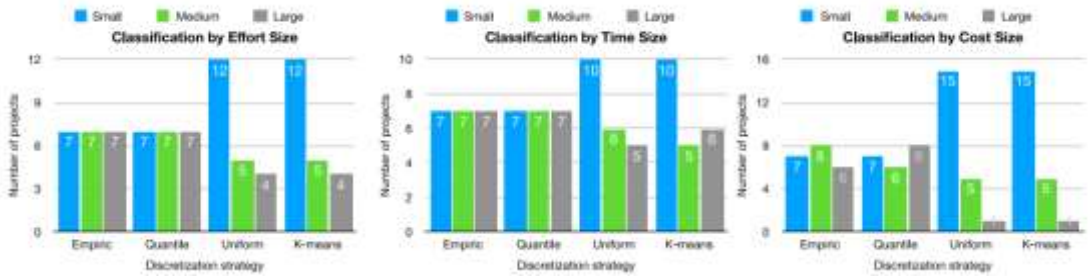

Figure 2. Different procedures were used for discretization, and the empirical approach produced results comparable to quantiles.

3.3. Coding Algorithms

Python and a number of open-source libraries were used to develop this project. These libraries allowed for the management of data, the implementation of machine learning algorithms, the execution of training, and the evaluation of the results using error functions. Numpy and Pandas were used for data processing, scikit-learn was used for model building and cross-validation, and Since Google Colab allows online work without needing the installation of required tools locally, the free edition was used for coding. The workspace includes two 2.20 GHz Intel(R) Xeon(R) CPUs, 8 GB of RAM, 512 GB of storage, and a GPU at 2500 MHz for free.

All of the algorithm's hyperparameters that aren't explicitly changed get their default values from the API. Smaller architectures are the foundation of algorithm creation, which expands until results accuracy no longer rises noticeably. A class called DecisionTreeRegressor, which implements decision trees for regression, is found in code Listing 1. The random forest and AdaBoost algorithms are built on top of this. The maximum depth of the tree is indicated by max_depth, the lowest number of samples needed to split an internal node is indicated by min_samples_split, and the randomness of the estimator is controlled by random_state. The method begins to overfit the data if max_depth is higher than five. The squared error is the default function used to assess a split's quality. RegrDT was used as the estimator by regrRF, regrAda, and regrDT after it was configured.

regrAda = AdaBoostRegressor(n_estimators = 9, random_state = 10, regrDT)

4. Results

Data preparation involves scaling in the interval [0, 1] (normalisation) and labelling before to training. Because the trials are carried out individually, estimates are obtained from the decision tree, random forest, and AdaBoost in order of precedence. Every method used identical data and cross-validation configurations. Due to the two runs of 10-fold cross-validation during training, nine trained models were obtained. Each trained model yields a coefficient of determination of at least 0.90 and an accuracy of more than 80%.

After the model is trained, fresh estimates are produced by feeding it with the set of ten actual projects as input. The final ensemble will be constructed using the trained model's final predictions from this testing set. The following metrics were used to evaluate the training, validation, and testing phases: variance, accuracy, mean squared error, root of mean squared error, coefficient of determination, and mean relative error. For every strategy, tables were created that grouped the data for each stage.

4.2. Multi-Model Experiments

The RF method, consisting of 10 separate trees, is the first multi-model test. Two examples were analysed again for evaluating model estimates. The MMRE drops when data is labelled in relation to cost and time estimates. In terms of time and cost projections, Pred(l) performs less well than DT, suggesting a decline in RF efficiency.

Table 4. Metrics for RF assessment are compared.

	Completion Time		Total Cost	
Criteria	Unlabeled Data	Labeled Data	Unlabeled Data	Labeled Data
MMRE	0.0489	0.0385	0.0564	0.06121
MdMRE	0.0562	0.0369	0.0321	0.1263
Pred(MMRE)	58.32	62.35	63.021	62.39
Pred(l)	81.23	93.25	79.23	81.23
Pred(0.25)	110	110	110	110

In order to generate a higher-performing model, Several machine learning models are combined in the mixing models technique, and each prediction is then totaled together to provide a single outcome [36]. Many techniques, including hyperparameter tuning ensembles, may be used. Since we are working with a numerical regression issue in this instance, averaging is helpful. Additionally, blending models is a wise decision since various machine-learning approaches are assessed.

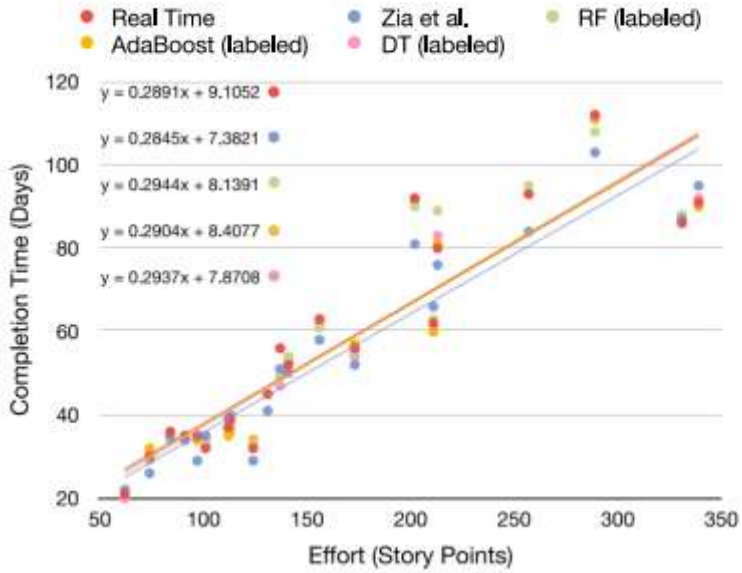

Figure 3: Completion time with effort

The final findings together with their standard deviation. The table presents the actual time and cost numbers, ensemble estimates together together with respect to cost estimations, the MLP yields findings with a relative error of 0.0748, a coefficient of determination of 0.9341, and an accuracy of 92.52%. Even under the same conditions, the decision tree's depth makes it more susceptible to overfitting in relation to the MLP. However, once the ensemble is applied, The single-model's overfitting is reduced. This is particularly true for the total cost and completion time, where the decision tree produced a better accuracy and smaller relative error.

5 Discussion

This paper used an ensemble of the random forest, decision tree, and AdaBoost algorithms to estimate the agile software development effort. The dependent variables include completion time and total cost, whereas pace and effort—measured in story points—are the independent components. When comparing the preclassification results, accuracy increased and error dropped. results for data that was tagged with

data that was not. Project labelling improved the learning process of the machine learning algorithms. The standard deviation of the completion time estimates in most projects is less than three business days, as can be seen when calculating the disparity between actual and projected values. Among the twenty-one projects, only two had expenses that were more than PKR 100,000 when the cost difference was considered. The whole cost's standard deviation was kept below PKR 90,000. These results are considered trustworthy since all of the approaches approach a particular trend for the dataset, the regression lines are quite close to one another, and the data dispersion is not very large.

According to the data, the RF algorithm has the lowest accuracy, with a 96.92% accuracy but a R 2 of 0.9869. In both the completion time estimate and the overall cost estimate, AdaBoost performs better than the competition. While none of them works well in terms of cost and time, the combination of the three produces trustworthy outcomes. Parameters are crucial while creating the model to prevent the DT from overfitting the data, which is a problem that arises during the training process. Machine learning algorithms can estimate the function that corresponds to the software projects, as seen by the accuracy values being close to 1 and high, and the forecasts being close to the actual values.

6 Conclusions

The resources, the scope, and the completion date are the three main factors to consider while creating a software project. Ensuring development quality is achieved by meeting customer expectations, controlling overhead costs, and completing projects on time. The study has important practical implications, such as the requirement for the team to use some degree of professional judgment when deciding how many and how complex user stories to include in the project scope. Methodologies, such as supervised learning, to predict the project's length and cost is viable. This technique might assist a novice user make better project management decisions. Because a model using machine learning can generalize information gathered from a product backlog and team qualities like velocity and provide consistent results, the hybrid model described in this research becomes a useful tool for story point approach project planning. In a real-world context, the idea helps the project management group schedule releases.

When the models are uncorrelated, ensemble approaches perform optimally. The likelihood that the assembly algorithm will enhance the performance of its constituent parts increases with the variety of the models that comprise the assembly. One method is used in homogeneous ensemble learning; in this instance, A decision tree served as the foundation learner for both AdaBoost and the random forest. Heterogeneous ensemble learning makes use of several base learners; the final ensemble is composed of AdaBoost, a random forest, and a single decision tree. decreasing bias can be achieved by boosting approaches, whereas decreasing variance may be achieved through bagging techniques.
Further investigation is necessary to address the short dataset issue via discretization and data augmentation. It is necessary to research unsupervised machine learning approaches in order to identify the algorithms that might contribute to even higher prediction accuracy..

Conflicts of Interest: No conflicts of interest are disclosed by the writers. The choice to publish the findings, write the publication, collect, analyse, or interpret data, or plan the research were all made without the funders' input.

References

1. Wysocki, R.K. Effective Project Management: Traditional, Agile, Hybrid, Extreme; Wiley: Hoboken, NJ, USA, 2019.
2. Hohl, P.; Klünder, J.; van Bennekum, A.; Lockard, R.; Gifford, J.; Münch, J.; Stupperich, M.; Schneider, K. Back to the future: Origins and directions of the 'Agile Manifesto'—Views of the originators. J. Softw. Eng. Res. Dev. 2018, 6, 1–27. [CrossRef]
3. Sommerville, I. Software Engineering, 10th ed.; Pearson Education: London, UK, 2019.
4. Vyas, M.; Bohra, A.; Lamba, D.C.S.; Vyas, A. A Review on Software Cost and Effort Estimation Techniques for Agile Development Process. Int. J. Recent Res. Asp. 2018, 5, 1–5. ISSN: 2349-7688.
5. Mahniˇc, V.; Hovelja, T. On using planning poker for estimating user stories. J. Syst. Softw. 2012, 85, 2086–2095. [CrossRef]
6. Rashid, J.; Nisar, M.W.; Mahmood, T.; Rehman, A.; Syed, Y.A. A study of software development cost estimation techniques and models. Mehran Univ. Res. J. Eng. Technol. 2020, 39, 413–431. [CrossRef]
7. Fedotova, O.; Teixeira, L.; Alvelos, A.H. Software effort estimation with multiple linear regression: Review and practical application. J. Inf. Sci. Eng. 2013, 29, 925–945.
8. Sharma, B.; Purohit, R. Review of current software estimation techniques. In Data Science and Analytics: 4th International Conference on Recent Developments in Science, Engineering and Technology; Springer: Singapore, 2018.
9. Hoc, H.T.; Hai, V.V.; Nhung, H.L.T.K. A Review of the Regression Models Applicable to Software Project Effort Estimation. Comput. Stat. Math. Model. Methods Intell. Syst. Adv. Intell. Syst. Comput. 2019, 2, 399–407.
10. Barenkamp, M.; Rebstadt, J.; Thomas, O. Applications of AI in Classical Software Engineering. AI Perspect. 2020, 2, 1. [CrossRef]
11. Hidmi, O.; Sakar, B.E. Software Development Effort Estimation Using Ensemble Machine Learning. Int. J. Comput. Commun. Instrum. Eng. 2017, 4, 143–147.
12. Ziauddin, K.Z.K.; Tipu, S.K.; Zia, S. An Intelligent Software Effort Estimation System. J. Expert Syst. (JES) 2012, 1, 4. ISSN 2169-3064.
13. Khan, M.W.; Qureshi, I. Neural Network based Software Effort Estimation: A Survey. Int. J. Adv. Netw. Appl. 2014, 5, 1990–1995.
14. Abnane, I.; Hosni, M.; Idri, A.; Abran, A. Analogy Software Effort Estimation Using Ensemble KNN Imputation. In Proceedings of the 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea-Chalkidiki, Greece,
28–30 August 2019.
15. Kumar, P.S.; Behera, H.S.; Nayak, J.; Naik, B. A pragmatic ensemble learning approach for effective software effort estimation. Innov. Syst. Softw. Eng. 2021, 18, 283–299. [CrossRef]
16. Kumar, P.S.; Behera, H.; Nayak, A.K.K.J.; Naik, B. Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. Comput. Sci. Rev. 2020, 38, 100288. [CrossRef]

17. Hussein, L.A.; Nassar, K.A.; Naser, M.A.U. Recurrent Neural Network based Prediction of Software Effort. Int. J. Comput. Appl.2017, 177, 8887. [CrossRef]
18. Mittal, K.; Khanduja, D.; Tewari, P.C. An insight into decision tree analysis. World Wide J. Multidiscip. Res. Dev. 2017, 3, 111–115.
19. Loh, W.-Y. Fifty years of classification and regression trees. Int. Stat. Rev. 2014, 82, 329–348. [CrossRef]
20. Prasadu Peddi, & Dr. Akash Saxena. (2016). STUDYING DATA MINING TOOLS AND TECHNIQUES FOR PREDICTING STUDENT PERFORMANCE. *International Journal Of Advance Research And Innovative Ideas In Education*, 2(2), 1959-1967.
21. Nassif, A.B.; Azzeh, M.; Capretz, L.F.; Ho, D. A comparison between decision trees and decision tree forest models for software development effort estimation. In Proceedings of the 2013 Third International Conference on Communications and Information
Technology (ICCIT), Beirut, Lebanon, 19–21 June 2013.
22. Srinivasan, K.; Fisher, D. Machine learning approaches to estimating software development effort. IEEE Trans. Softw. Eng. 1995, 21, 126–137. [CrossRef]
23. Najm, A.; Zakrani, A.; Marzak, A. Decision trees based software development effort estimation: A systematic mapping study. In Proceedings of the 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco,
22–24 July 2019.
24. Coelho, E.; Basu, A. Effort Estimation in Agile Software Development using Story Points. Int. J. Appl. Inf. Syst. 2012, 3, 7–10. [CrossRef]
25. Fernandez-Diego, M.; Mendez, E.R.; Gonzalez-Ladron-De-Guevara, F.; Abrahao, S.; Insfran, E. An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. IEEE Access 2020, 8, 166768–166800. [CrossRef]
26. Dave, C.V. Estimation approaches of machine learning in scrum projects: A Review. Int. J. Res. Appl. Sci. Eng. Technol. 2021, 9, 1110–1118. [CrossRef]
27. Sudarmaningtyas, P.; Mohamed, R. A review article on software effort estimation in agile methodology. Pertanika J. Sci. Technol. 2021, 29. [CrossRef]
28. Mahmood, Y.; Kama, N.; Azmi, A. A systematic review of studies on use case points and expert-based estimation of software development effort. J. Softw. Evol. Process. 2020, 32, 7. [CrossRef]
29. Horgan, G.; Khaddaj, S.; Forte, P. Construction of an FPA-type metric for early lifecycle estimation. Inf. Softw. Technol. 1998, 40, 409–415. [CrossRef]
30. Giray, G. A software engineering perspective on Engineering Machine Learning Systems: State of the art and Challenges. J. Syst.Softw. 2021, 180, 111031. [CrossRef]
31. Ziauddin, S.K.T.; Zia, S. An Effort Estimation Model for Agile Software Development. Adv. Comput. Sci. Its Appl. 2012, 2, 314–324.
32. Popli, R.; Chauhan, N. Cost and effort estimation in agile software development. In Proceedings of the 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, India, 6–8 February 2014; pp. 57–61.[CrossRef]
33. Raslan, A.T.; Darwish, N.R. Effort Estimation in Agile Software Projects using Fuzzy Logic and Story Points. In Proceedings of the 50th Annual Conference on Statistics, Computer Sciences, and Operation Research, Cairo, Egypt, 27–30 December 2015; pp. 27–30.
34. Choudhari, J.; Suman, U. Story Points Based Effort Estimation Model for Software Maintenance. Procedia Technol. 2012, 4, 761–765. [CrossRef]

35. Scott, E.; Pfahl, D. Using developers features to estimate story points. In Proceedings of the 2018 International Conference on Software and System Process, Gothenburg, Sweden, 26–27 May 2018.
36. Malgonde, O.; Chari, K. An ensemble-based model for predicting agile software development effort. Empir. Softw. Eng. 2018, 24, 1017–1055. [CrossRef]
37. Prasadi Peddi and Dr. Akash Saxena (2015), "The Adoption of a Big Data and Extensive Multi-Labled Gradient Boosting System for Student Activity Analysis", International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 3, Issue 7, pp:68-73.
38. Durán, M.; Juárez-Ramírez, R.; Jiménez, S.; Tona, C. User Story Estimation Based on the Complexity Decomposition Using Bayesian Networks. Program. Comput. Softw. 2020, 46, 569–583. [CrossRef]
39. Gultekin, M.; Kalipsiz, O. Story Point-Based Effort Estimation Model with Machine Learning Techniques. Int. J. Softw. Eng. Knowl. Eng. 2020, 30, 43–66. [CrossRef]
40. Adnan, M.; Afzal, M. Ontology Based Multiagent Effort Estimation System for Scrum Agile Method. IEEE Access 2017, 5, 25993–26005. [CrossRef]
41. Sembhoo, A.; Gobin-Rahimbux, B. A SLR on Deep Learning Models Based on Textual Information for Effort Estimation in Scrum. 2023. Available online: https://www.researchsquare.com/article/rs-2461583/latest.pdf (accessed on 30 January 2023).
42. Choetkiertikul, M.; Dam, H.K.; Tran, T.; Pham, T.; Ghose, A.; Menzies, T. A Deep Learning Model for Estimating Story Points.IEEE Trans. Softw. Eng. 2019, 45, 637–656. [CrossRef]