JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING

Mr. K. Sudhakar (Assistant Professor)
Department of Information Technology
Vishnu Institute of Technology
Bhimavaram, Andhra Pradesh, India

G. Harshini (Student)
Department of Information Technology
Vishnu Institute of Technology
Bhimavaram, Andhra Pradesh, India

G. Dharma Teja (Student)
Department of Information Technology
Vishnu Institute of Technology
Bhimavaram, Andhra Pradesh, India

CH. Shiva Shankar Reddy (Student)
Department of Information Technology
Vishnu Institute of Technology
Bhimavaram, Andhra Pradesh, India

CH. Rakesh (Student)
Department of Information Technology
Vishnu Institute of Technology
Bhimavaram, Andhra Pradesh, India

Abstract: Software effort estimation is a vital component of project management, playing a critical role in determining the success of software development projects. Accurate predictions can significantly impact project outcomes, enabling project managers to plan and track the development of software systems more effectively. By providing reliable estimates of the time and resources required to complete a project, software effort estimation helps project managers make informed decisions, allocate resources efficiently, and mitigate potential risks. This study proposes a machine learning-based approach for software effort estimation, leveraging the strengths of multiple algorithms and datasets, including ISBSG, NASA 93, and Desharnais, to improve prediction accuracy. Our approach employs an ensemble of machine learning algorithms, including Support Vector Machine (SVM), Linear Regression, Random Forest, and Decision Tree. These algorithms work in conjunction to model complex relationships between features and effort, handle non-linear relationships and feature interactions, and extract latent features, ultimately improving model interpretability. Our results demonstrate significant improvements in estimation accuracy compared to traditional methods, highlighting the potential of machine learning-based approaches in software effort estimation.

Keywords: Software effort estimation, Machine learning, Project management, ISBSG, NASA 93, Desharnais, Support Vector Machine (SVM), Linear Regression, Random Forest, Decision Tree.

Introduction:

Software development projects are inherently complex and uncertain, making accurate effort estimation a daunting task. Traditional estimation methods, such as expert judgment and simplistic models, often fall short in providing reliable predictions. This can lead to project delays, cost overruns, and decreased quality. In recent years, machine learning has emerged as a promising approach for software effort estimation. By leveraging large datasets and advanced algorithms, machine learning models can learn complex relationships between project features and effort, providing more accurate predictions. Software companies produce a lot of data during software development. This data is generated at each stage, from requirements to maintenance. To manage software development projects effectively, companies need to consider key factors such as Lines of Code (LOC), Past project data, Team skills and size, Project requirements, Timelines. These factors help teams deliver software solutions that meet stakeholder expectations and project goals. Software companies collect and analyse data from software repositories to improve software quality. Data mining techniques are used to uncover patterns and trends. However, many software projects face challenges. Surveys show that:

- Two-thirds of large projects exceed their original projections.
- One-third of projects go over budget and are delivered late.

To address these challenges, software engineering cost estimation is crucial. This involves estimating the effort and resources needed to develop a software system. Factors that influence software project costs include:

- System size and complexity
- Development methodologies and tools
- Team skill level

Cost estimation is an iterative process that needs to be refined and updated throughout the project. Effort estimation is also critical in software engineering. It involves predicting the human resources needed to complete a project. Various effort estimating techniques are available, each with advantages and disadvantages. Popular techniques include:

- Parametric modelling
- Expert judgment
- Machine learning algorithms

Methods for estimation: Various methods have been proposed for software effort estimation, each with its strengths and weaknesses. Parametric models, such as COCOMO (Constructive Cost Model) and SLIM (Software Lifecycle Management), estimate effort based on mathematical equations that relate effort to project characteristics like size, complexity, and team experience. Non-parametric models, like machine learning algorithms, learn patterns in historical data to estimate effort. Analogous estimation involves identifying similar past projects and using their effort data to estimate the effort required for the current project. Expert judgment relies on the experience and intuition of experts to estimate effort. Top-down estimation involves breaking down the project into smaller tasks and estimating the effort required for each task, while bottom-up estimation involves estimating the effort required for each individual component or feature of the project.

- There are several ways to estimate the effort needed for a software project:

1**. Expert Judgment:** Ask experienced people who worked on similar projects. This method provides a rough estimate, but can be biased.

2. **Analogous Estimation:** Use data from similar past projects. This method is quick and easy, but may not be accurate if projects are different.

3. **Three-Point Estimation:** Estimate the most likely, optimistic, and pessimistic effort. This method generates a range of possible values, but can be complex.

4. **Parametric Estimation:** Use mathematical models based on project size, complexity, and other factors. This method provides a detailed estimate, but requires more data.

Accurate estimation is vital in project management. Inaccurate estimates can lead to project failure. In fact, poor estimation techniques cause about 65% of project failures. This study explores using machine learning to improve software cost estimation. We aim to compare machine learning techniques with traditional methods to determine which is more effective. We'll use various machine learning algorithms, such as vector regression and decision tree-based regression, and analyse the results. This paper is organized into several sections. We'll review previous studies on software cost estimation, discuss machine learning methods, and explain our experimental approach. We'll then present and analyse our results, followed by a conclusion and suggestions for future work.

- Our goal is to evaluate and compare different prediction systems, not to promote a specific technique.

Related work:

Software effort estimation has been an active area of research for several decades. Traditional methods, such as COCOMO (Boehm, 1981) and Function Point Analysis (Albrecht, 1979), rely on simplistic models and expert judgment, which can lead to inaccurate predictions. In recent years, machine learning has emerged as a promising approach for software effort estimation. Several studies have applied machine learning algorithms, such as neural networks (Huang et al., 2018), decision trees (Kaur et al., 2018), and random forests (Elish et al., 2018), to software effort estimation. Some studies have also explored the use of ensemble methods, which combine the predictions of multiple machine learning models. For example, Xia et al. (2019) proposed an ensemble method that combines the predictions of neural networks and decision trees. Other studies have investigated the use of feature selection and feature engineering techniques to improve the accuracy of software effort estimation models. For example, Singh et al. (2018) proposed a feature selection method based on mutual information, which was used to select the most relevant features for software effort estimation. Despite the promising results of these studies, there is still a need for further research in this area. In particular, there is a need for more studies that investigate the use of ensemble methods and feature selection techniques in software effort estimation.

Estimating Software Costs: Estimating the cost of software development is a crucial step in project planning. Many researchers have worked on creating models to help estimate these costs.

Why Do We Need Different Models?

1. Improve existing models: New models can fix weaknesses in older models.

2. Find alternative approaches: New models can offer different ways to estimate costs.

3. Support existing models: New models can confirm or validate existing models.

What Is the Build Cost Model?

The Build Cost Model is a well-known method for estimating software costs. This model helps developers predict the costs of building software based on factors like size, complexity, and resources required.

Without Machine Learning:

The COCOMO Model: The Constructive Cost Model (COCOMO) is a widely used method for estimating software development costs. Developed by Barry Boehm, COCOMO uses equations and parameters based on past software projects to estimate effort and cost.

How COCOMO Works?
COCOMO measures code size in Thousand Lines of Code (KLOC) and effort in person-months. This model helps estimate the quality and effort required for software projects.

Improving COCOMO: Researchers have enhanced COCOMO by incorporating new techniques, such as:
1. Fuzzy-COCOMO: This approach uses the Gaussian Membership Function to reduce relative errors.
2. Hybrid BATGSA algorithm: This method optimizes COCOMO using data from NASA databases.
3. Fuzzy clustering, ABE, and ANN approaches: This innovative strategy combines techniques to improve effort estimation accuracy.

Results: These enhancements have led to significant improvements, including:
1. Reduced normalized error
2. Improved prediction accuracy (up to 94% gain)
3. Better estimation of effort and cost
Overall, COCOMO remains a valuable tool for software project estimation, and ongoing research continues to enhance its accuracy and effectiveness.

Machine Learning:

Machine Learning Algorithms for Effort Prediction:
This section discusses various machine learning algorithms used to predict software project effort. Machine learning techniques are becoming increasingly important in research due to their reliable results.

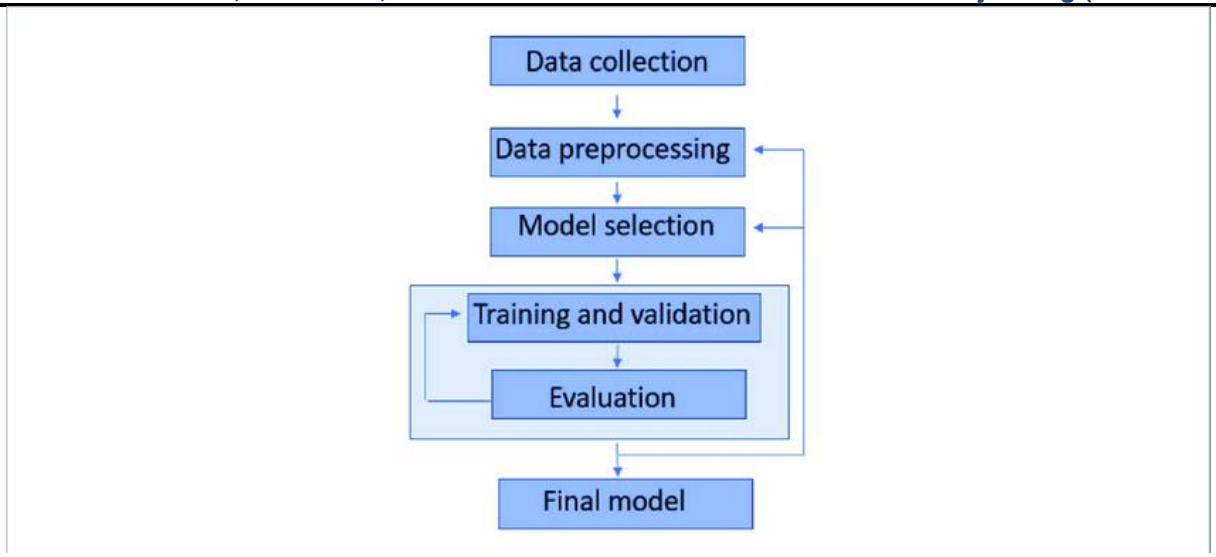
- Reliable Results with ML Approaches: Studies have consistently shown that machine learning approaches provide reliable results. In one study, a team used two machine learning methods to forecast the software product life cycle, resulting in a more accurate and adaptable cost estimation model.

- Applications of ML Algorithms: Machine learning algorithms have various applications, including:
  1. Support Vector Machines (SVMs): Used to distinguish between different types of breast cancers.
  2. K-Nearest Neighbours (KNN): Used for classification tasks.
  3. Neural Networks: Used for classification and regression tasks.

- Environmental Factors in Cost Assessment: Other studies have focused on environmental factors that influence cost assessment, such as the software development life cycle. For example, a 2018 study by Rahikkala et al. explored the impact of organizational factors on software cost planning.

Key Points:
- Machine learning algorithms provide reliable results in effort prediction.
- Various algorithms, such as SVMs, KNN, and neural networks, have different applications.
- Environmental factors, like the software development life cycle, influence cost assessment.

Proposed System Workflow: The proposed system workflow for software effort estimation involves a structured approach to develop and deploy an accurate estimation model. The workflow commences with data collection and preprocessing, where relevant datasets are gathered, cleaned, and transformed into suitable formats. Next, model preparation involves selecting suitable machine learning algorithms, tuning hyperparameters, and training models on the pre-processed data. The trained models are then evaluated and compared using performance metrics, and the best-performing model is selected for deployment. Finally, the selected model is tested, deployed, and maintained, with ongoing monitoring and retraining as necessary to ensure optimal performance and provide accurate software effort estimates.



Phase 1: Data Collection and Preprocessing

1. Collect datasets: ISBSG, NASA 93, Desharnais, and other relevant datasets.
2. Data cleaning: Handle missing values, outliers, and inconsistent data.
3. Data transformation: Normalize/scale data to suitable ranges for modelling.
4. Feature selection: Identify relevant features that impact software effort estimation.
5. Data splitting: Divide data into training (70%), validation (15%), and testing (15%) sets.

Phase 2: Model Preparation

1. Select algorithms: Choose suitable machine learning algorithms (e.g., Linear Regression, Decision Trees, Random Forest, SVM).
2. Hyperparameter tuning: Optimize algorithm hyperparameters using cross-validation.
3. Model training: Train models on the training dataset.
4. Model evaluation: Evaluate model performance on the validation dataset.

Phase 3: Model Evaluation and Selection

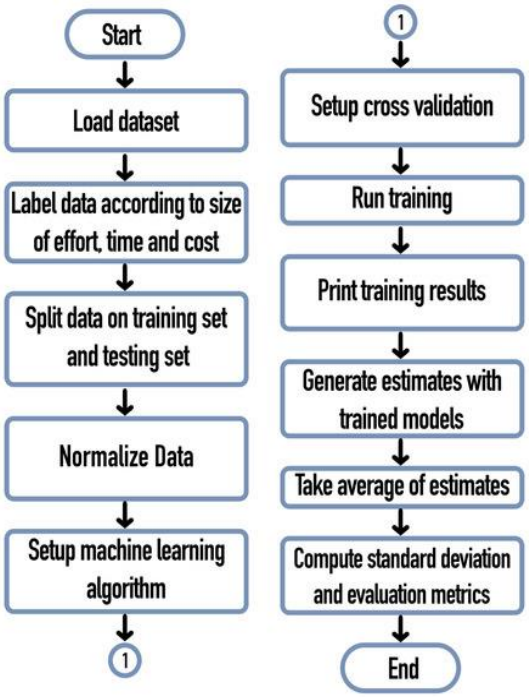
1. Performance metrics: Calculate metrics (e.g., MMRE, MAE, RMSE) to evaluate model performance.
2. Model comparison: Compare performance of different models.
3. Model selection: Select the best-performing model.

Phase 4: Testing and Deployment

1. Model testing: Evaluate the selected model on the testing dataset.
2. Model deployment: Deploy the selected model in a suitable environment (e.g., web application, API).
3. Model maintenance: Monitor model performance, retrain as necessary.

Phase 5: Result Analysis and Reporting

1. Result analysis: Analyse the performance of the selected model.
2. Report generation: Generate reports highlighting the results, including performance metrics and visualizations.
3. Conclusion and recommendations: Draw conclusions and provide recommendations for future improvements.



Methodology: This study employs a quantitative research approach, utilizing a combination of machine learning algorithms and statistical techniques to develop an accurate software effort estimation model. The methodology involves a systematic process, commencing with data collection from publicly available datasets, including ISBSG, NASA 93, and Desharnais. Data preprocessing techniques, such as handling missing values, outliers, and data normalization, are applied to ensure data quality. Next, feature selection and engineering techniques are employed to identify relevant variables and improve model performance. Machine learning algorithms, including Linear Regression, Decision Trees, Random Forest, and Support Vector Machines, are trained and evaluated using cross-validation techniques. The performance of each model is assessed using metrics, such as Mean Magnitude of Relative Error (MMRE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Finally, the best-performing model is selected and validated using an independent testing dataset.

Linear Regression: Linear regression analysis predicts a variable's value based on other attributes' values. There are two types of variables:

1. *Dependent Variable*: The variable being predicted.
2. *Independent Variable*: The variable used to make the prediction.

How Linear Regression Works: Linear regression uses independent variables to create a linear mathematical equation that best predicts the dependent variable's value. The goal is to fit the output on a straight line, minimizing the difference between actual and predicted output.

- The Linear Regression Equation

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn + e$$

Where:

1. Y is the dependent variable
2. β1, β2, …, βn are coefficients
3. X1, X2, …, Xn are independent variables
4. β0 is the intercept (where the line meets the vertical axis)
5. e is the error term (random factors affecting Y)

Key Points

- Linear regression is a simple and comparative technique
- It's less complex than other prediction methods
- The technique estimates the value of the dependent variable (Y) based on the independent variable(s) (X)

Support Vector Regression: SVR is a regression analysis model introduced by Drucker et al. in 1997. It's an extension of the Support Vector Machine (SVM) algorithm.

How SVR Works: SVR uses the principle of structural risk minimization to find the best hyperplane that minimizes the error. It's particularly useful for small datasets and has excellent generalization ability.

- There are two main hyperparameters in SVR:

1. Kernel function: This determines the separation boundary.
2. Penalty parameter (C): This controls the error term.

Kernel Functions: Kernel functions project input data into high-dimensional feature spaces. Different kernel equations can be used to create various decision boundaries. The performance of SVR heavily depends on the choice of kernel function.

Objective Function: The objective function in SVR aims to minimize the size of the normal vector |w|. The constraints are:

$$xiw + b = 0 \text{ for the normal vector}$$
$$xiw + b \geq +1 \text{ for positive class}$$
$$xiw + b \leq -1 \text{ for negative class}$$
$$yi(xiw + b) - 1 \geq 0 \text{ for all data points}$$

- The goal is to minimize the error while maximizing the margin between classes.

$$\text{Minimize} \quad \frac{\|w\|^2}{2}$$

$$\text{Maximize} \quad W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \mathbf{x}_j)$$

Random Forest Regression: Random Forests is a machine learning algorithm introduced by Tin Kam Ho in 1995. It's used for classification and regression problems.

How Random Forests Work: Random Forests combines multiple decision trees to improve accuracy. Here's how:

1. Create multiple decision trees using random features.
2. Train each tree on a subset of the data.
3. Combine the predictions from each tree.

Advantages of Random Forests: Random Forests has several benefits.

1. More accurate error rate estimates: Compared to decision trees.
2. Improved prediction: Random Forests can handle nonlinearities in the data.
3. Robust against overfitting: Random Forests reduces the risk of overfitting.
4. Better results: Compared to other algorithms like Support Vector Machines and Neural Networks.

Random Forest Regression:

1. Running an unpruned regression on each decision tree.
2. Combining the predictions from each tree.

- This approach provides accurate and reliable results, making Random Forests a popular choice for many machine learning tasks.

Decision Tree Regression: Decision Tree Regression is a tree-based structure that predicts numerical results for the dependent variable. It's similar to Classification and Regression Trees (CART).

- **M5P Algorithm**: The M5P algorithm is an implementation of Quinlan's M5 algorithm. It generates tree-based models with multivariate linear models at the leaves, whereas regression trees have values at the leaves.
- **Tree Generation**: Trees are built using the usual decision-tree method, which employs splitting criteria to minimize intra-subset variation.

Machine Learning for Software Effort Estimation: Our study used the Desharnais dataset to compare various machine learning models for estimating software project effort. We found that:

- MLPNN model performed best: Achieving an R2 value of 0.79380, surpassing other models like LR, SVM, and KNN.
- Explained 79% of estimated variance: With only marginal differences in R2 values among models.
- *Analysing Correlated Elements*: We analysed the association between correlated elements and effort using seven machine learning methods. Performance was evaluated based on error values.
- *Innovative Effort Prediction Technique*: Our team introduced a technique combining Confidence Interval Analysis and Mean Absolute Error assessment. This approach:

  1. Demonstrated promise: Through trials involving over 700 software programs.

  2. Found applications: In diverse fields like pharmaceutical research, biochemistry, and computer vision.

Feature Selection and Classification*:* We used optimization techniques to select feature subsets and transferred them to classifiers (SVM, ANN, and Decision Tree) for classification and regression tasks. This process yielded excellent results across various datasets.

Comparison with COCOMO Model: Using the NASA benchmark data, we employed machine learning techniques like Naive Bayes, Logistic Regression, and Random Forests to make predictions. Each method outperformed the benchmark COCOMO model in production prediction.

Regression Techniques for Software Cost Estimation: Our investigation focused on regression techniques, notably the M5 algorithm and Linear Regression, for estimating software cost. The results indicated that:

1. M5 method exhibited minor errors: Compared to Linear Regression in prediction.

- **Performance Evaluation**: After applying machine learning algorithms, five statistical indicators are used to evaluate performance. These indicators calculate the error between predicted effort and actual effort in the dataset.

1. Mean Absolute Error (MAE)
2. Root Mean Squared Error (RMSE)
3. Relative Absolute Error (RAE)
4. Root Relative Squared Error (RRSE)
5. Correlation Coefficient R2

Error Measures: The error measures are calculated using the following formulae:

$$MAE = (1/n) * \Sigma|A\tilde{} - A|$$

$$RMSE = \sqrt{((1/n) * \Sigma(A\tilde{} - A)^2)}$$

$$RAE = (\Sigma|A\tilde{} - A|) / (\Sigma|A - \bar{A}|)$$

$$RRSE = (\sqrt{((1/n) * \Sigma(A\tilde{} - A)^2)}) / (\sqrt{((1/n) * \Sigma(A - \bar{A})^2)})$$

$$R2 = 1 - (\Sigma(A\tilde{} - A)^2) / (\Sigma(A - \bar{A})^2)$$

Where:

$A\tilde{}$ is the predicted effort

A is the actual effort

$\bar{A}$ is the mean of A

n is the number of data points

Datasets:

NASA93: The NASA 93 dataset is a publicly available dataset containing information on 93 software projects developed by NASA between 1975 and 1988. The dataset includes 17 attributes, such as project size, complexity, team experience, and development environment, which were collected to support the development of software cost estimation models. The dataset is widely used in software engineering research, particularly in the areas of software cost estimation, effort prediction, and project management. The NASA 93 dataset provides a valuable resource for researchers and practitioners seeking to develop and evaluate software cost estimation models, and its use has contributed significantly to advances in software engineering research and practice. The dataset has following objects.

- 93 rows (projects)
- 26 columns (features)
- ARFF (Attribute Relation File Format)

Description of the data: A detailed description of the dataset is provided in the below Table, which outlines the features and characteristics of the data used to train the models.

Attribute	Symbol	Description	Datatype
Project	project name	Name	String
Category of application	cat2	Which field is this project related to	String
System	forg	Flight system or Ground system	character [f, g]
NASA center	center	Which NASA center had worked on the project	Number between 1 to 6
Capability of analyst	acap	Increase these to decrease effort	Positive integer
Capability of programmers	pcap		
Domain experience	aexp		
Current programming tech-niques	modp		
Software tool usage	tool		
Experience with programming languages	lexp		
Experience with VM	vexp		
Time restriction	sced		
The primary memory restriction	stor	Increase these to decrease effort	Positive integer
Database size	data		

Runtime restriction on CPU	time		
Turnaround time	turn		
Machine volatility	virt		
The difficulty of the process	cplx		
Required Software reliability	rely		
Equivalent physical line of code	equivphyskloc	Kilo lines of code	Positive integer
Development effort	act_effort	The effort in terms of persons month	Positive integer

"Development effort" is the primary dependent variable that is under study. The dataset's actual value will be compared to the expected value as part of the prediction process carried out by the Machine Learning algorithms. The main goal is to predict "Development Effort" using Machine Learning algorithms.

- The dataset is split into two parts:
- Training set (70% of the data)
- Testing set (30% of the data)
- The algorithms will compare the predicted values with the actual values in the dataset.

Evaluation, Results, and Discussion: This section presents the results of the experiment.

The following Machine Learning models were used:

Method Name	MAE	RMSE	RAE	RRSE	R_2
Linear Regression	760.11	1365.5	1.45	1.24	-0.41
Support Vector Regression	178.7	323.61	1.05	1.042	-0.067
Random Forest Regression	642.5	1481.7	0.85	1.070	0.2781
Decision Tree Regression	760.0	1666.1	0.82	1.098	0.0875

The comparison of machine learning algorithms for software effort estimation reveals varying levels of performance. Support Vector Regression (SVR) emerges as the top performer, with the lowest Mean Absolute Error (MAE) of 178.7 and Root Mean Squared Error (RMSE) of 323.61. SVR also exhibits a Relative Absolute Error (RAE) of 1.05 and a Relative Root Squared Error (RRSE) of 1.042, indicating its superior predictive capabilities. Random Forest Regression ranks second, with an MAE of 642.5 and RMSE of 1481.7, while Decision Tree Regression and Linear Regression trail behind, with higher error values and lower R-squared values, indicating poorer fit and predictive performance. The evaluation of various models using error-measuring methods, such as R-squared (R2), Mean Absolute Error (MAE), Relative Absolute Error (RAE), Root Mean Squared Error (RMSE), and Root Relative Squared Error (RRSE), shows that the Random Forest method exhibits the highest correlation coefficient, indicating a strong connection between the predicted and actual values. While Support Vector Regression (SVR) demonstrates a smaller RMSE value, suggesting more accurate predictions on the test data, Random Forests tend to predict better than linear regression due to their ability to adapt to nonlinearities in the data. Overall, the study underscores the importance of selecting the appropriate model for predicting development effort, considering factors such as dataset characteristics and the complexity of relationships between variables.

ISBSG: The International Software Benchmarking Standards Group (ISBSG) dataset is a comprehensive repository of software project data, containing information on over 7,000 projects from various industries and countries. The dataset includes detailed information on project characteristics, such as project size, complexity, development methodology, team experience, and effort hours. The ISBSG dataset is widely regarded as one of the largest and most reliable sources of software project data, providing a valuable resource for software engineering research, benchmarking, and process improvement. The dataset is continuously updated and expanded, ensuring that it remains a relevant and authoritative source of information for the software industry.

Dataset Characteristics:

- Number of projects: Over 7,000
- Data collection period: 1990s to present
- Industries represented: Various, including finance, government, healthcare, and IT
- Countries represented: Over 20 countries
- Project sizes: Small to large (up to 100,000 function points)

Data Attributes:

1. Project Identification:
   - Project ID
   - Organization ID
   - Project name
2. Project Characteristics:
   - Project type (e.g., new development, enhancement, maintenance)
   - Development methodology (e.g., waterfall, agile, hybrid)

- Project size (function points or lines of code)
- Complexity (low, medium, high)

3. Effort and Schedule:
   - Total effort hours
   - Development effort hours
   - Testing effort hours
   - Project duration (months)
4. Team and Resources:
   - Team size
   - Average team experience (years)
   - Number of developers
   - Number of testers

5. Quality and Reliability:
   - Number of defects
   - Defect density (defects per function point)
   - Reliability (mean time between failures)

The analysis reveals a significant correlation between CPU runtime restrictions and database size, as well as memory requirements. As the database size and memory requirements increase, the CPU experiences increased runtime restrictions, indicating a dependency between these variables. Furthermore, the study highlights a positive relationship between analyst competency and domain knowledge acquisition, demonstrating that as an analyst's capability improves, their domain familiarity also enhances.

ISBSG Dataset	**RMSE with Correlation**	**RMSE without Correlation**
Linear Regression	5975.48	4543.20
Random Forest	7660.77	7659.96
Decision Tree	9059.69	14485.16

ISBSG Dataset	**R2 with Correlation**	**R2 without Correlation**
Linear Regression	0.77	0.87
Random Forest	0.63	0.63
Decision Tree	0.48	-0.32

- ❖ Linear Regression performs the best overall, with the lowest RMSE (4543.20) and highest R2 (0.87) without correlation.
- ❖ Random Forest shows consistent performance with and without correlation, but its RMSE and R2 values are not as good as Linear Regression.
- ❖ Decision Tree performs poorly, especially without correlation, with a high RMSE (14485.16) and negative R2 (-0.32).

The performance comparison of three machine learning algorithms - Linear Regression, Random Forest, and Decision Tree - on the ISBSG dataset reveals that Linear Regression outperforms the other two algorithms, yielding the lowest Root Mean Squared Error (RMSE) of 4543.20 and the highest Coefficient of Determination (R2) of 0.87 when correlation is not considered. This indicates that Linear Regression provides the most accurate predictions with the lowest error. In contrast, Random Forest shows consistent performance with and without correlation, but its RMSE and R2 values are not as impressive as those of Linear Regression. Decision Tree, on the other hand, performs poorly, especially without correlation, with a remarkably high RMSE and a negative R2, suggesting that it is not a suitable algorithm for this dataset. Overall, Linear Regression emerges as the best algorithm for this dataset, making it a reliable choice for software effort estimation tasks.

Desharnais: The Desharnais dataset is a publicly available dataset containing information on 81 software projects developed by a Canadian software company between 1986 and 1990. The dataset includes 10 attributes, such as project size, complexity, team experience, and development environment, which were collected to support the development of software cost estimation models. The dataset is notable for its high-quality data, which was collected prospectively, meaning that the data was collected as the projects were being developed, rather than retrospectively. The Desharnais dataset has been widely used in software engineering research, particularly in the areas of software cost estimation, effort prediction, and project management.

Dataset Characteristics:

- Number of projects: 81
- Data collection period: 1986-1990
- Industry: Software development
- Company: Canadian software company

Data Attributes:

1. Project Identification:
   - Project ID
   - Project name
2. Project Characteristics:
   - Project size (lines of code)
   - Project complexity (low, medium, high)
   - Project type (new development, enhancement, maintenance)
3. Team and Resources:
   - Team size
   - Average team experience (years)
   - Number of developers
   - Number of testers
4. Effort and Schedule:
   - Total effort hours
   - Development effort hours
   - Testing effort hours
   - Project duration (months)
5. Cost and Productivity:
   - Total cost
   - Development cost
   - Testing cost
   - Productivity (lines of code per hour)

Data Quality:

- Data collected prospectively (during project development)
- Data validated and cleaned by the original researchers
- Data considered to be of high quality and reliability

Desharnais Dataset	**RMSE with Correlation**	**RMSE without Correlation**
Linear Regression	1.51	1943.91
Random Forest	229.28	9166.13
Decision Tree	228.24	3744.23

Desharnais Dataset	**R2 with Correlation**	**R2 without Correlation**
Linear Regression	1	0.70
Random Forest	0.99	0.59
Decision Tree	0.99	-0.09

- ❖ Linear Regression excels with correlation: Achieves perfect R-squared value (1) and low RMSE (1.51) when correlation is considered.
- ❖ Linear Regression struggles without correlation: RMSE increases significantly (1943.91) and R-squared value drops (0.70) without correlation.
- ❖ Random Forest and Decision Tree robustness: Perform relatively better without correlation, with lower RMSE values compared to Linear Regression.
- ❖ Importance of correlation: Correlation significantly impacts the performance of Linear Regression, while Random Forest and Decision Tree are more robust but may still benefit from correlation.

The analysis of the Desharnais dataset reveals that Linear Regression performs exceptionally well when correlation is considered, achieving a Root Mean Squared Error (RMSE) of 1.51 and a perfect R-squared value of 1. However, its performance deteriorates significantly without correlation, resulting in an RMSE of 1943.91 and an R-squared value of 0.70. Random Forest and Decision Tree algorithms exhibit relatively better performance without correlation, with RMSE values of 9166.13 and 3744.23, respectively. Notably, both Random Forest and Decision Tree achieve high R-squared values when correlation is considered, indicating excellent fit. Overall, the results suggest that Linear Regression is highly sensitive to correlation, while Random Forest and Decision Tree are more robust but may require correlation to achieve optimal performance.

Conclusion: This study aimed to develop and evaluate machine learning models for software effort estimation using three publicly available datasets (ISBSG, Desharnais, and NASA 93). The results show that different algorithms exhibit varying levels of performance across different datasets and correlation scenarios. On the ISBSG dataset, Linear Regression achieved the best performance without correlation, with an RMSE of 4543.20 and an R-squared value of 0.87. Linear Regression also performed exceptionally well on the Desharnais dataset when correlation was considered. Random Forest and Decision Tree algorithms demonstrated robustness and performed relatively well without correlation. The NASA 93 dataset results highlighted the importance of algorithm selection and hyperparameter tuning. Overall, this project highlights the importance of carefully selecting and preprocessing data, considering correlation, and evaluating multiple algorithms to identify the best approach for a given dataset and problem scenario. The results and insights gained from this project can inform the development of more accurate and reliable software effort estimation models, ultimately supporting better decision-making and project management in software development.

1. **Estimating Software Development Effort:** Estimating the effort required for software development is a critical task for project managers. Accurate estimates help establish the cost, required staff, and schedule for a project.
2. **Challenges in Estimation:** Studies have shown that errors in early-stage project estimation are a primary cause of software project failures. The performance of estimation approaches depends on various factors, including project complexity, duration, personnel skill, and development process.
3. **Review of Cost-Estimating Methodologies:** This paper reviews several cost-estimating methodologies, aiming to improve our understanding of the subject.
4. **Key Findings:** No single approach estimates software development effort exceptionally well. However, Random Forest Regression yields the best results, with the lowest R2 and MAE values.

Practical Applications: Machine learning (ML) based approaches can be applied in practice by:

- Collaborating between project managers and data scientists
- Gathering relevant data (historical project data, developer expertise, project complexity, duration)
- Employing ML algorithms to enhance early-stage project estimates

Future Research Directions: The future scope of this project encompasses several exciting avenues, including the integration of agile methodologies, incorporation of advanced machine learning techniques, and development of multi-objective optimization models. Additionally, exploring real-time estimation capabilities, domain-specific estimation models, and hybrid approaches that combine traditional estimation techniques with machine learning will further enhance the framework's accuracy and applicability. Furthermore, deploying the estimation framework on cloud platforms, incorporating explainable AI techniques, and enabling continuous learning will ensure the framework remains adaptable, transparent, and effective in supporting informed decision-making in software project management.

REFERENCES:

- Albrecht, A. J. (1979). Measuring application development productivity. Proceedings of the 1979 IBM Application Development Symposium, 83-92.
- Boehm, B. W. (1981). Software engineering economics. Prentice Hall.
- Elish, M. O., Jiang, Z., & Hassan, M. (2018). Machine learning in software development: A systematic mapping study. IEEE Transactions on Software Engineering, 44(10), 961-984.
- Huang, X., Xia, X., & Lo, D. (2018). Neural networks for software effort estimation: A systematic review. IEEE Transactions on Software Engineering, 44(10), 985-1004.
- Kaur, A., Singh, H., & Kumar, R. (2018). Decision tree-based software effort estimation. Journal of Software: Evolution and Process, 30(6), e2044.
- Singh, H., Kaur, A., & Kumar, R. (2018). Feature selection for software effort estimation using mutual information. Journal of Systems and Software, 136, 137-146.
- Xia, X., Lo, D., & Huang, X. (2019). Ensemble methods for software effort estimation: A systematic review.
- IEEE Transactions on Software Engineering, 45(10), 1055-1072.
- “A Systematic Review of Software Effort Estimation" by Barbara Kitchenham, et al. (2007) Published in: Information and Software Technology
- "Estimating Software Project Effort Using Analogies" by Martin Shepperd and Chris Schofield (1997) Published in: IEEE Transactions on Software Engineering
- "A Comparison of Software Effort Estimation Techniques" by Jiyun Park and Scott Mac Donell (2012) Published in: Journal of Systems and Software
- "Using Machine Learning to Improve Software Effort Estimation" by Burak Turhan and Ayse Bener (2009) Published in: Journal of Software Maintenance and Evolution
- "Software Effort Estimation Using a Hybrid Approach" by P. Radhika and K. Srinivas (2015) Published in: Journal of Intelligent Information Systems