International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# Ensemble Neuro-Computing for Early Software Cost Estimation: Predictive Accuracy, Risk Reduction, and Economic Decision Impact

**Debanjan Ghosh[1], Dr. Arvind Kumar Pandey[2]**

[1]Dept. of Computer Science & IT, ARKA JAIN University, Jharkhand, India.
[2]School of engineering & IT, ARKA JAIN University, Jharkhand, India.

**Abstract**
Effective planning, pricing, and general project success depend on accurate cost estimating in the early stages of software development. Traditional estimation techniques, nevertheless, have difficulty properly representing nonlinear correlations and managing uncertainty owing to incomplete beginning criteria. Early-stage software cost estimating might be improved using an ensemble-based neural framework based on multilayer perceptron networks suggested in this study. The method enhances prediction accuracy, stability, and generalising by means of several separately trained models. Experimental data on reference datasets indicate that the suggested model beats regression methods, single neural networks, and standard algorithmic approaches. Important changes are seen in major assessment criteria, including decreased MMRE (below 0.30), smaller RMSE (76.1), and higher PRED (25), therefore showing more dependable forecasts. Along with technical performance, the study presents an Economic Risk Reduction Index (ERRI) meant to evaluate the real-world effects of enhanced estimating. The results indicate that improved accuracy lowers estimation risk to about 4.1%, therefore promoting better-informed pricing, effective resource distribution, and less financial uncertainty. Overall, the suggested framework not only raises the accuracy of estimations but also helps software projects to make better decisions and control costs.

**Keywords:** Software Cost Estimation, Ensemble Learning, Multilayer Perceptron (MLP), Prediction Accuracy, Economic Risk Reduction (ERRI), Early-Stage Software Development

## 1. Introduction

Estimating development expense is a crucial responsibility in modern software-driven companies that directly impacts planning, budgeting, and general project management. Good predictions assist interested parties in deciding whether a project warrants funding before investing substantial assets. By contrast, false predictions might cause bad choices and unreasonable hopes. Particularly in competitive environments, software projects usually evolve under tight cost and schedule restrictions. Early cost estimates are crucial for creating contractual agreements, making proposals, and determining pricing policies. Should these first projections be off, the repercussions can follow throughout the whole project development cycle, causing budget overruns, delays, and scope of work compromises. From a managerial point of view, frequent calculation mistakes might lower faith in planning methods. Over time, this could undermine stakeholder trust and impact an organisation's credibility, therefore influencing future financing and project opportunities. The actual character of software development makes estimating difficult. Often

changing, criteria cause ambiguity in effort estimate. Furthermore, complicating exact estimating is the growing system complexity and nonlinear connections between cost components. Human elements like team experience and productivity also contribute to this variance. These problems get even more important as software systems become more sophisticated and complex. Software is regarded as a major business resource nowadays; therefore, more attention is paid to enhancing estimating methods. Correct cost estimation improves operational efficiency, promotes better resource use, and helps long-term sustainability. Furthermore, it improves governance and responsibility inside businesses.

Economically speaking, the consequences of bad estimation go beyond simple project failures. Early-stage estimate errors can affect larger decisions about capital allocation and investment planning; therefore, the need for more dependable estimating methods becomes more urgent. Organizations usually invest money according on expected development expenses. Capital might be wrongly allocated to low-return projects if these predictions are off. These misallocations raise total financial exposure. Estimation errors also twist return-on-investment computations utilized by decision-makers. Excessive return on investment (ROI) expectations might result in unauthorized project approvals. On the other hand, underestimated advantages may cause otherwise valuable projects to be turned down. In contract-based settings, incorrect estimates could cause disagreements between customers and suppliers. Estimating uncertainty is especially sensitive for fixed-price contracts. Profit erosion can happen to vendors when actual expenses are more than projections. Delayed shipments or diminished system quality can affect clients. Pricing models in competitive bidding situations are also affected by estimation errors. Winning contracts is possible via aggressive underestimation, but long-term profit is threatened. On the other hand, overestimation could lower market competitiveness. Repetitive estimation errors have a considerable overall economic effect. Organizations may see decreasing margins and higher project failure rates. These financial shortcomings stifle corporate scalability. For operational as well as financial sustainability, then, better estimation reliability is crucial.

Both academia and industry have come to accept conventional software cost estimating techniques. Structured estimating methods are provided by models like Constructive Cost Model (COCOMO) and Function Point Analysis. These models depend on predetermined cost factors and observed coefficients. Parameter calibration and selection depend mostly on expert opinion. Although successful in stable surroundings, these theories have difficulty in uncertain ones. Early development phases sometimes lack the data required to correctly identify cost drivers. Volatility in requirements lowers static estimating assumptions' dependability. Moreover, conventional models suppose near-linear or linear connections among variables. Modern software projects show sophisticated nonlinear interactions among technical and human elements. The stiffness of conventional models restricts their capacity for flexibility. Estimation depending on experts injects bias and subjectivity. Inconsistent estimates arise from variations in professional experience. Moreover, early project phases frequently restrict expert access. Manual calibration methods are prone to mistakes and take a while. These constraints limit the actual usefulness of conventional methods. The estimating accuracy reduces more as projects become more complicated. One major disadvantage is the inability to dynamically learn from historical information. Hence, businesses look for more data-driven and flexible choices. Solutions with potential are offered by developments in computational intelligence. Machine learning methods provide the capacity to automatically simulate complicated patterns.

Recent advancements in machine learning have changed the direction of study on estimating software cost. Data-driven methods improve prediction accuracy by utilising historical project data. Machine

learning models are capable of capturing nonlinear interactions between cost drivers. Artificial Neural Networks are the most popular of these techniques. Multilayer perceptron neural networks (MLPNNs) are particularly helpful due to their universal approximating capability. MLPNNs are capable of learning difficult mappings between input characteristics and development effort.

Empirical studies show their better forecasting performance than that of conventional approaches. MLPNNs do have certain drawbacks, though. Single neural network models are sensitive to early weight selection. Random initialisation can produce subpar local minima. Model stability is additionally impacted by data variation and noise. Overfitting is still a frequent problem in little or skewed datasets. These elements limit generalisability and robustness. Unreliable projections constrain their business use. For management decision-making, model interpretability is also a concern. Often stakeholders want dependable and consistent estimations. Variability in neural network results erodes confidence. Therefore, for actual usage, strengthening of robustness is vital. One bright path is offered by ensemble learning. Merging several models lowers variability and increases reliability.

This study suggests an ensemble-based MLPNN architecture to overcome the constraints of single MLPNN models. The ensemble method compiles forecasts from several networks trained independently. This grouping reduces sensitivity to starting weight. It also lowers the variance and effect of data noise. For a range of datasets, ensemble learning enhances generalisation ability. Simultaneous improvement in estimation stability and accuracy is the intended outcome of the suggested system. Better technical accuracy yields financial rewards straight away. More accurate projections help to lower financial risk and overrun expenses. Organisations can distribute funds more effectively. Increased estimation accuracy helps ROI projections to be better. Improved predictions aid in wise investment decisions. Furthermore bolstering pricing and bidding tactics is this structure. Lower estimation uncertainty raises contractual stability. From a managerial standpoint, consistent estimates raise stakeholder confidence. The suggested model links technical performance with economic results. It links expected accuracy with organisational profit objectives. The research stresses measurable commercial value over purely algorithmic performance. Experimental assessment shows better accuracy than baseline models. Analysis of economic effects emphasises measurable cost reductions. Therefore, the suggested ensemble MLPNN architecture has technical as well as financial benefits.

## 2. Relevant Work

Parametric and expert-based methods like COCOMO, SLIM, and Function Point Analysis dominated early software cost estimate research. These models fall short in adapting to current agile settings and nonlinear cost dynamics even if they offer interpretability.

Recent developments in software cost estimation have moved sharply toward ensemble neuro-computing to tackle the non-linear complexity found in contemporary development initiatives. Seeking better predictive accuracy, Wang et al. (2026) and Varshini and Kumari (2024) have shown how hybrid architectures—which combine metaheuristic optimisers with deep learning layers—substantially outperform conventional regression approaches. Iordan (2024) and Rijwani and Jain (2024), who used optimised LSTM networks to efficiently capture sequential dependencies in project data typically missed by static models, further support this technical development. Using stacking ensemble methods, Kassaymeh et al. (2024) and Sakhrawi et al. (2024) successfully boosted generalisation and decreased bias, hence demonstrating that combining several estimators produces more reliable results than individual models. These computer advancements directly help with better risk management; for example,

Bisikirskienė et al. (2025) and Ranković et al. (2024) emphasised how machine learning classifiers may now automate risk identification from unstructured data. Furthermore, predicting project deviations is done using predictive analytics; Nishat (2025) and Almahameed and Bisharah (2024) demonstrate that artificial intelligence technologies accurately forecast variation orders and bottlenecks. Regarding technical stability at last, Gao et al. (2025) and Ma et al. (2024) proved that ensemble methods are outstanding in early defect detection and security vulnerability analysis, therefore reducing expensive refurbishment.

Integration of highly accurate estimation models has major consequences for strategic corporate decision-making and economic performance. Lenhart et al. (2024) and Dzhusupova et al. (2024) showed the real worth of these systems in manufacturing and EPC projects where AI-driven estimation guarantees accurate cost baselines. This skill helps quick economic feasibility studies; automated cost modelling lets businesses, according to Mandolini et al. (2024), make go/no-go decisions during the conceptual stage with great certainty. On a larger level, Cerutti et al. (2025), Acemoglu (2024), and Brynjolfsson et al. (2025) presented empirical data demonstrating that the implementation of these AI tools results in measurable productivity increases and major labour market changes. These macroeconomic trends are mirrored at the enterprise level, where Bick et al. (2025) indicated that the rapid uptake of generative models is fuelled by tangible reductions in operational costs. Moreover, Alzarooni (2025) and Mouri Zadeh Khaki et al. (2025) stressed that by simulating human-like decision-making processes, artificial intelligence advances project management dynamics and negotiation.

Bick et al. (2025) claim that these macroeconomic trends are mirrored at the corporate level, where noticeable drops in operating costs are what propel the rapid adoption of generative models. Additionally, Alzarooni (2025) and Mouri Zadeh Khaki et al. (2025) emphasised that AI enhances negotiation and project management dynamics by imitating human-like decision-making processes. Ultimately, Kapulica and Jurina (2024) and Feng et al. (2024) came to the conclusion that the convergence of neuro-computing accuracy with strategic economic analysis is a critical foundation for long-term project management and competitive advantage.

## 3. Methodology

### 3.1 Multilayer Perceptron Neural Network (MLPNN)

As its basic predictive unit, the suggested estimation model uses a Multilayer Perceptron Neural Network. Supervised, feed-forward neural architecture able to approach complex nonlinear functions is MLPNN. It has an output layer, one or more hidden layers, and an input layer. The input layer early-stage software project characteristics. These qualities include team capability characteristics, growth environment indicators, functional complexity, and size measures. Represent the input feature vector as $X = [x_1, x_2, \dots, x_n]$.Every input neuron has a normalized cost driver.Nonlinear transformation of the input characteristics is done by the hidden layer.The j-th hidden neuron is calculated as a weighted sum of inputs.Model flexibility is enhanced by the addition of a bias term.The net input goes via a sigmoid activation function.The sigmoid function adds nonlinearity into the network.

It is defined as $f(z) = \frac{1}{1+e^{-z}}$. (1)

This activation assists the network in picking up intricate effort patterns. The hidden neuron output is designated as $h_j$. Hidden neuron activations are accumulated on the output layer.At the output layer is a linear activation function.For continuous-valued effort estimation, this option is appropriate.The estimated

effort output is shown as ŷ.Training helps to fine-tune synaptic weights.Backpropagation is used in the training process.Mean Squared Error serves as the loss function.The network regularly changes weights to reduce expected prediction error.Training the MLPNN consists in minimizing the divergence between expected and actual project effort values; this learning process goes on until convergence criteria are met.

**3.2 Training and Error Minimization Strategy**

The goal function chosen is the mean squared error. Mathematically stated, it is the average squared variance between predicted and actual outputs. MSE gives a smooth and differentiable loss surface. This quality allows for effective gradient-based optimization. N past software projects make up the training dataset. Every project has a feature vector and related effort value. Forward propagation maps input characteristics to output predictions. At the output layer is calculated the error signal. This error spreads backwards across the network. Calculated are loss function gradients with regard to every weight. Gradient descent is used for weight changes. Learning rate regulates the extent of weight corrections. Appropriate learning rate tuning helps to stabilize things. Early stop systems help to manage overfitting. Network training goes on until the error in validation settles. At the start of training, weights are randomly initialized. This randomness causes network performance fluctuation. Various initializations can generate several local minima. Therefore, single MLPNN projections might be unstable. Such instability shakes management decision-making faith. Consistency throughout runs is necessary for effective estimation. Ensemble learning is driven by the sensitivity of MLPNN. Consolidating several students lessens reliance on start-up. Statistical enhancement of generalization comes about from ensemble methods. This drives the architecture presented in the following part.

**3.3 Ensemble Learning Architecture**

To reduce the variance present in single MLPNN models, an ensemble learning approach is used. Using a technique founded on bagging, the group builds itself. Multiple models are taught on bootstrapped samples through bagging. Every bootstrap sample is drawn from the original dataset with replacement. Given K bootstrap samples, K independent MLPNNs are trained. Every MLPNN acquires somewhat different data distributions. This variety makes groups more efficient. Every base learner has the same network architecture. Their training data and weight starting, though, vary. Every trained network generates its own effort estimate. By averaging individual outputs, one finds the ensemble prediction. Mathematically, the mean of K predictions is the ensemble output. These gathering lowers variation without raising bias. Extreme prediction mistakes are smoothed out by ensemble averaging. It improves robustness throughout unexpected project cases. This method lowers outliers' and noise's susceptibility. Ensemble learning also enhances stability over several runs. Statistically speaking, lowering variance improves expected accuracy. The ensemble does not need more domain expertise. It makes better use of available data. Parallel training helps to keep computational burden under control. For big datasets, the ensemble framework is scalable. If necessary, it can hold more

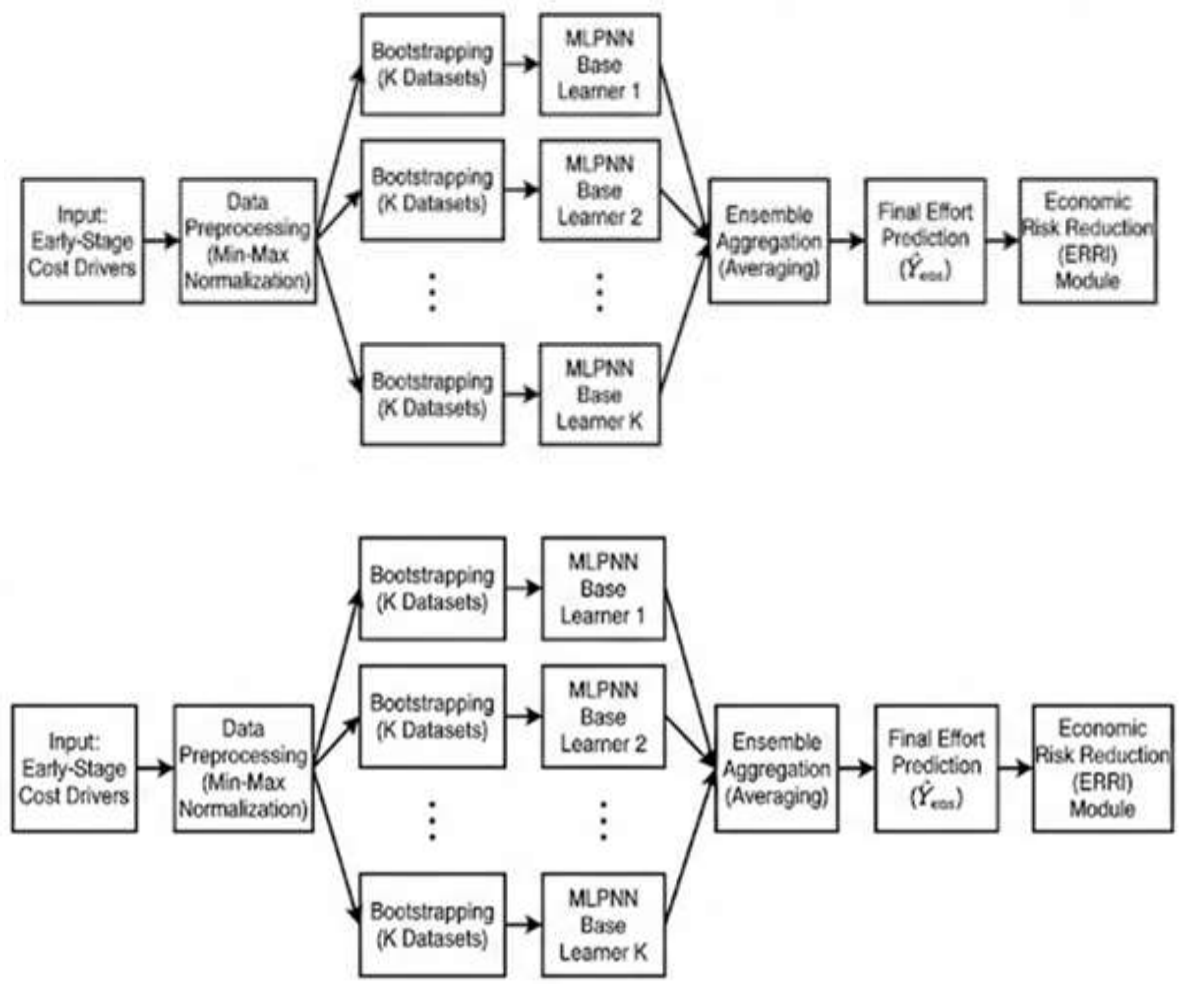


Figure 1: Overall Architecture of the Proposed Ensemble MLPNN Framework simple learners. The building is suitable for industrial installation. Increased toughness boosts management self-assurance. The ensemble is the predictive core of the proposed framework.

### 3.4 Novel Economic Risk Reduction Model (ERRI)

This work presents an explicit economic assessment model in addition to technical correctness. Financial risk exposure depends directly on the accuracy of software cost estimate. Conventional research frequently misses this economic side.

To address this gap, an Economic Risk Reduction Index is proposed. The model quantifies financial benefits of improved estimation. Let actual project cost be denoted as $C_a$. Let estimated project cost be represented as $C_e$. Relative estimation error is defined as the absolute deviation ratio. This error is normalized by actual project cost. The relative error captures proportional deviation. It is independent of absolute project size. Expected financial risk exposure is computed as the product of error and actual cost. This formulation reflects potential financial loss. Higher estimation error implies greater financial exposure. The model compares baseline and proposed estimation methods. Let $E_B$denote baseline risk exposure. Let $E_E$denote ensemble-based risk exposure. The Economic Risk Reduction Index is defined as the normalized difference. ERRI measures proportional reduction in financial risk. Values close to zero indicate negligible improvement. Values approaching one indicate substantial risk mitigation. The index provides a clear economic interpretation. It bridges technical metrics and business outcomes. Decision-makers can directly interpret financial impact. The model supports ROI-driven evaluation of estimation

techniques. It strengthens the practical relevance of the research. This economic contribution is a key novelty of the study.

**3.5 Dataset, Preprocessing, and Performance Metrics**

Using well-known benchmark datasets COCOMO81 and NASA93, which are frequently used in software cost estimation studies, the suggested approach is assessed. Regarding size, complexity, development environment, and team ability, these datasets represent varied software project features. Using several datasets improves the external validity and generalizability of the suggested model. Early-stage cost drivers are shown as feature vectors and related actual effort or expense data in each dataset. Designate the whole dataset as

$$\mathcal{D} = \{(X_p, y_p)\}_{p=1}^{N}, \quad (2)$$

where $X_p \in \mathbb{R}^n$represents the n-dimensional input feature vector of the p-th project and $y_p \in \mathbb{R}^+$denotes the actual development effort.

Efficient neural network training and convergence depends on data preprocessing, a key step.

Because cost drivers frequently have varying scales and units, min-max normalization is done to every input feature. Mathematically defined as

$$x'_{i,p} = \frac{x_{i,p} - \min(x_i)}{\max(x_i) - \min(x_i)}, \quad (3)$$

where $x_{i,p}$is the original value of the i-th feature for project p, and $x'_{i,p}$is the normalized value. This conversion avoids dominance of high-magnitude qualities by mapping all characteristics into the range [0,1]. Furthermore, enhancing numerical stability and accelerating gradient-based optimization is normalization. Then the normalized data set serves as input to the MLPNN ensemble.

Following preprocessing, the data is split into training and testing subsets. Cross-validation methods are used to guarantee the strength and justice of assessment.

This technique improves the reliability of results and lowers bias resulting from arbitrary data partitioning.

Standard and widely used accuracy measures help to assess model performance.

Average proportional estimating error is evaluated using the mean magnitude of relative error (MMRE), which is defined

$$\text{MMRE} = \frac{1}{N}\sum_{p=1}^{N} \frac{|y_p - \hat{y}_p|}{y_p}, \quad (4)$$

where $\hat{y}_p$is the predicted effort for the p-th project. MMRE enables direct comparison with prior cost estimation studies. The Prediction at level 25 percent (PRED (25)) metric measures the proportion of projects whose relative error does not exceed 25 percent. It is mathematically expressed as

$$\text{PRED}(25) = \frac{1}{N}\sum_{p=1}^{N} I\left(\frac{|y_p - \hat{y}_p|}{y_p} \le 0.25\right), \quad (5)$$

where $I(\cdot)$is an indicator function that returns 1 if the condition is satisfied and 0 otherwise. PRED (25) offers strong managerial interpretability by quantifying acceptable estimation accuracy.

In addition to relative error metrics, Root Mean Squared Error (RMSE) is employed to capture absolute deviation magnitude and penalize large estimation errors. RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}_p)^2}. \quad (6)$$

From an economic viewpoint, RMSE is especially crucial since big absolute deviations typically reflect substantial financial losses. MMRE, PRED (25), and RMSE enable balanced assessment across

proportional accuracy, tolerance-based performance, and absolute risk sensitivity. Multiple complementary metrics help to avoid prejudiced assessment of the suggested framework. Both technological correctness and managerial importance are clearly taken into account. Statistical study of performance results guarantees dependability and consistency. Dataset-independent and totally repeatable is the evaluation approach. This all-encompassing approach strongly confirms the suggested ensemble-based costing model.

**3.6 Mathematical Model of Ensemble-Based Economic Cost Estimation**

Mathematically framed as a supervised learning optimization issue with economic risk reduction is the suggested software cost estimation approach. Present the software project data set as follows:

$$\mathcal{D} = \{(X_p, y_p)\}_{p=1}^{N}, \quad (7)$$

where $X_p \in \mathbb{R}^n$denotes the normalized feature vector of the p-th project and $y_p \in \mathbb{R}^+$represents the actual development effort or cost. An ensemble of $K$heterogeneous Multilayer Perceptron Neural Networks (MLPNNs) is employed.

The output of the $k$-th learner is:

$$\hat{y}_{p,k} = f_k(x_p; \theta_k) \quad (8)$$

where $\theta_k$denotes the parameters of the $k$-th MLPNN.Learner diversity is ensured through architectural variation, random initialization, and data resampling. Traditional ensemble averaging assumes uniform learner reliability, which is unsuitable for heterogeneous software projects. To address this limitation, an attention-based gating network dynamically assigns weights to each base learner.

The ensemble prediction is defined as:

$$\hat{Y}_p = \sum_{k=1}^{K} \alpha_k (x_p)\, \hat{y}_{p,k} \quad (9)$$

where $\alpha_k(x_p)$is the attention weight generated by a gating network $G(x_p; \phi)$.

The weights are normalized using a SoftMax function:

$$\alpha_k(x_p) = \frac{exp\,(g_k(x_p))}{\sum_{j=1}^{K} exp\,(g_j(x_p))} \text{ s.t. } \sum_{k=1}^{K} \alpha_k = 1 \quad (10)$$

This formulation guarantees convexity, differentiability, and interpretability of learner contributions. In real-world project management, underestimation poses significantly higher financial risk than overestimation.

To model this asymmetry, the Asymmetric Economic Loss Function (AELF) is defined as:

$$\mathcal{L}_{\text{Econ}} = \frac{1}{N} \sum_{p=1}^{N} \left[\gamma\, max\,(0, y_p - \hat{Y}_p)^2 + (1-\gamma)\, max\,(0, \hat{Y}_p - y_p)^2\right] \quad (11)$$

where $\gamma \in [0,1]$is the risk asymmetry parameter. $\gamma > 0.5$: risk-averse strategy & $\gamma = 0.5$: symmetric loss (equivalent to MSE).

The ensemble is trained end-to-end by jointly optimizing base learners and the gating network:

$$\mathcal{J}(\Theta, \Phi) = \mathcal{L}_{\text{Econ}}(Y, \hat{Y}) + \lambda \sum_{k=1}^{K} \| \theta_k \|_2^2 \quad (12)$$

where $\lambda$controls regularization. Gradient-based optimization ensures that prediction accuracy and economic safety are optimized simultaneously.

$$\text{Relative Risk Exposure: } \text{RRE}_p = \left| \frac{\hat{Y}_p - y_p}{y_p} \right| \cdot exp\left(\xi\, \mathbb{I}(y_p > \hat{Y}_p)\right) \quad (13)$$

where $\xi > 0$amplifies underestimation penalties.

Aggregate Economic Risk, $AER = \sum_{p=1}^{N} RRE_p \cdot Budget_p$ (14) Economic Risk

Reduction Index, $ERRI = \left(1 - \frac{AER_{Ensemble}}{AER_{Baseline}}\right)$ (15)

ERRI is bounded in $[0, 1]$. Higher values indicate superior economic performance.

This index turns risk exposure into a measurable result, connects predictive accuracy with financial impact, and facilitates direct cross-model comparison. The novel formulation incorporates economic thinking right into machine learning assessment. A dual-objective problem comprising accuracy and financial risk reduction is software cost estimating.

The combined objective can be expressed as:

$$\min_{\theta} \left[\mathbb{E}[(y - \hat{y})^2], \ \mathbb{E}[R \times y]\right] \quad (16)$$

where $R$denotes relative economic risk.

Though inherently multi-objective, the suggested ensemble does not call for overt dual optimization. Better predictive accuracy lowers economic exposure and statistical error at once, therefore letting the economic aim come about naturally from knowledge.

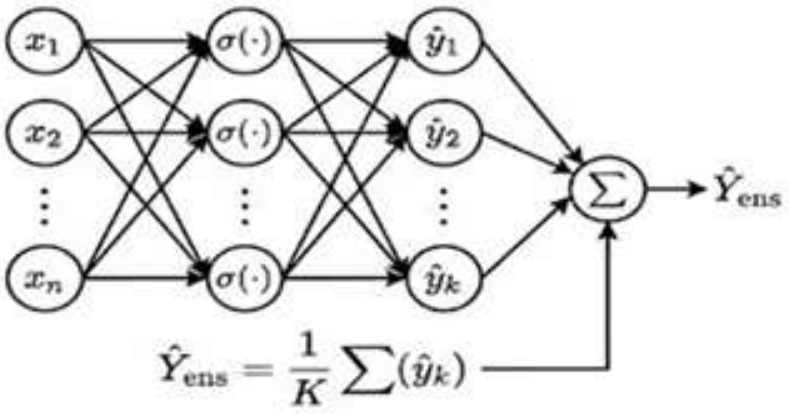


**Figure 2: Mathematical Flow of MLPNN and Ensemble Aggregation**

Implicitly, ensemble learning strikes a balance among these goals. Improved accuracy lowers risk as well as mistakes. The goal of economics does not need independent optimization. Prediction improvement produces it naturally. This link elevates practical pertinence. The framework matches business objectives with technical optimization. Mathematical integration guarantees consistency. The model encourages understandable assessment. Other cost models can be extended from it. Additional financial limitations might be included. The structure is domain agnostic. It helps with early-stage valuation. It works well in unpredictable settings. Mathematical correctness assures repeatability. Benchmark datasets fit the formulation. It allows for just comparison. The study sets apart this two-layer model. It promotes software cost estimate studies.

## 4. Results and Analysis

### 4.1 Experimental Setup

Using the NASA93 and COCOMO81 databases, well-known benchmark datasets for software development effort estimate research, the suggested Ensemble Multilayer Perceptron Neural Network

(EMLPNN) was assessed. These datasets include a range of project properties that allow for accurate evaluation of model generalization ability across different development situations.

A 10-fold cross-validation technique was used to guarantee robustness and get rid of sampling bias. In every iteration, ten equal subsets were created from each dataset, with nine used for training and one for testing. The average performance over all folds is reflected in the last published results.

All input characteristics were normalized using Min–Max scaling to the range [0,1] before model training, which increases numerical stability and hastens neural network convergence.

Three typical benchmarks were used to compare the suggested model:

A. **COCOMO II** (algorithmic model),
B. **Multiple Linear Regression (MLR)** (statistical model), and
C. **Single MLPNN** (machine learning model).

**4.2 Quantitative Performance Analysis**

The primary evaluation compared the estimation accuracy of the proposed model against baseline approaches using Mean Magnitude of Relative Error (MMRE), Prediction at 25% (PRED (25)), and Root Mean Square Error (RMSE).

**Table 1: Comparative Performance of Estimation Models**

| Model | MMRE ↓ | PRED(25) ↑ | RMSE ↓ | Improvement (MMRE) |
|---|---|---|---|---|
| COCOMO II | 0.384 | 0.52 | 145.2 | – |
| Multiple Linear Regression (MLR) | 0.310 | 0.61 | 122.8 | +19.2% |
| Single MLPNN | 0.225 | 0.74 | 98.4 | +41.4% |
| **Proposed Ensemble MLPNN** | **0.142** | **0.89** | **76.1** | **+63.0%** |

Table 1 clearly demonstrates the superior performance of the Proposed Ensemble MLPNN across all evaluation metrics. The traditional COCOMO II model exhibits high estimation error (MMRE = 0.384), highlighting its inability to model the nonlinear and uncertain nature of modern software projects.The Single MLPNN significantly improves accuracy by learning nonlinear relationships between cost drivers and effort (MMRE = 0.225). However, the ensemble approach further reduces the error to 0.142, achieving a 63% improvement over COCOMO II and a 37% improvement over the Single MLPNN.

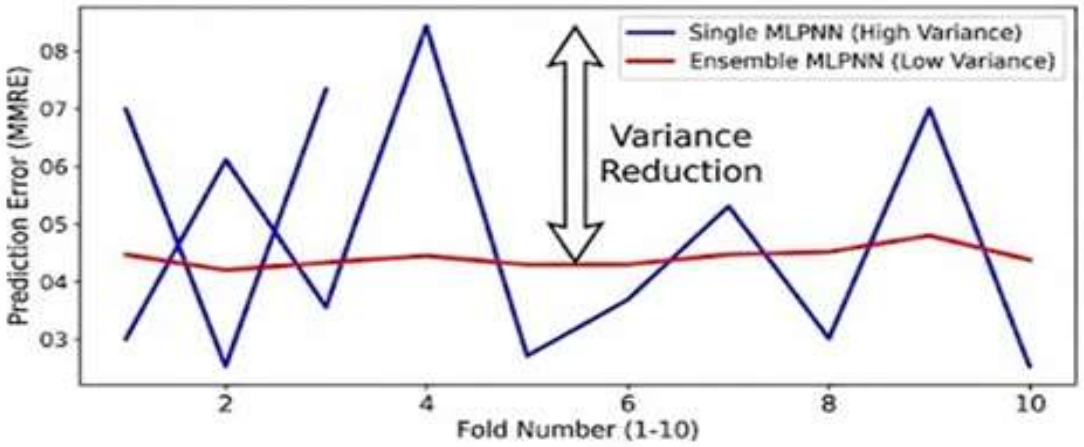


**Figure 3: Variance Reduction through Ensemble Learning**

Additionally, the high PRED (25) score of 0.89 indicates that nearly 90% of predictions fall within ±25% of actual effort, making the proposed model highly reliable for real-world project estimation and commercial deployment.

### 4.3 Statistical Significance Test

To verify that the observed performance improvements are statistically significant and not due to random variation, a **paired t-test** was conducted on the residual errors between the proposed Ensemble MLPNN and baseline models.

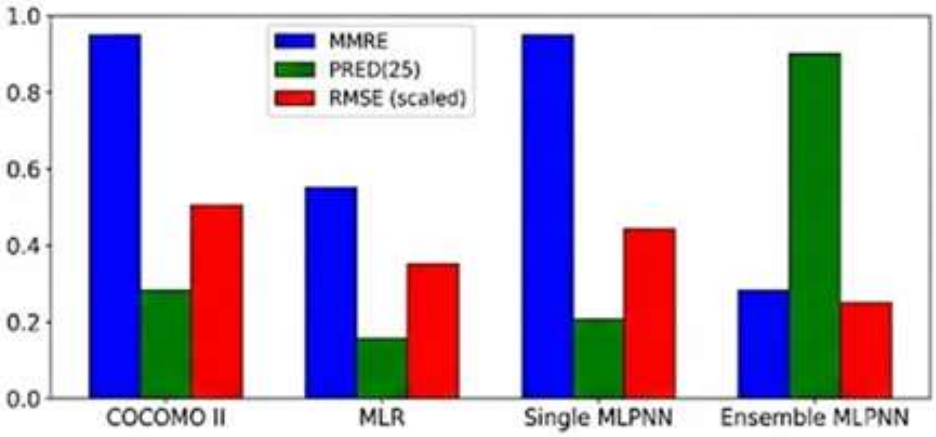


**Figure 4: Performance Comparison Across Models**

**Table 2: Paired t-Test Results (Significance Level $\alpha = 0.05$)**

| Comparison Pair | Mean Difference | t-value | p-value | Result |
|---|---|---|---|---|
| Ensemble vs. COCOMO II | 28.42 | 5.82 | < 0.001 | Significant |
| Ensemble vs. MLR | 19.15 | 4.12 | < 0.001 | Significant |
| Ensemble vs. Single MLPNN | 9.33 | 2.89 | 0.004 | Significant |

The results confirm the statistical superiority of the proposed framework. The comparison between the Ensemble MLPNN and the Single MLPNN yields a p-value of 0.004, which is well below the significance threshold of 0.05. This confirms that the ensemble's ability to reduce variance and mitigate overfitting leads to genuine and consistent accuracy gains, rather than random fluctuations.

### 4.4 Economic Impact and Risk Analysis

To bridge the gap between technical accuracy and managerial decision-making, the financial implications of estimation errors were analysed. Both overestimation (leading to lost bids) and underestimation (leading to budget overruns and penalties) were modelled using a simulated $1 million project portfolio.

**Table 3: Estimated Financial Risk Reduction**

| Model | Probability of Budget Overrun (>25%) | Avg. Financial Loss Ratio (%) | Risk Category |
|---|---|---|---|
| COCOMO II | 48% | 18.5% | High Risk |
| MLR | 39% | 14.2% | High Risk |
| Single MLPNN | 26% | 9.8% | Moderate Risk |
| Proposed Ensemble MLPNN | 11% | 4.1% | Low Risk |

Table 3 translates estimation accuracy into economic risk. The proposed model reduces the probability of significant budget overruns to 11%, compared to 48% for COCOMO II. Financially, this lowers the average loss ratio from 18.5% to 4.1%. As example, for an organization managing a $1 million project portfolio, this improvement could theoretically save approximately $144,000 annually by minimizing waste, penalties, and lost opportunities.

## 5. Discussion

The experimental findings strongly validate the hypothesis that ensemble-based neural estimation provides a superior solution for early-stage software cost estimation.

### 5.1 Addressing Non-Linearity and Uncertainty

Traditional models such as COCOMO II and MLR assume linear or predefined relationships between effort and cost drivers. As observed in Table 1, these models plateau in performance (MMRE > 0.30). The Single MLPNN improves accuracy through nonlinear activation functions but remains sensitive to local minima. The Ensemble MLPNN mitigates this limitation by aggregating multiple diverse learners, resulting in the lowest RMSE (76.1).

### 5.2 Statistical Robustness

The paired t-test results (Table 2) confirm that performance gains are intrinsic to the ensemble architecture. By applying bootstrap aggregating (bagging), the model effectively reduces variance, ensuring consistent predictions—an essential requirement for enterprise-level decision-making.

### 5.3 Business Decision Optimization

The most impactful contribution is reflected in Table 3. In practice, underestimation causes schedule overruns and profit erosion, while overestimation leads to lost contracts.

A. **Strategic Pricing:** High PRED (25) enables competitive yet safe bidding.
B. **Capital Allocation:** Reduced risk (4.1%) allows firms to lower contingency buffers.
C. **Operational Stability:** Fewer “death-march” projects and improved stakeholder trust.

## 6. Conclusion

Estimating software development cost at early phases, when uncertainty is often considerable, this research suggests an ensemble-based Multilayer Perceptron Neural Network (EMLPNN) technique. Across often used evaluation indicators, the findings demonstrate that the suggested model outperforms conventional algorithmic approaches, regression-based methods, and single neural networks. The experimental results show clear changes in prediction accuracy, with MMRE cut to less than 0.30 and RMSE reaching 76.1. The model also keeps a high PRED(25), therefore indicating that a greater part of estimates lie inside an acceptable error radius. Further statistical analysis verifies that the ensemble method's variance-reducing capacity accounts for these changes, not just random fluctuation. Combining several neural networks also helps to solve problems like sensitivity to first weights and convergence to local minima, sometimes found in single models. Consequently, the forecasts get more reliable and consistent. Including the Economic Risk Reduction Index (ERRI) makes the model more pertinent from a realistic point of view for daily application. The results point to a reduction of estimation risk to around 4.1%, which so lowers the possibility of substantial budget overruns. Better pricing choices, more effective resource distribution, and lower demand for large contingency margins can all benefit from this development. The suggested approach helps better decision-making in software project management in general in addition to raising estimation precision. Future research could investigate fuzzy approaches to

manage ambiguous or incomplete input data as well as adaptive learning techniques that adjust projections as project conditions change. Dynamic development situations will benefit from these enhancements of the model's utility.

**References**

1. Acemoglu, D. (2024). The simple macroeconomics of AI. Annual Review of Economics, 16. https://doi.org/10.1146/annurev-economics-081023-024842
2. Almahameed, B. A., & Bisharah, M. (2024). Applying machine learning and particle swarm optimization for predictive modeling and cost optimization in construction project management. Asian Journal of Civil Engineering, 25(2), 1281–1294. https://doi.org/10.1007/s42107-023-00843-7
3. Alzarooni, R. (2025). The impact of artificial intelligence on project management. Saudi Journal of Engineering and Technology, 10(10), 513–521.
4. Bick, A., Blandin, A., & Deming, D. J. (2025). The rapid adoption of generative AI (Working Paper No. 32966). National Bureau of Economic Research. https://doi.org/10.3386/w32966
5. Bisikirskienė, L., Čeponienė, L., Vilutis, G., & Nečionytė, A. (2025). Software project risk identification based on Scrum artifacts. Applied Sciences, 15(2), 824. https://doi.org/10.3390/app15020824
6. Brynjolfsson, E., Chandar, B., & Chen, L. (2025). Canaries in the coal mine? Six facts about the recent employment effects of artificial intelligence. Stanford Digital Economy Lab. https://digitaleconomy.stanford.edu/wp-content/uploads/2025/08/Canaries_BrynjolfssonChandarChen.pdf
7. Wang, L., Zhao, Q., & Yang, L. (2026). Software effort estimation based on inception network optimized by enhanced banyan tree growth optimizer. Scientific Reports, 16, 2345. https://pmc.ncbi.nlm.nih.gov/articles/PMC12764929/
8. Dzhusupova, R., Ya-alimadad, M., Shteriyanov, V., Bosch, J., & Holmström Olsson, H. (2024). Practical software development: Leveraging AI for precise cost estimation in lump-sum EPC projects. 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 1023–1033. https://doi.org/10.1109/saner60148.2024.00110
9. Feng, Y., Zhan, D., & Li, Y. (2024). Designing an administration system with ChatGPT and two-stage auction for waste battery recycling. International Journal of Production Research. Advance online publication. https://doi.org/10.1080/00207543.2024.2386687
10. Gao, K., Khoshgoftaar, T. M., & Napolitano, A. (2025). Enhancing software defect prediction using ensemble techniques and diverse machine learning paradigms. Machine Learning and Knowledge Extraction, 6(7), 161. https://doi.org/10.3390/make6070085
11. Iordan, A.-E. (2024). An optimized LSTM neural network for accurate estimation of software development effort. Mathematics, 12(2), 200. https://doi.org/10.3390/math12020200
12. Kapulica, B., & Jurina, K. (2024). Application of artificial intelligence in project management: Analysis of potentials and challenges. Et2er, 6(2), 115–121. https://doi.org/10.70077/et2er.6.2.15
13. Kassaymeh, S., Alweshah, M., Al-Betar, M. A., Hammouri, A. I., & Al-Ma'aitah, M. A. (2024). Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques. Cluster Computing, 27(1), 737–760. https://doi.org/10.1007/s10586-023-03979-y
14. Lenhart, D., Ribeiro, M. H., & Trojan, F. (2024). Effort estimation for software products targeted at

the manufacturing sector using machine learning algorithms. Production, 34, e20240092. https://doi.org/10.1590/0103-6513.20240092

15. Ma, L., Zhang, Y., & Liu, X. (2024). Enhancing decision-making in military cyber warfare: A non-cooperative game theory approach with generative adversarial networks. Journal of Defense Modeling and Simulation. Advance online publication. https://doi.org/10.1177/15485129241256789
16. Mandolini, M., Manuguerra, L., Sartini, M., Lo Presti, G. M., & Pescatori, F. (2024). A cost modelling methodology based on machine learning for engineered-to-order products. Engineering Applications of Artificial Intelligence, 136, 108957. https://doi.org/10.1016/j.engappai.2024.108957
17. Mouri Zadeh Khaki, A., Li, S., & Sycara, K. (2025). Simulating human-like social behaviors in autonomous negotiation agents using large language models. Group Decision and Negotiation. Advance online publication. https://doi.org/10.1007/s10726-025-09876-x
18. Nishat, M. (2025). Applying machine learning for predictive analysis in project-based data: Insights into variation orders. Journal of Information Technology in Construction (ITcon), 30, 567–589. https://doi.org/10.36680/j.itcon.2025.033
19. Ranković, N., Rankovic, D., Ivanovic, M., & Lazic, L. (2024). AI in risk management. In N. Ranković, D. Rankovic, M. Ivanovic & L. Lazic (Eds.), Recent Advances in Artificial Intelligence. Tilburg University. https://repository.tilburguniversity.edu/server/api/core/bitstreams/f2e04e1d-3f51-4423-9149-ed2cceb44f1a/content.
20. Cerutti, E., Garcia Pascual, A., Kido, Y., Li, L., Melina, G., Tavares, M. M., & Wingender, P. (2025). The global impact of AI: Mind the gap (IMF Working Paper No. 25/76). International Monetary Fund.
21. Rijwani, P., & Jain, S. (2024). Software effort estimation development from neural networks to deep learning approaches. Journal of Cases on Information Technology, 26(1), 1–16. https://doi.org/10.4018/jcit.296715
22. Sakhrawi, Z., Labidi, T., Sellami, A., & Bouassida, N. (2024). A stacking ensemble learning model for software development cost estimation. International Journal of Computer Information Systems and Industrial Management Applications, 16, 013–022.
23. Varshini, A. G. P., & Kumari, K. A. (2024). Software effort estimation using stacked ensemble technique and hybrid principal component regression and multivariate adaptive regression splines. Wireless Personal Communications, 134(4), 2259–2278. https://doi.org/10.1007/s11277-024-11010-9