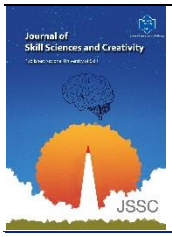


Journal of Skill Sciences and Creativity
National University of Skill
Winter 2026, Vol. 2, No. 2, p. 13-28
Journal Homepage: https://jssc.tvu.ac.ir/?lang=en
:10.48301/jssc.2026.562170.1043

Optimizing Software Cost Estimation with MLP Hyperparameter Tuning Using Grid Search

Saba Beiranvand*[1], Kazem Taghandiki[2]

[1] Department of Computer Engineering, Technical and Vocational University (TVU), Tehran, Iran.
[2] Department of Computer Engineering, Technical and Vocational University (TVU), Tehran, Iran.

ARTICLE INFO

Article Type:
Original Research

Received: 14.10.2025
Revised: 10.12.2025
Accepted: 15.12.2025

Keyword:
Software Cost Estimation
Machine Learning
MLP Neural Network
Hyperparameter Optimization
Grid Search

***Corresponding Author:**
Kazem Taghandiki
Email: *ktaghandiki@tvu.ac.ir*

ABSTRACT

Software cost estimation plays a critical role in software project management, as inaccurate predictions can lead to budget overruns, resource shortages, or project failure. Over the years, several approaches have been proposed to improve estimation accuracy, with machine learning and data mining techniques attracting significant attention. Among these, the Multi-Layer Perceptron (MLP) neural network is widely applied due to its ability to capture complex nonlinear relationships. However, the predictive performance of MLP is highly sensitive to hyperparameter settings, making proper optimization essential. This study employs the Grid Search technique to optimize MLP hyperparameters for software cost estimation. Experiments were conducted on six benchmark datasets frequently used in this domain: Desharnais, Maxwell, Kemerer, Albrecht, COCOMO81, and COCONASA. Performance was evaluated using three well-established metrics: MMRE (Mean Magnitude of Relative Error), PRED(0.25), and EF (Error Function). The results demonstrate that Grid Search-based optimization significantly improves MLP performance, reducing MMRE while increasing PRED(0.25) and EF across all datasets. These findings highlight the importance of systematic hyperparameter tuning in enhancing model accuracy and reliability, and they confirm the potential of optimized MLP models for practical software cost estimation tasks.

E-ISSN: 3060-6691

EXTENDED ABSTRACT

Introduction

Accurate software cost estimation (SCE) is one of the most critical challenges in software engineering and project management. Reliable cost prediction allows managers to allocate resources effectively, schedule tasks realistically, and ensure that projects are delivered within budget. Conversely, poor estimation can result in cost overruns, project delays, or even total failure. Over the past decades, several approaches have been introduced for SCE, ranging from traditional algorithmic models such as COCOMO and SLIM, to expert-based assessments and, more recently, machine learning (ML) and data mining techniques.

Among ML methods, neural networks have proven effective due to their capacity for nonlinear modeling and pattern learning. Specifically, the Multi-Layer Perceptron (MLP) has been extensively applied for regression tasks, including effort and cost prediction. MLP's flexibility in learning complex mappings between input attributes (e.g., software size, complexity, and productivity factors) and project effort makes it particularly valuable. However, one major limitation is its sensitivity to hyperparameters, such as the number of layers, neurons, learning rate, activation functions, and batch size. Inappropriate parameter settings may lead to overfitting, underfitting, or slow convergence.

To address this limitation, the present study investigates the use of Grid Search, a systematic and exhaustive hyperparameter optimization technique, to enhance MLP performance in software cost estimation. The research aims to answer the following key question: Can Grid Search-based tuning significantly improve the accuracy and reliability of MLP models across multiple software cost datasets?

Methodology

The research methodology consists of four major stages: data preprocessing, model construction, hyperparameter optimization, and performance evaluation.

1. Data Collection and Preprocessing

Six benchmark datasets widely used in software engineering cost estimation research were employed: Desharnais, Maxwell, Kemerer, Albrecht, COCOMO81, and COCONASA. These datasets vary in project types, number of attributes, and effort measurement units, offering a comprehensive testbed. Prior to modeling, each dataset underwent standard preprocessing steps:

- Normalization: All numeric features were scaled to a [0,1] range to improve convergence stability.
- Missing Value Handling: Incomplete records were either removed or imputed based on statistical distributions.
- Outlier Detection: Extreme values were identified using the interquartile range (IQR) method and addressed to prevent bias.

2. Model Construction

The MLP model was implemented as a feedforward neural network trained via the backpropagation algorithm. The architecture included one or two hidden layers, depending on dataset complexity, and employed various activation functions such as ReLU, sigmoid, and tanh. The output layer contained a single neuron with a linear activation function to predict software effort.

3. Hyperparameter Optimization Using Grid Search

Grid Search was applied to systematically explore predefined combinations of hyperparameters. Each combination was evaluated using k-fold cross-validation to ensure robust and unbiased performance. The search space included:

- Number of hidden layers: 1–3
- Neurons per layer: 5–100
- Learning rate: 0.0001–0.1
- Batch size: 8–64
- Activation functions: ReLU, tanh, sigmoid
- Epochs: 50–500

For each configuration, the MLP model was trained and evaluated. The best-performing set of parameters was selected based on the lowest Mean Magnitude of Relative Error (MMRE) and highest PRED (0.25) and Error Function (EF) values.

4. Evaluation Metrics

Model performance was assessed using three widely adopted metrics:

- MMRE (Mean Magnitude of Relative Error): Measures average relative deviation between predicted and actual effort values.
- PRED (0.25): Represents the percentage of predictions within 25% of the actual effort.
- EF (Error Function): A composite measure combining both accuracy and reliability indicators.

c) Results and Discussion

The experiments revealed that hyperparameter optimization using Grid Search significantly improved MLP performance across all six datasets.

1. Quantitative Improvements

- Desharnais Dataset: MMRE reduced from 0.87 (default MLP) to 0.38 (optimized MLP), while PRED (0.25) increased from 32% to 61%.
- Maxwell Dataset: MMRE decreased from 0.93 to 0.41, with EF scores improving correspondingly.
- Albrecht Dataset: PRED (0.25) rose from 25% to 54%, confirming the enhanced reliability of predictions.
- COCONASA Dataset: The most notable improvement was observed, with MMRE dropping from 5.18 to 1.18, indicating a substantial reduction in estimation error.

- Kemerer and COCOMO81 Datasets: Both showed moderate yet consistent performance gains after tuning.
- Overall, the average MMRE improvement across datasets was approximately 57%, while PRED (0.25) increased by an average of 23%, confirming the effectiveness of the optimization strategy.

2. Comparative Analysis

The optimized MLP results were compared with those of traditional algorithmic models (e.g., COCOMO) and previously published ML-based methods. The proposed approach consistently achieved better accuracy, demonstrating that parameter tuning can be as critical as model selection. Furthermore, the findings align with recent research emphasize the need for data-driven calibration rather than relying on generic model configurations.

3. Discussion

Despite the strong performance improvements, Grid Search's exhaustive nature entails high computational costs, particularly when dealing with large datasets or wide parameter ranges. Nevertheless, the method guarantees the identification of near-optimal configurations within the predefined search space, making it suitable for small to medium-scale SCE problems.

Another insight derived from the results is that the optimal hyperparameter combinations varied across datasets, indicating that no universal configuration

d) Conclusions

This research highlights the significant impact of hyperparameter optimization on improving the predictive accuracy of neural network-based software cost estimation models. By integrating Grid Search with an MLP architecture, the study achieved remarkable reductions in estimation error and improvements in consistency across multiple benchmark datasets. Key conclusions include:

1. Systematic tuning enhances accuracy: Proper adjustment of MLP parameters such as the number of neurons, the number of layers, and the learning rate substantially improves cost estimation outcomes**.**
2. Generalizability across datasets: The optimized models demonstrated reliable performance on diverse datasets, confirming the robustness of the approach.
3. Foundations for future work: The methodology provides a baseline for exploring more advanced optimization methods such as Bayesian Optimization, Genetic Algorithms, or Particle Swarm Optimization, which can reduce computation time while maintaining accuracy.

نشریه علوم مهارتی و خلاقیت
دانشگاه ملی مهارت
زمستان ۱۴۰۴، دوره۲، شماره ۲، ۱۳-۲۸
آدرس نشریه: https://jssc.tvu.ac.ir/
doi:10.48301/jssc.2026.562170.1043

بهینه‌سازی تخمین هزینه نرم‌افزار با تنظیم هایپرپارامترهای MLP از طریق Grid Search

صبا بیرانوند[۱]، کاظم تقندیکی*[۲]

۱- گروه مهندسی کامپیوتر، دانشگاه ملی مهارت، تهران، ایران.
۲- گروه مهندسی کامپیوتر، دانشگاه ملی مهارت، تهران، ایران.

اطلاعات مقاله

نوع مقاله: مقاله پژوهشی

دریافت مقاله: ۱۴۰۴/۰۷/۲۲
بازنگری مقاله: ۱۴۰۴/۰۹/۱۹
پذیرش مقاله: ۱۴۰۴/۰۹/۲۴

کلید واژگان:
تخمین هزینه نرم‌افزار
یادگیری ماشین
شبکه عصبی پرسپترون چندلایه
بهینه‌سازی پارامترها
Grid Search

***نویسنده مسئول:** کاظم تقندیکی
پست الکترونیکی:
ktaghandiki@tvu.ac.ir

چکیده

تخمین هزینه نرم‌افزار نقشی اساسی در مدیریت پروژه‌های نرم‌افزاری دارد، زیرا برآوردهای نادرست می‌تواند منجر به کمبود منابع، افزایش هزینه‌ها یا حتی شکست پروژه شود. در سال‌های اخیر، روش‌های متعددی برای بهبود دقت تخمین ارائه شده‌اند که در این میان، الگوریتم‌های یادگیری ماشین و داده‌کاوی توجه ویژه‌ای را به خود جلب کرده‌اند. شبکه عصبی پرسپترون چندلایه (MLP) به دلیل توانایی در مدل‌سازی روابط پیچیده و غیرخطی، یکی از مدل‌های پرکاربرد در این حوزه است. بااین‌حال، عملکرد این شبکه به شدت به تنظیم مناسب هایپرپارامترها وابسته است. در این پژوهش، از روش جست‌وجوی شبکه‌ای (Grid Search) برای بهینه‌سازی هایپرپارامترهای شبکه MLP در تخمین هزینه نرم‌افزار استفاده شد. آزمایش‌ها روی شش مجموعه داده‌ی مرجع شامل Desharnais، Maxwell، Kemerer، Albrecht، COCOMO81 و COCONASA انجام گرفت. برای ارزیابی عملکرد، از سه معیار شناخته‌شده شامل MMRE، PRED(۰,۲۵) و EF استفاده شد. نتایج نشان داد که بهینه‌سازی هایپرپارامترها با استفاده از Grid Search منجر به کاهش قابل‌توجه خطای MMRE و بهبود معیارهای PRED(۰,۲۵) و EF در تمامی مجموعه داده‌ها شده است. این یافته‌ها بر اهمیت تنظیم سیستماتیک پارامترها در افزایش دقت و قابلیت اعتماد مدل‌های یادگیری ماشین در تخمین هزینه نرم‌افزار تأکید دارد.

شاپای الکترونیکی: ۶۶۹۱-۳۰۶۰

مقدمه

تخمین هزینه‌ی نرم‌افزار[1] (SCE) بخش مهمی از فرآیند توسعه‌ی پروژه‌ی نرم‌افزاری است. در این فعالیت، پیش‌بینی هزینه‌ی مورد نیاز برای توسعه و نگهداری یک محصول نرم‌افزاری بر پایه‌ی دیگر ویژگی‌های پروژه صورت می‌پذیرد. اصطلاح تخمین هزینه‌ی نرم‌افزار در اغلب کارهای پژوهشی معادل تخمین تلاش توسعه‌ی نرم‌افزار به کار می‌رود. SCE چنانچه در شروع پروژه‌ای با دقت بالایی صورت نپذیرد ممکن است آن پروژه در اواسط کار محکوم به شکست شود. از سال ۱۹۸۰ تاکنون روش‌های مختلفی برای تخمین هزینه‌ی نرم‌افزار ارائه شده‌اند. روش‌هایی که توسط محققان برای SCE مورد استفاده قرار گرفته‌اند را می‌توان در سه دسته روش‌های (۱) الگوریتمی، (۲) مبتنی بر قضاوت خبرگان[2] و (۳) مبتنی بر روش‌های یادگیری ماشین [3] تقسیم نمود.

در روش‌های الگوریتمی، تخمین هزینه‌ی نرم‌افزار براساس رابطه‌ای به شکل رابطه‌ی ۱ صورت می‌پذیرد.

$$EFFORT=F(x_1,x_2,x_3,\dots) \quad (1)$$

در رابطه‌ی ۱، پارامترهای $x_1, x_2, x_3, \dots$ بیانگر ویژگی‌های تشریح‌کننده‌ی هر نمونه پروژه‌ی نرم‌افزاری هستند. تخمین تلاش مبتنی بر مدل‌های الگوریتمی مختلف، معمولاً به طور قابل توجهی متفاوت از سایر مدل‌هاست. در این روش‌ها مدلی براساس الگوریتمی مشخص فرموله شده و از آن فرمول جهت SCE استفاده می‌شود. تاکنون مدل‌های الگوریتمی مختلفی برای SCE ارائه شده است. مدل هزینه COCOMO[4] معروف‌ترین مدل از بین مدل‌های ارائه شده الگوریتمی است. این مدل، روشی پایه‌ای است که از آن برای پیش‌بینی تعداد افراد مورد نیاز در هر ماه برای توسعه‌ی نرم‌افزار، استفاده می‌شود. این مدل هم‌چنین قادر است تخمینی از زمان توسعه در ماه را ارائه دهد. با کمک این مدل، می‌توان میزان تلاش مورد نیاز در هر فاز توسعه‌ی نرم‌افزار را هم تخمین زد [۱]. از آنجاییکه مدل‌های الگوریتمی مبتنی بر داده‌های قدیمی هستند، از این‌رو نمی‌توانند پیشرفت‌های فعلی در زبان‌های برنامه‌نویسی، سخت‌افزار و مهندسی نرم‌افزار را در تخمین صحیح هزینه نرم‌افزار بازتاب دهند [۲].

روش‌های مبتنی بر قضاوت خبرگان، دسته‌ی دوم از روش‌های تخمین هزینه‌ی نرم‌افزار می‌باشند. معمولاً داده‌های قدیمی برای قضاوت خبرگان نیاز نیستند. قضاوت خبرگان، اغلب مبتنی بر استفاده‌ی مجدد از تخمینگرِ پروژه‌های قبلی است که ممکن است مستند نشده باشند. نتایج بدست آمده بیانگر آن است که ۶۲ درصد از تخمینگران در سازمان‌ها، از این روش استفاده می‌کنند [۳]. مزیت این نوع از روش‌ها آن است که برای فرهنگِ سازمانی خاصی، سفارشی شده‌اند. پس در مقایسه با رویکرد الگوریتمی، دقیق‌تر خواهند بود و در خیلی از موارد ثابت شده است که دقت بهتری نسبت به سایر مدل‌ها ارائه داده‌اند. ولی این تخمین کاملا ذهنی و براساس منطقی شخصی استوار است. بنابراین مزیت آن را می‌توان به‌عنوان عیب آن هم در نظر گرفت. چراکه هر خبره فقط براساس تجربیات و فرهنگی که در آن سازمانِ خاص وجود دارد هزینه را تخمین میزند و همان خبره در سازمان دیگر ممکن است با دقتی پایین‌تر، هزینه‌ی نرم‌افزار را تخمین بزند [۲].

روش‌های یادگیری ماشین، الگوهایی را از داده‌های پروژه‌های قدیمی یاد می‌گیرند و از این الگوها برای پیش‌بینی تلاش استفاده می‌کنند. ایده‌ی اساسی استفاده از روش‌های یادگیری ماشین برای تخمین و پیش‌بینی تلاش، این است که دادگان قدیمی، شامل پروژه‌های قدیمی زیادی هستند که با ویژگی‌های با ارزششان برای توصیف هر پروژه، تشریح شده‌اند. پروژه‌های با ویژگی‌های مشابه، ممکن است شامل تلاش‌های پروژه‌ی تقریبا مشابهی نیز باشند. کار روش‌های یادگیری ماشین، یادگیری الگوهای ذاتی بین مقادیر ویژگی‌ها و ارتباطات بین آن‌ها و تلاش پروژه است که می‌تواند برای

[1] Software Cost Estimation

[2] Expert Judgment Based

[3] Machine Learning

[4] Constructive Cost Model

پیش‌بینی تلاش پروژه‌ای جدید، استفاده شود [۲]. تاکنون متخصصان و کارشناسان زیادی، از روش‌های یادگیری ماشین مختلفی در این زمینه استفاده نموده و به دقت‌های قابل توجهی دست یافته‌اند [۳-۷].

بنابراین با رویکرد یادگیری ماشین و با داشتن اطلاعاتی درباره‌ی پروژه‌هایی که تاکنون توسعه پیدا کرده‌اند می‌توان به مدلی دست یافت که امکان تخمین میزان تلاش مورد نیاز را برای پروژه‌های جدید فراهم می‌کند[۸, ۹]. با توجه به وجود عدم قطعیت در تخمین نرم‌افزاری، استفاده از روش‌های غیرقطعی و انعطاف‌پذیر یادگیری ماشین، می‌تواند نقش به‌سزایی در افزایش دقت تخمین داشته باشد. مزیت‌های این نوع از روش‌های تخمین، شامل توانایی مدل‌های هوش محاسباتی برای مدل کردن مجموعه‌ای پیچیده از ارتباطات بین تلاش و ویژگی‌های پروژه با یادگیری از داده‌های پروژه‌های گذشته است [۵].

مطالعه‌ی نظام‌مند [10] با مرور ۳۰۴ مقاله در حوزه‌ی تخمین هزینه و تلاش نرم‌افزار، کسب دانش از منابعِ متنوع و گسترده را هدف قرار داده است. این مقالات در ۷۶ مجله مختلف طبقه‌بندی شده‌اند و بر مبنای موضوع پژوهش، روش تخمین، رویکرد پژوهشی، زمینه‌ی مطالعه و داده‌های مورد استفاده دسته‌بندی شده‌اند. از نکات برجسته این مرور، ارائه‌ی توصیه‌هایی برای پژوهش‌های آینده است. از جمله مهم‌ترین این توصیه‌ها؛ گسترش دامنه جستجو برای منابع پژوهشی، استفاده از جستجوی دستی در مجلات منتخب در صورت نیاز به جامعیت، تمرکز پژوهش‌ها بر روش‌های رایج در صنعت نرم‌افزار، و توجه ویژه به تأثیر ویژگی‌های داده‌ها بر ارزیابی روش‌های تخمین است.

تخمین تلاش توسعه‌ی نرم‌افزار [۱] یا SDEE، فرآیند پیش‌بینی تلاش مورد نیاز برای توسعه‌ی یک سیستم نرم‌افزاری است. SDEE را نه تنها می‌توان شامل تلاش توسعه‌ی نرم‌افزار دانست بلکه می‌توان آن را در برگیرنده‌ی تلاش مورد نیاز، جهت نگهداری نرم‌افزار نیز به شمار آورد. تخمین دقیق تلاش مورد نیاز در مراحل اولیه‌ی توسعه‌ی نرم‌افزار نقش بسیار مهمی در مدیریت پروژه بر عهده دارد [۱۱]. بر اساس پژوهش [۱۲]، از سال ۱۹۸۰، یعنی برهه‌ای از زمان که تحولات اساسی در زمینه‌ی تخمین تلاش نرم‌افزار رخ داده است، بسیاری از روش‌های تخمین در زمینه‌ی SDEE ارائه شده‌اند که از میان آن‌ها روش مبتنی بر قضاوت متخصصان و روش‌های یادگیری ماشین بیشتر مورد استفاده قرار گرفته‌اند. در سال‌های اخیر روش‌های مبتنی بر یادگیری ماشین در این زمینه، بیشتر مورد توجه قرار گرفته‌اند [۱۱].

در میان الگوریتم‌های یادگیری ماشین، شبکه‌های عصبی مصنوعی به دلیل توانایی مدل‌سازی روابط پیچیده میان متغیرهای ورودی و خروجی، یکی از روش‌های پرکاربرد برای تخمین هزینه‌ی نرم‌افزار محسوب می‌شوند. با این حال، عملکرد این مدل‌ها به شدت وابسته به انتخاب بهینه‌ی پارامترها است. انتخاب نامناسب پارامترهای مدل، ممکن است منجر به بیش‌برازش یا کم‌برازش شود که در نهایت دقت پیش‌بینی را کاهش می‌دهد. از این رو، بهینه‌سازی پارامترهای یادگیری ماشین، به‌ویژه در شبکه‌های عصبی، نقش مهمی در بهبود دقت تخمین هزینه‌ی نرم‌افزار ایفا می‌کند[۱۳].

یکی از روش‌های متداول برای تنظیم بهینه‌ی پارامترهای مدل‌های یادگیری ماشین، جستجوی شبکه‌ای است. در این روش، مجموعه‌ای از مقادیر ممکن برای پارامترها تعریف شده و مدل با ترکیب‌های مختلف این پارامترها آموزش داده می‌شود. در نهایت، ترکیبی که بهترین عملکرد را از نظر معیارهای ارزیابی داشته باشد، انتخاب می‌شود. در این پژوهش، از روش Grid Search برای تنظیم بهینه‌ی پارامترهای شبکه‌ی عصبی MLP استفاده شده است. هدف اصلی این کار، بهبود دقت مدل در پیش‌بینی هزینه‌ی نرم‌افزار با تنظیم مناسب پارامترهایی مانند تعداد لایه‌های مخفی، تعداد نرون‌ها در هر لایه، نرخ یادگیری و تابع فعال‌سازی است [۱۴].

بر این اساس، در ادامه مقاله ابتدا در بخش دوم به مرور بخشی از پژوهش‌های صورت گرفته در زمینه SCE خواهیم پرداخت. در بخش سوم روش پیشنهادی تشریح می‌شود، سپس در بخش چهارم نتایج آزمایش‌ها ارائه خواهد شد و در بخش پنجم به نتیجه‌گیری و پیشنهاد کارهای آتی پرداخته خواهد شد.

[1] Software Development Effort Estimation

مروری بر کارهای گذشته

با اینکه در سال‌های اخیر با هدف بالا بردن دقت، استفاده از مدل‌های یادگیری ماشین در تخمین تلاش توسعه‌ی نرم‌افزار، بیشتر مورد توجه قرار گرفته است اما هیچ کدام از مدل‌های موجود در تمام شرایط، مناسب نیستند و از دادگانی به دادگان دیگر، دقت متفاوتی را ارائه می‌دهند. بنابراین نیازی اساسی به ایجاد مدلی که قابلیت اطمینان بالایی برای نمونه‌های جدید داشته باشد، وجود دارد. براساس مروری که در این پژوهش بر مطالعات صورت گرفته در زمینه‌ی SCE از جمله [۱۱] انجام شده است از سال ۱۹۹۱ تاکنون هشت دسته الگوریتم یادگیری ماشین شامل بر استدلال مبتنی بر مورد[۱] (CBR)، شبکه‌های عصبی مصنوعی[۲](ANN)، درخت تصمیم[۳](DT)، شبکه‌های بیزین[٤](BN)، رگرسیون بردار پشتیبان[٥](SVR)، الگوریتم ژنتیک[٦] (GA)، برنامه‌نویسی ژنتیک[۷](GP)، قوانین وابستگی[۸](AR) در حوزه‌ی SCE مورد استفاده واقع شده‌اند. مقالات متعددی تاکنون از روش‌های یادگیری ماشین برای SCE استفاده نموده‌اند. نتایج حاصل از پژوهش‌هایی که در زمینه‌ی انجام روش‌های مختلف پیش‌پردازش در این حوزه انجام شده‌اند نشان‌دهنده‌ی موثر بودن این عمل در راستای افزایش دقت و رسیدن به دقت بهتر می‌باشند [۷, ۱۱, ۱۲].

علاوه بر این، اخیرا پژوهشگران زیادی به مرور پژوهش‌های انجام شده در حوزه تخمین هزینه نرم‌افزار پرداخته‌اند که همگی نشان از اهمیت و نقش تاثیرگذار پیش‌بینی دقیق هزینه توسعه نرم‌افزار در ابتدای مسیر توسعه و موفقیت پروژه است [۱۵-۱۷].

پژوهش [18] یک بررسی جامع از روش‌ها و مدل‌های تخمین هزینه و تلاش توسعه نرم‌افزار ارائه می‌دهد. در این مطالعه، مدل‌ها و تکنیک‌های مورد استفاده در SCE در دسته‌بندی‌هایی مانند مدل‌های پارامتریک، تکنیک‌های مبتنی بر خبره، روش‌های یادگیری‌گرایانه، مدل‌های مبتنی بر دینامیک، روش‌های رگرسیونی، روش‌های مبتنی بر منطق فازی، مدل‌های مبتنی بر اندازه، و تکنیک‌های ترکیبی بررسی شده‌اند. این مرور جامع به پژوهشگران آینده کمک می‌کند که با یک منبع واحد دیدی کلی و کامل نسبت به روش‌های مختلفSCE پیدا کنند. یکی از نتایج کلیدی این مطالعه این است که هیچ تکنیک یا مدلی به‌تنهایی برای تمامی شرایط مناسب نیست؛ هر رویکردی دارای معایب و مزایای خاص خود بوده و تحت تأثیر تحولات سریع صنعت نرم‌افزار قرار می‌گیرد. به همین دلیل، پژوهشگران پیشنهاد می‌کنند به‌جای اتکا به یک روش منفرد، از رویکردهای ترکیبی استفاده شود تا محدودیت‌های یک مدل توسط مزایای مدل دیگر جبران گردد. همچنین تأکید شده که کالیبراسیون مدل‌ها برای استفاده در محیط‌های متفاوت، ضروری است، چرا که مدل‌های توسعه‌یافته در یک بستر خاص، ممکن است در محیطی جدید قابل اعتماد نباشند.

در پژوهش [۱۹]، یک مدل ترکیبی پیشرفته برای تخمین هزینه نرم‌افزار ارائه شد که شبکه عصبی پیچشی[۹] (CNN)را با الگوریتم بهینه‌سازی ازدحام ذرات[۱۰] (PSO) ترکیب می‌کند. این مدل در بستر پیش‌بینی سری‌های زمانی طراحی شده که امکان استخراج ویژگی‌های پیچیده و تنظیم خودکار هایپرپارامترها را فراهم می‌سازد و از تلاش دستی برای تنظیم پارامترها می‌کاهد. استفاده از PSO به تقویت قابلیت تعمیم و پایداری مدل CNN کمک کرده و از طریق فرآیندهای تکراری، تنظیم بهینه‌های پارامتری را به صورت کارآمد تسهیل می‌کند.

[1] Case-Based Reasoning
[2] Artificial Neural Networks
[3] Decision Tree
[4] Bayesian Networks
[5] Support Vector Regression
[6] Genetic Algorithm
[7] Genetic Programming
[8] Association Rule
[9] Convolutional Neural Network
[10] Particle Swarm Optimization

نویسندگان در [۲۰] برای بهبود دقت تخمین در مدل COCOMO II از الگوریتم‌های فراابتکاری استفاده کرده‌اند. ابتدا الگوریتم دلفین برای تنظیم پارامترهای مدل به‌کار گرفته شده و سپس یک الگوریتم ترکیبی جدید از دلفین و خفاش به نام DolBat معرفی شده است.. نتایج بررسی مدل‌های ارائه شده در این پژوهش نشان می‌دهد که الگوریتم دلفین عملکرد بهتری نسبت به روش‌های قبلی دارد، اما الگوریتم ترکیبی DolBatبهترین دقت را در بهینه‌سازی ضرایب مدل COCOMO II به دست آورده است و می‌تواند تخمین هزینه نرم‌افزار را قابل اعتمادتر سازد.

پژوهشگرانی در [۲۱] با تمرکز و تاکید بر محدودیت روش‌های پارامتریک کلاسیک مانند COCOMO-II در پردازش دقیق داده‌ها، یک رویکرد بهینه‌سازی مبتنی بر الگوریتم گرده‌افشانی گل‌ها[1] (FPA) ارائه کرده‌اند. در این روش، پارامترهای مدل COCOMO-II با کمک FPA روی دادگان واقعی از صنعت نرم‌افزار ترکیه تنظیم می‌شوند. نتایج آزمایش‌ها نشان می‌دهد که این الگوریتم در مقایسه با مدل اصلی COCOMO-II و همچنین الگوریتم خفاش، عملکرد بهتری داشته و خطای تخمین را کاهش داده است.

برآورد اشتباه می‌تواند منجر به هدررفت منابع، شکست پروژه و کاهش ارزش آن شود. نویسندگان در مطالعه [۱۵] با توجه به محدودیت روش‌های الگوریتمی و غیرالگوریتمی سنتی، یک رویکرد جدید مبتنی بر الگوریتم گرده‌افشانی گل‌ها را برای تخمین هزینه نرم‌افزار پیشنهاد داده‌اند. برای ارزیابی، معیار MMRE استفاده شده و عملکرد FPA با مدل پایه COCOMO مقایسه گردیده است. آزمایش‌ها نشان داد که روش پیشنهادی FPA بهبود قابل‌توجهی نسبت به COCOMO دارد.

در پژوهش [۳] روش جدیدی برای افزایش دقت در تخمین هزینه نرم‌افزار شده که مبتنی بر دو اصل کلیدی خوشه‌بندی داده‌ها و انتخاب ویژگی مؤثر است. ابتدا داده‌های پروژه‌های نرم‌افزاری با استفاده از الگوریتم K-Means خوشه‌بندی می‌شوند تا نمونه‌های همگن در هر خوشه قرار گیرند. سپس برای هر خوشه به‌صورت جداگانه، ترکیبی از روش‌های فیلتر و بسته‌بندی برای انتخاب ویژگی موثر اعمال می‌شود. ترکیبی بهینه‌سازی دقت در این پژوهش با استفاده از یک معیار ارزیابی انجام می‌شود.

همانطور که در پژوهش‌های مختلف نشان داده شده است، استفاده از روش‌های ترکیبی در حوزه های مختلف یادگیری ماشین از جمله تخمین هزینه نرم‌افزار می‌تواند باعث کاهش نقاط ضعف این دسته از روش‌ها گردد. با توجه به اهمیت روزافزون بهینه‌سازی پارامترهای مدل‌های یادگیری ماشین، این پژوهش به ارائه رویکردی مبتنی بر Grid Searchبرای تنظیم بهینه‌ی هایپرپارامترهای شبکه عصبی MLP در تخمین هزینه نرم‌افزار می‌پردازد. هدف اصلی آن است که با بهبود دقت مدل در پیش‌بینی هزینه‌ها، گامی در جهت ارتقای فرآیندهای تصمیم‌گیری در مدیریت پروژه‌های نرم‌افزاری و تخصیص منابع برداشته شود.

روش پیشنهادی

تخمین دقیق هزینه نرم‌افزار یکی از حیاتی‌ترین مراحل مدیریت پروژه‌های نرم‌افزاری است، چرا که مسیر موفقیت یا شکست یک پروژه را تعیین می‌کند. وقتی بتوانیم به درستی منابع، زمان و بودجه مورد نیاز را پیش‌بینی کنیم، نه تنها از هدررفت سرمایه و انرژی جلوگیری می‌شود، بلکه امکان برنامه‌ریزی هوشمندانه، تخصیص مناسب نیروی انسانی و تصمیم‌گیری به موقع برای اصلاح مسیر پروژه فراهم می‌شود. تخمین اشتباه هزینه توسعه پروژه نرم‌افزاری، حتی در پروژه‌های کوچک، می‌تواند منجر به تأخیر در تحویل، کاهش کیفیت محصول و در نهایت از دست رفتن اعتماد مشتری شود. به همین دلیل، داشتن ابزارها و روش‌های بهینه برای برآورد دقیق هزینه، نه یک گزینه بلکه یک ضرورت برای هر

1 Flower Pollination Algorithm

سازمان توسعه دهنده نرم‌افزار است که می‌خواهد در رقابت بازار موفق باشد و پروژه‌های خود را با امنیت و اطمینان پیش برد.

تاکنون روش‌های مختلفی برای تخمین دقیق هزینه ارائه شده‌اند، اما الگوریتم‌های یادگیری ماشین و داده‌کاوی توجه زیادی را به خود جلب کرده‌اند. در میان این الگوریتم‌ها، شبکه عصبی پرسپترون چندلایه به دلیل قابلیت یادگیری روابط پیچیده و مدل‌سازی الگوهای غیرخطی، به‌عنوان یکی از مدل‌های پرکاربرد شناخته می‌شود. با این حال، عملکرد این مدل به شدت وابسته به تنظیم صحیح هایپرپارامترها است. در این پژوهش، به‌منظور بهبود دقت تخمین هزینه نرم‌افزار، از روش جستجوی شبکه‌ای برای تنظیم بهینه‌ی پارامترهای MLP استفاده شده است. شبکه عصبی چندلایه یکی از مدل‌های پرکاربرد در یادگیری عمیق است که از چندین لایه‌ی پنهان برای مدل‌سازی روابط پیچیده میان ورودی‌ها و خروجی‌ها استفاده می‌کند [۲۲]. این شبکه‌ها از سه نوع لایه اصلی تشکیل شده‌اند:

- لایه ورودی: داده‌ها یا ویژگی‌های پروژه‌ی نرم‌افزاری در این لایه وارد شبکه می‌شوند.
- لایه‌های پنهان: این لایه‌ها مسئول پردازش اطلاعات و استخراج ویژگی‌های مفهومی از داده‌ها هستند. تعداد این لایه‌ها و تعداد نورون‌ها در هر لایه از مهم‌ترین پارامترهای شبکه هستند که باید بهینه‌سازی شوند.
- لایه خروجی: در این لایه پیش‌بینی نهایی مدل ارائه می‌شود، که در اینجا می‌تواند تخمین هزینه‌ی نرم‌افزار باشد.
- فرآیند آموزش شبکه عصبی MLP به‌طور کلی شامل مراحل زیر است [۲۳]:
- انتخاب تابع هزینه (Loss Function): این تابع مسئول اندازه‌گیری تفاوت بین پیش‌بینی‌های مدل و مقادیر واقعی است.
- انتخاب الگوریتم بهینه‌سازی (Optimization Algorithm): الگوریتم‌هایی مانند گرادیان کاهشی و انواع مختلف آن مانند Adam برای بهینه‌سازی وزن‌های شبکه استفاده می‌شوند.
- تنظیم پارامترهای مدل: برای بهبود دقت مدل، پارامترهایی مانند تعداد لایه‌ها، تعداد نورون‌ها در هر لایه، و نرخ یادگیری باید بهینه‌سازی شوند.

جستجوی شبکه‌ای یکی از ساده‌ترین و در عین حال مؤثرترین روش‌ها برای بهینه‌سازی پارامترهای مدل‌های یادگیری ماشین است [۲۴, ۲۵]. در این روش، یک فضای جستجو برای پارامترهای مدل تعریف می‌شود و سپس تمامی ترکیب‌های ممکن از مقادیر پارامترها آزمایش می‌شوند. به عبارت دیگر، جستجوی شبکه‌ای به‌صورت سیستماتیک همه ترکیب‌های ممکن از مقادیر پارامترها را بررسی کرده و بهترین ترکیب برای تعداد لایه‌های مخفی، تعداد نورون‌ها در هر، نرخ یادگیری و نوع تابع فعال‌سازی را انتخاب می‌کند. سپس، مدل با تمامی ترکیب‌های مختلف این پارامترها آموزش داده می‌شود و ارزیابی می‌شود. بهترین ترکیب پارامترها که کمترین مقدار تابع هزینه را داشته باشد، به عنوان بهترین مدل انتخاب می‌شود. از مزایای جستجوی شبکه‌ای می‌توان به مواردی هم چون سادگی و دقت (این روش به سادگی قابل اجرا است و به‌طور سیستماتیک تمامی ترکیب‌های ممکن را بررسی می‌کند.)، قابلیت استفاده در مدل‌های پیچیده (در مدل‌های پیچیده‌ای مانند شبکه‌های عصبی که چندین پارامتر قابل تنظیم دارند، جستجوی شبکه‌ای می‌تواند انتخاب بهینه را در هر مورد مشخص انجام دهد.) اشاره کرد. با این حال، محدودیت‌های جستجوی شبکه‌ای عبارتند از:

- هزینه محاسباتی بالا: این روش می‌تواند بسیار زمان‌بر باشد، به‌ویژه وقتی که فضای جستجو شامل مقادیر زیاد برای هر پارامتر باشد.
- عدم کارایی در جستجوی فضای بسیار بزرگ: در فضای جستجوهای با ابعاد زیاد، جستجوی شبکه‌ای ممکن است بهینه‌ترین ترکیب را نیابد.

در پژوهش حاضر، برای غلبه بر چالش تنظیم دستی پارامترها و ارتقای دقت مدل، از جستجوی شبکه‌ای به منظور بهینه‌سازی هایپرپارامترهای شبکه عصبی MLP استفاده شده است. در این چارچوب، فضای جستجوی مشخصی برای پارامترهایی چون تعداد لایه‌های پنهان، تعداد نورون‌ها در هر لایه، نرخ یادگیری و نوع تابع فعال‌سازی تعریف گردید. سپس تمامی ترکیب‌ها به‌صورت سیستماتیک مورد بررسی قرار گرفتند و مدلی که بهترین عملکرد را بر اساس معیارهای ارزیابی ارائه می‌داد، انتخاب شد.

بدین ترتیب، روش پیشنهادی علاوه بر بهره‌گیری از توانایی MLP در مدل‌سازی روابط غیرخطی، از مزیت بهینه‌سازی سیستماتیک پارامترها برخوردار است. هرچند جستجوی شبکه‌ای از نظر محاسباتی پرهزینه است، اما در مقایسه با روش‌های متداول‌تر تنظیم دستی پارامترها، رویکردی پایدارتر و قابل اعتمادتر برای بهبود دقت در تخمین هزینه نرم‌افزار فراهم می‌آورد .شبه کد روش پیشنهادی در الگوریتم ۱ آمده است.

نتایج ارزیابی

در این بخش، نتایج ارزیابی مدل شبکه عصبی پرسپترون چندلایه برای تخمین هزینه نرم‌افزار بر روی چندین مجموعه داده ارائه شده است. برای بررسی تأثیر بهینه‌سازی هایپرپارامترها، ابتدا مدل MLP با مقادیر پیش‌فرض اجرا شده و سپس با استفاده از روش جستجوی شبکه‌ای، پارامترهای مدل تنظیم شده‌اند. در نهایت، برای ارزیابی دقت مدل، از روش 10-fold استفاده شده و میانگین معیارهای)۰,۲۵(PRED، MMRE و EF گزارش شده است. در ادامه ابتدا به معرفی دادگان مورد استفاده در مرحله ارزیابی پرداخته شده و سپس معیارهای ارزیابی مورد استفاده به صورت کلی شرح داده خواهند شد. در ادامه به ارائه نتایج بدست آمده خواهیم پرداخت.

دادگان مورد استفاده

در این پژوهش از چندین مجموعه داده مختلف برای ارزیابی مدل‌های تخمین هزینه نرم‌افزار استفاده شده است. این دادگان به شرح زیر هستند:

دادگان Albrecht

این مجموعه داده شامل ۲۴ پروژه نرم‌افزاری است که توسط سازمان خدمات پردازش داده شرکت IBM توسعه یافته‌اند. سیستم‌های موجود در این دادگان با زبان‌های برنامه‌نویسی DMS، COBOL و PL/I پیاده‌سازی شده‌اند. این دادگان دارای ۶ متغیر مستقل شامل تعداد ورودی‌ها، تعداد خروجی‌ها، تعداد پرس‌وجوها، تعداد فایل‌ها، فانکشن پوینت‌ها (FP) و خطوط کد (LOC) است. متغیر وابسته این دادگان، میزان تلاش موردنیاز برای توسعه نرم‌افزار است که بر حسب تعداد ساعت کاری اندازه‌گیری شده است. پیچیدگی و اندازه پروژه‌ها نیز با استفاده از روش Function Point ارائه‌شده توسط آلبرشت محاسبه شده است.

الگوریتم ۱: تخمین هزینه نرم‌افزار با استفاده از شبکه عصبی پرسپترون چندلایه (MLP) و جستجوی شبکه‌ای (Grid Search)

Inputs:

D = {(x_i, y_i)}: Dataset with features and target cost
G = Hyperparameter search space
L: Number of hidden layers
H: Number of neurons per layer
LR: Learning rate
ACT: Activation function
BS: Batch size
EP: Number of training epochs
K = Number of folds for cross-validation
Metric = Evaluation metric (e.g., RMSE, MMRE)

Outputs:

g*: Optimal hyperparameter combination
θ*: Final trained model parameters

Steps:

1. **Data Preprocessing:**
   - Handle missing values
   - Correct or remove outliers
   - Normalize features
2. **Split the dataset into training (Train) and testing (Test) sets**
3. **Initialization:**
   best score∞
   g* = None
4. **For each hyperparameter combination g in G:**

a) **Split training data into K folds**

b) **For each fold k:**
   - Define training subset D_tr and validation subset D_val
   - Initialize MLP with hyperparameters g
   - Train the model using an optimizer (e.g., Adam or SGD)
   - Compute evaluation metric on D_val

c) **Compute the mean evaluation metric across K folds**

d) **If score < best score:**
   g* ← g
   best score ← score

5. **Retrain MLP on full training set using g***
6. **Evaluate final model on the test set**
7. **Return g* and θ***

دادگان Maxwell

این مجموعه داده شامل ۶۲ پروژه نرم‌افزاری است که از بانک‌های بزرگ فنلاند جمع‌آوری شده‌اند. این دادگان دارای ۲۶ متغیر مستقل مرتبط با ویژگی‌های مختلف نرم‌افزار مانند نوع کاربرد، اندازه و مشخصات پروژه هستند. متغیر وابسته این دادگان، میزان تلاش توسعه نرم‌افزار است که بر حسب تعداد ساعت کاری اندازه‌گیری شده است. این پروژه‌ها در بازه زمانی ۱۹۸۵ تا ۱۹۹۳ اجرا شده‌اند. معیار اندازه‌گیری اندازه پروژه‌ها در این دادگان Function Point است.

دادگان COCOMO۸۱

این دادگان شامل ۶۳ پروژه نرم‌افزاری از انواع مختلف، شامل سیستم‌های تجاری، علمی، سیستمی، بلادرنگ و حمایتی است. این مجموعه شامل ۱۶ متغیر مستقل است که ویژگی‌هایی مانند خصوصیات محصول، پروژه، سخت‌افزار و افراد درگیر را توصیف می‌کنند. متغیر وابسته آن، تلاش توسعه نرم‌افزار است که بر حسب ساعت کاری در هر فرد اندازه‌گیری شده است. اندازه پروژه‌ها در این دادگان براساس خطوط کد (LOC) محاسبه شده است.

دادگان Desharnais

این دادگان شامل ۸۱ پروژه نرم‌افزاری است که توسط ده سازمان مختلف در کانادا در بازه زمانی ۱۹۸۳ تا ۱۹۸۸ اجرا شده‌اند. این دادگان دارای ۱۲ ویژگی مستقل است که شامل تجربه تیم، تجربه مدیر، مدت‌زمان پروژه، تعداد معاملات، تعداد موجودیت‌ها، تیم فانکشن پوینت (FP)، محیط توسعه و زبان برنامه‌نویسی است. متغیر وابسته آن، میزان تلاش توسعه نرم‌افزار است که بر حسب ساعت کاری در هر فرد اندازه‌گیری شده است. اندازه پروژه‌ها در این دادگان با استفاده از Function Point محاسبه شده است.

دادگان Kemerer

این دادگان شامل ۱۵ پروژه نرم‌افزاری است که در سال ۱۹۸۵ جمع‌آوری شده‌اند. قدیمی‌ترین پروژه موجود در این مجموعه در سال ۱۹۸۱ آغاز شده است و بیشتر پروژه‌های آن مربوط به سال ۱۹۸۳ هستند. این دادگان دارای ۸ متغیر مستقل است و اندازه پروژه‌ها براساس KSLOC (هزاران خط کد) در دو سطح متوسط و بزرگ قرار گرفته‌اند.

دادگان COCONASA

این دادگان شامل ۹۳ پروژه نرم‌افزاری است که توسط سازمان ناسا توسعه داده شده‌اند. این مجموعه شامل ۱۷ متغیر مستقل است که ویژگی‌های مختلف پروژه‌های نرم‌افزاری را پوشش می‌دهند.

معیارهای ارزیابی

برای ارزیابی دقت مدل‌های تخمین هزینه نرم‌افزار، از معیارهای مختلفی استفاده شده است. این معیارها شامل موارد زیر هستند:

میانگین خطای نسبی (MRE - Mean Relative Error)

این معیار میزان خطای نسبی در پیش‌بینی‌های مدل را نسبت به مقدار واقعی اندازه‌گیری می‌کند. فرمول آن به‌صورت رابطه ۱ است که در آن Actual Effort مقدار واقعی تلاش و Estimated Effort مقدار تخمینی تلاش است.

$$MRE = \frac{|\text{Actual Effort} - \text{Estimated Effort}|}{\text{Actual Effort}} \times 100 \quad (۱)$$

میانگین قدر مطلق خطای نسبی(MMRE - Mean Magnitude Relative Error)

این معیار مشابه MRE است، اما مقدار قدر مطلق میانگین را در نظر می‌گیرد تا از تأثیر منفی مقادیر مثبت و منفی جلوگیری شود. فرمول آن به‌صورت زیر است که در آن n تعداد کل نمونه‌ها است.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|Estimated(i) - Actual(i)|}{Actual(i)}\right) \quad (۲)$$

دقت پیش‌بینی(PRED - Prediction at p%)

این معیار نشان می‌دهد که چه درصدی از پیش‌بینی‌ها در یک بازه p% از مقدار واقعی قرار دارند. فرمول آن به‌صورت زیر است که در آن k تعداد پیش‌بینی‌هایی است که مقدار خطای آن‌ها کمتر از p٪ است و n تعداد کل نمونه‌ها است. مقدار p معمولاً ۲۵٪ در نظر گرفته می‌شود.

$$PRED(x) = \frac{k}{n} \quad (۳)$$

تابع خطا(EF - Error Function)

این معیار برای سنجش میزان دقت مدل به‌صورت کلی استفاده می‌شود و بر اساس نسبت مقدار PRED و MMRE محاسبه می‌شود. فرمول آن به‌صورت زیر است که در آن (۰,۲۵) PRED نشان‌دهنده درصد پیش‌بینی‌هایی است که در محدوده ۲۵٪ از مقدار واقعی قرار دارند.

$$EF = \frac{PRED(0.25)}{1 + MMRE} \quad (۴)$$

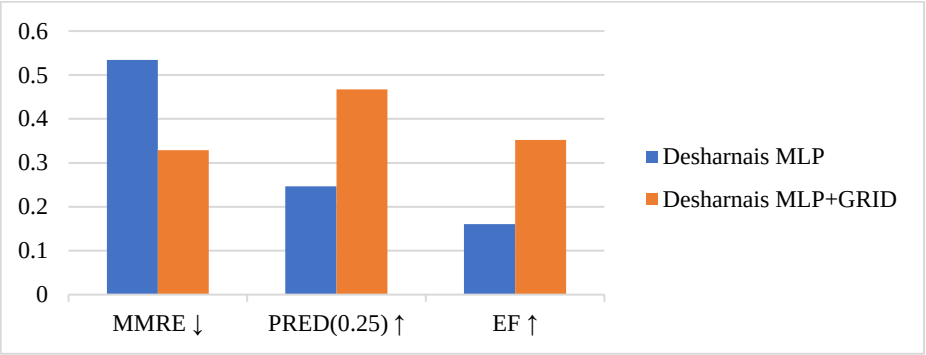
نتایج بدست آمده

روش پیشنهادی ما بر روی یک سیستم دسکتاپ با پردازنده Intel Core i5 و ۱۲ گیگابایت حافظه رم اجرا شد. نتایج به‌دست آمده نشان داد که مدل پیشنهادی در عمل، عملکرد قابل اعتمادی دارد و توانایی پردازش داده‌های استاندارد پروژه‌های نرم‌افزاری را با دقت و کارایی مناسب فراهم می‌کند. جدول ۱ مقادیر به‌دست‌آمده برای هر دادگان را نشان می‌دهد.

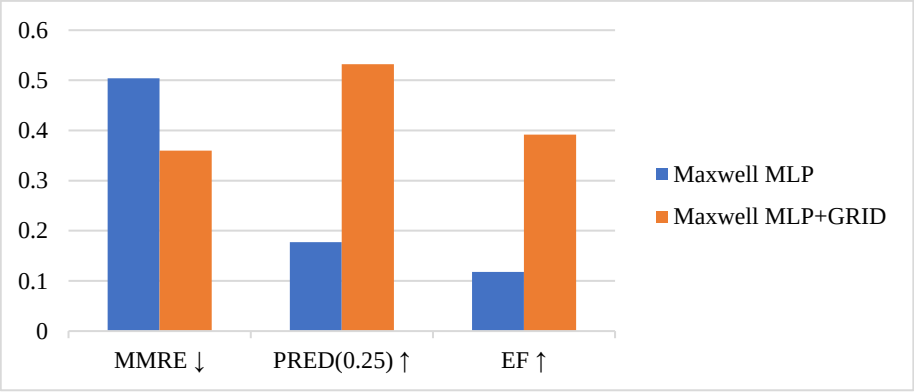
جدول ۱. نتایج ارزیابی روش پیشنهادی

Dataset	Model	↓ MMRE	↑ PRED)۰/۲۵(	↑ EF
Desharnais	MLP	۰/۵۳۴۱	۰/۲۴۶۸	۰/۱۶۰۸
	MLP+GRID	۰/۳۲۸۵	۰/۴۶۷۵	۰/۳۵۱۹
Maxwell	MLP	۰/۵۰۴۴	۰/۱۷۷۴	۰/۱۱۷۹
	MLP+GRID	۰/۳۵۹۷	۰/۵۳۲۳	۰/۳۹۱۵
Kemerer	MLP	۰/۹۸۰۹	۰/۱۴۲۹	۰/۰۷۲۱
	MLP+GRID	۰/۷۳۱۵	۰/۳۵۷۱	۰/۲۰۶۳
Albrecht	MLP	۱/۳۷۸۶	۰/۲۵۰۰	۰/۱۰۵۱
	MLP+GRID	۰/۴۹۰۰	۰/۵۴۱۷	۰/۳۶۳۵
COCOMO۸۱	MLP	۲/۳۵۷۳	۰/۱۴۲۹	۰/۰۴۲۶
	MLP+GRID	۰/۸۵۱۹	۰/۲۲۲۲	۰/۱۲۰۰
COCONASA	MLP	۵/۱۸۰۹	۰/۱۶۴۸	۰/۰۲۶۷
	MLP+GRID	۱/۱۸۵۳	۰/۲۸۵۷	۰/۱۳۰۷

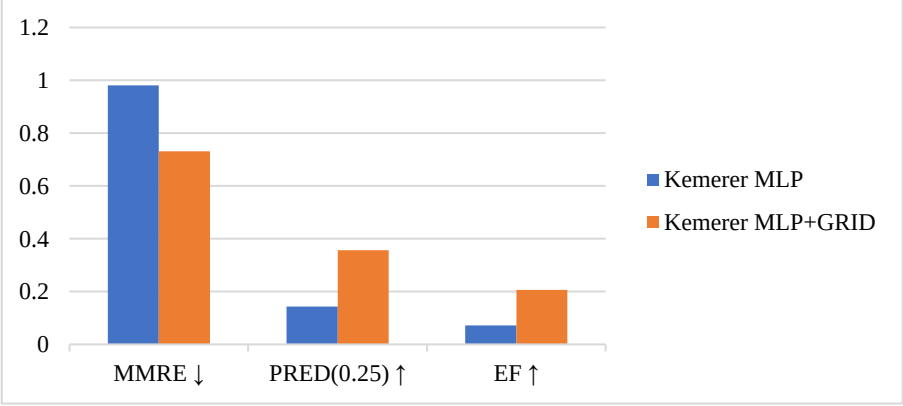
در ادامه در قالب شکل های ۱ تا ۶ به مقایسه بصری نتایج حاصل از ارزیابی روش پیشنهادی پرداخته شده است. همانطور که مشاهده میشود روش پیشنهادی تاثیر بسیار مثبتی روی کاهش خطای تخمین داشته است.



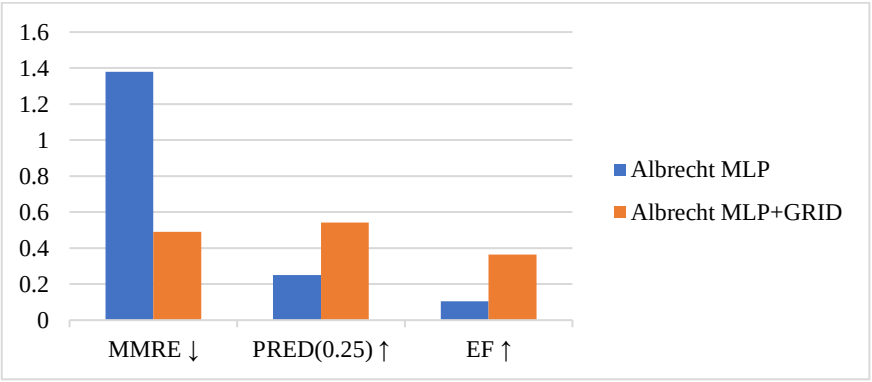
شکل ۱. مقایسه بصری نتایج ارزیابی روی دادگان Desharnais



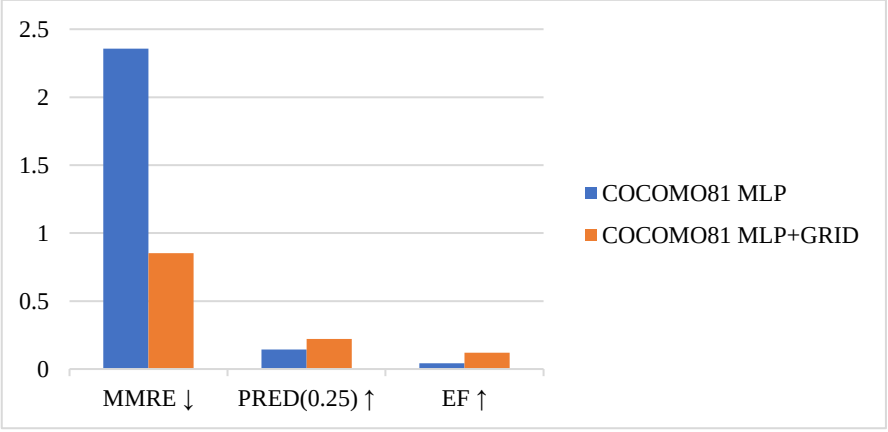
شکل ۲. مقایسه بصری نتایج ارزیابی روی دادگان Maxwell



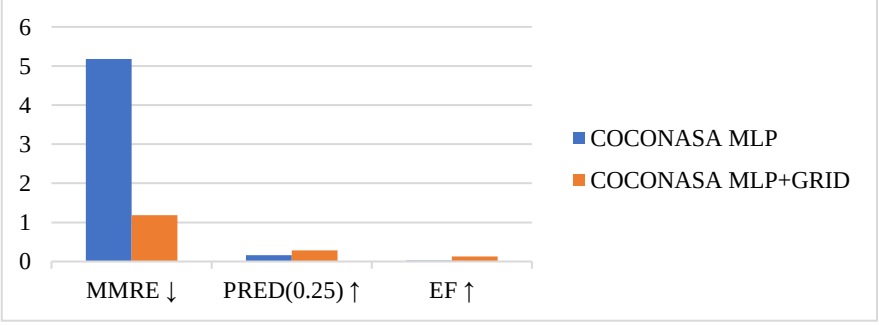
شکل ۳. مقایسه بصری نتایج ارزیابی روی دادگان Kemerer



شکل ۴) مقایسه بصری نتایج ارزیابی روی دادگان Albrecht



شکل ۵. مقایسه بصری نتایج ارزیابی روی دادگان COCOMO۸۱



شکل ۶. مقایسه بصری نتایج ارزیابی روی دادگان COCONASA

نتایج نشان می‌دهد که روش جستجوی شبکه‌ای به‌طور قابل توجهی دقت تخمین هزینه نرم‌افزار را بهبود داده است. مهم‌ترین مشاهدات عبارتند از:

- مقدار MMRE در تمامی مجموعه داده‌ها پس از بهینه‌سازی کاهش یافته است، که نشان‌دهنده‌ی کاهش خطای نسبی مدل می‌باشد.

- مقدار)۰,۲۵(PRED در مدل بهینه‌شده برای تمامی دادگان افزایش یافته است، به این معنی که درصد بیشتری از پیش‌بینی‌ها در بازه‌ی ۲۵٪ مقدار واقعی قرار دارند.
- مقدار EF که کیفیت کلی مدل را ارزیابی می‌کند، پس از بهینه‌سازی افزایش یافته است، که نشان‌دهنده‌ی بهبود دقت مدل می‌باشد.
- بیشترین بهبود در دادگان COCONASA مشاهده شد که MMRE آن از ۵,۱۸ به ۱,۱۸ کاهش یافته است. همچنین،)۰,۲۵(PRED آن از ۱۶,۴۸٪ به ۲۸,۵۷٪ افزایش یافته است.
- دادگان Albrecht نیز بهبود قابل توجهی داشته است، به طوری که MMRE آن از ۱,۳۷ به ۰,۴۹ کاهش یافته و مقدار)۰,۲۵(PRED آن از ۲۵٪ به ۵۴,۱۷٪ رسیده است.

بهینه‌سازی هایپرپارامترهای شبکه عصبی با استفاده از روش جستجوی شبکه‌ای توانسته است دقت مدل را برای تخمین هزینه نرم‌افزار به میزان چشمگیری بهبود بخشد. این موضوع اهمیت تنظیم دقیق پارامترهای مدل را برای بهبود نتایج در تخمین هزینه نشان می‌دهد.

مقایسه نتایج

به‌منظور ارزیابی کارایی و اعتبار روش پیشنهادی، نتایج حاصل بر روی چندین مجموعه‌داده متداول در حوزه تخمین هزینه نرم‌افزار با مطالعات پیشین مقایسه شده است. جدول ۲ نتایج این مقایسه را بر اساس معیارهای متداول ارزیابی نشان می‌دهد. همان‌طور که مشاهده می‌شود، روش پیشنهادی در اغلب مجموعه‌داده‌ها عملکرد بهتری نسبت به کارهای پیشین ارائه کرده است. به‌ویژه در مجموعه‌داده‌های Desharnais، Maxwell و Albrecht بهبود قابل‌توجهی در هر سه معیار حاصل شده است، که نشان‌دهنده توانایی بالای مدل در کاهش خطا و افزایش دقت پیش‌بینی است. هرچند در برخی مجموعه‌داده‌ها مانند COCOMO81 و COCONASA همچنان چالش‌هایی وجود دارد، اما روند کلی نتایج حاکی از تأثیر مثبت و برتری نسبی رویکرد پیشنهادی است.

جدول ۲. مقایسه نتایج حاصل از روش پیشنهادی با سایر پژوهش‌ها

Dataset	Model	↓ MMRE	↑ PRED)۰/۲۵(	↑ EF
Desharnais	[۲۶]	۰,۳۸	۰,۲۲	-
	This paper	۰,۳۲۸۵	۰,۴۶۷۵	۰,۳۵۱۹
Maxwell	[۱۱]	-	۰,۵	-
	This paper	۰,۳۵۹۷	۰,۵۳۲۳	۰,۳۹۱۵
Kemerer	[۲۷]	۱,۰۹	۰,۲۵	-
	This paper	۰,۷۳۱۵	۰,۳۵۷۱	۰,۲۰۶۳
Albrecht	[۲۷]	۰,۶۹۹۱	۰,۵۰	-
	This paper	۰,۴۹۰۰	۰,۵۴۱۷	۰,۳۶۳۵
COCOMO۸۱	[۲۷]	۱,۵۹	۰,۰۶	-
	This paper	۰,۸۵۱۹	۰,۲۲۲۲	۰,۱۲۰۰
COCONASA	[۲۷]	۲,۵	۰,۱۲	-
	This paper	۱,۱۸۵۳	۰,۲۸۵۷	۰,۱۳۰۷

نتیجه‌گیری و کارهای پیش‌رو

در این پژوهش، عملکرد مدل MLP در تخمین هزینه نرم‌افزار روی چندین دادگان استاندارد ارزیابی شد. در مرحله‌ی اولیه، مدل MLP با مقادیر پیش‌فرض هایپرپارامترها اجرا شد و سپس با استفاده از روش جستجوی شبکه‌ای، مقادیر بهینه برای هایپرپارامترها انتخاب گردید. نتایج نشان داد که استفاده از روش بهینه‌سازی هایپرپارامترها تأثیر قابل توجهی در بهبود دقت مدل دارد. معیارهای MMRE، PRED(0.25) و EF برای ارزیابی عملکرد مدل مورد استفاده قرار گرفتند.

با مقایسه نتایج، مشاهده شد که مدل تنظیم شده توسط Grid Search توانست خطای تخمین را در تمامی دادگان کاهش دهد و دقت پیش‌بینی را به میزان قابل توجهی بهبود بخشد. همچنین، میزان بهبود برای دادگان مختلف متفاوت بود؛ به عنوان مثال، دادگان‌هایی مانند Desharnais و Maxwell بیشترین بهبود را در دقت مدل تجربه کردند، در حالی که دادگان COCOMO81 و COCONASA همچنان چالش‌های زیادی در تخمین هزینه نرم‌افزار داشتند. این موضوع نشان می‌دهد که برخی از مجموعه داده‌ها ممکن است به مدل‌های پیچیده‌تر یا پیش‌پردازش داده‌های قوی‌تری نیاز داشته باشند. با توجه به نتایج به دست آمده، چندین مسیر برای تحقیقات آینده پیشنهاد می‌شود:

- بهینه‌سازی هایپرپارامترها با روش‌های پیشرفته‌تر
- استفاده از روش‌هایی مانند Bayesian Optimization یا Genetic Algorithm برای تنظیم بهتر پارامترهای مدل.
- استفاده از مدل‌های یادگیری عمیق پیشرفته‌تر
- ترکیب MLP با شبکه‌های عصبی پیچیده‌تر مانند CNN یا استفاده از Transformer-based modelsبرای بهبود دقت تخمین.
- بهبود فرآیند انتخاب ویژگی
- بررسی تأثیر روش‌های انتخاب ویژگی مبتنی بر یادگیری ماشین مانند LASSO Regression یا Recursive Feature Elimination (RFE) برای کاهش ابعاد داده و بهبود دقت مدل.

- بررسی روش‌های ترکیبی
- استفاده از مدل‌های ترکیبی مانند XGBoost, Bagging & Boosting Random Forest, برای افزایش دقت پیش‌بینی و کاهش واریانس مدل.
- تحلیل بیشتر روی دادگان با کیفیت پایین‌تر
- بررسی روش‌های پیش‌پردازش داده و حذف نویز برای دادگان چالش‌برانگیزی مانند COCOMO81 و COCONASA که مدل روی آن‌ها عملکرد ضعیف‌تری داشته است.
- ارزیابی مدل‌های دیگر یادگیری ماشین

نتایج این پژوهش و مقایسه آن با سایر گارهای پژوهشی صورت گرفته در این حوزه، نشان داد که بهینه‌سازی مدل‌های یادگیری ماشین و استفاده از روش‌های مناسب برای تنظیم پارامترها می‌تواند نقش بسزایی در افزایش دقت تخمین هزینه نرم‌افزار داشته باشد.

References

1. Beiranvand, S. and Z. Chahooki, *Bridging the semantic gap for software effort estimation by hierarchical feature selection techniques.* Journal of AI and Data Mining, 2016. **4**(2): p. 157–168.
2. Beiranvand, S. and M.A. Zare Chahooki, *A Review on Software Cost Estimation Based on Machine Learning.* Soft Computing Journal, 2021. **5**(1): p. 36–65.
3. Beiranvand, S. and M.A. Zare Chahooki, *Accuracy improvement in software cost estimation based on selection of relevant features of homogeneous clusters.* Journal of AI and Data Mining, 2023. **11**(3): p. 453–476.
4. Elish, M.O., T. Helmy, and M.I. Hussain, *Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation.* Mathematical Problems in Engineering, 2013. **2013**(1): p. 312067.
5. Emmanuel, T., et al., *A survey on missing data in machine learning.* Journal of Big data, 2021. **8**(1): p. 140.
6. García, S., J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Vol. 72. 2015: Springer.
7. Huang, J., Y.-F. Li, and M. Xie, *An empirical analysis of data preprocessing for machine learning-based software cost estimation.* Information and software Technology, 2015. **67**: p. 108–127.
8. Taghandiki, K., *Handwritten Digit Recognition on MNIST Using Transfer Learning with VGG16.* Journal of Skill Sciences and Creativity, 2024. **1**(2): p. 45–68.
9. Hoseini, F., H. Sepehrzadeh, and M. Kheyri, *Early Detection of Breast Cancer by LSD Analysis and KNN Classification on MIAS Mammography Database.* Skilled Sciences and Creativity, 2024. **1**(1): p. 13–34.
10. Jorgensen, M. and M. Shepperd, *A systematic review of software development cost estimation studies.* IEEE Transactions on software engineering, 2006. **33**(1): p. 33–53.
11. Moradbeiky, A., *FEEM: A Flexible Model based on Artificial Intelligence for Software Effort Estimation.* Journal of AI and Data Mining, 2023. **11**(1): p. 39–51.
12. Pandey, P. *Analysis of the techniques for software cost estimation*. in *2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT)*. 2013. IEEE.

13. Rashid, C.H., et al., *Software cost and effort estimation: current approaches and future trends.* IEEE Access, 2023. **11**: p. 99268–99288.
14. Sheikhan, M. and M. Abbasnejad Arabi, *Joint Optimization of Structure and Parameters of Neural Networks Using Hybrid Gravitational Search Algorithms in Classification and Function Approximation Tasks.* Soft Computing Journal, 2021. **3**(2): p. 2–19.
15. Khan, B., et al., *Software cost estimation using flower pollination algorithm.* Journal of Internet Technology, 2020. **21**(5): p. 1243–1251.
16. Saleem, M.A., et al., *Systematic literature review of identifying issues in software cost estimation techniques.* International Journal of Advanced Computer Science and Applications, 2019. **10**(8).
17. Chirra, S.M.R. and H. Reza, *A survey on software cost estimation techniques.* Journal of Software Engineering and Applications, 2019. **12**(6): p. 226.
18. Rashid, J., et al., *Study of software development cost estimation techniques and models.* Mehran University Research Journal of Engineering & Technology, 2020. **39**(2): p. 413–431.
19. Draz, M.M., O. Emam, and S.M. Azzam, *Software cost estimation predication using a convolutional neural network and particle swarm optimization algorithm.* Scientific Reports, 2024. **14**(1): p. 13129.
20. Fadhil, A.A., R.G. Alsarraj, and A.M. Altaie, *Software cost estimation based on dolphin algorithm.* IEEE Access, 2020. **8**: p. 75279–75287.
21. Ullah, A., et al. *A novel technique of software cost estimation using flower pollination algorithm*. in *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*. 2019. IEEE.
22. Lin, S., et al. *Mlp can be a good transformer learner*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.
23. El-Hassani, F.Z., et al., *A new optimization model for MLP hyperparameter tuning: modeling and resolution by real-coded genetic algorithm.* Neural Processing Letters, 2024. **56**(2): p. 105.
24. Zhao, Y., W. Zhang, and X. Liu, *Grid search with a weighted error function: Hyper-parameter optimization for financial time series forecasting.* Applied Soft Computing, 2024. **154**: p. 111362.
25. Kanwar, M., B. Pokharel, and S. Lim, *A new random forest method for landslide susceptibility mapping using hyperparameter optimization and grid search techniques.* International Journal of Environmental Science and Technology, 2025: p. 1–16.
26. Zhang, W., et al., *Dimensionality reduction and machine learning based model of software cost estimation.* Frontiers in Physics, 2024. **12**: p. 1324719.
27. Rehman, I.U., Z. Ali, and Z. Jan, *An empirical analysis on software development efforts estimation in machine learning perspective.* 2021.