UDC 004.8:519.86

P. O. KACHMAR
Ivan Franko National University of Lviv

DESIGNING LOSS FUNCTIONS FOR TRANSFORMER-BASED TIME ESTIMATION IN AI PROJECT

Accurate time estimation is critical for planning and managing artificial intelligence (AI) projects. However, traditional approaches often fall short due to the domain-specific nature of the task, causing high variability and complexity in such projects. This study dives into how transformer-based models can be improved for project duration prediction by designing a custom loss function. A dataset containing structured project metadata – such as domain, stage, and difficulty – was used to train a transformer model using multiple loss functions. These include standard Mean Squared Error (MSE), a domain-weighted loss with custom metadata mapping, and an uncertainty-aware loss. The final design of the loss function integrates all benefits of having a domain knowledge baseline and the structure of the uncertainty-based loss model. Evaluation results demonstrate that this combined approach outperforms standard loss functions based on MAE, MSE, and RMSE metrics.

This study addresses the challenge of improving time estimation accuracy in artificial intelligence (AI) projects by focusing on designing custom loss functions for transformer-based regression models. Traditional estimation approaches often fail to capture the complexity and variability inherent in AI development processes, where project duration is influenced by factors such as high differences in domains and sudomains, targeted development stages, or just project difficulty. To overcome these limitations, this research looks into how loss function design can be used for better model performance.

A structured dataset of AI projects with metadata-based features like primary domain, additional domain, project stage, and difficulty score. Based on this dataset, a transformer-based model was trained and evaluated using multiple loss function strategies. These include the standard mean squared error (MSE), a domain-weighted MSE that uses and relies on expert-driven metadata mappings, an uncertainty-aware loss function, and a final stabilised loss that combines domain weighting with probabilistic constraints.

Experimental results demonstrate that the final design of a custom loss function outperforms traditional approaches across standard accuracy metrics, including mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). The findings confirm that incorporating domain knowledge and uncertainty modeling directly into the loss function leads to more accurate, stable, and interpretable predictions.

Overall, this study highlights the importance of loss function design in applied machine learning tasks and sets a framework for improving transformer-based time estimation models in AI project management contexts.

Keywords: *AI project forecasting, project time estimation, loss function design, transformer model, machine learning, uncertainty modeling, domain-weighted loss.*

П. О. КАЧМАР
Львівський національний університет імені Івана Франка

РОЗРОБКА ФУНКЦІЙ ВТРАТ ДЛЯ ТРАНСФОРМЕРНИХ МОДЕЛЕЙ ОЦІНЮВАННЯ ЧАСУ В ШІ-ПРОЄКТАХ

Точне оцінювання часу є критично важливим для планування та управління проєктами у сфері штучного інтелекту (ШІ). Однак традиційні підходи часто не забезпечують достатньої точності через доменно-специфічний характер таких завдань, що призводить до високої варіативності та складності ШІ-проєктів. У цій роботі досліджується, як можна покращити моделі на основі трансформерів для прогнозування тривалості проєктів шляхом розробки спеціалізованих функцій втрат.

Для навчання моделі було використано структурований набір даних ШІ-проєктів, що містить метадані, зокрема основний домен, додатковий домен, етап розробки та рівень складності. На основі цього набору даних було навчено трансформерну модель із застосуванням кількох підходів до побудови функцій втрат. Серед них – стандартна середньоквадратична помилка (MSE), функція втрат зважена за доменом із використанням експертних відповідностей метаданих, функція втрат із урахуванням невизначеності, а також фінальна стабілізована функція втрат, що поєднує доменне зважування з імовірнісними обмеженнями.

Експериментальні результати показали, що запропонована комбінована функція втрат перевершує традиційні підходи за основними метриками точності, зокрема середньою абсолютною похибкою (MAE), середньоквадратичною похибкою (MSE) та коренем середньоквадратичної похибки (RMSE). Отримані результати підтверджують, що інтеграція доменних знань та моделювання невизначеності безпосередньо у функцію втрат дозволяє отримати більш точні, стабільні та інтерпретовані прогнози.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-13 ISSN 2618-0332 (print), 2618-0340 (online)

Загалом, дана робота підкреслює важливість проєктування функцій втрат як ключового елементу прикладних задач машинного навчання та задає необхідну практичну основу необхідну для подальшого дослідження, тестування, та покращення моделей оцінювання часу в контексті управління ШІ-проєктами.

Ключові слова: *прогнозування ШІ-проєктів, оцінювання тривалості проєктів, функції втрат, трансформерна модель, машинне навчання, моделювання невизначеності, доменно-зважені функції втрат.*

Statement of the problem

Project time estimation is an important part of planning and creating artificial intelligence (AI) projects. Accurate estimates allow teams to correctly allocate resources, therefore increasing development efficiency. However, real-world AI projects are very diverse, ranging from simple proof-of-concept (PoC) prototypes to complex production-ready systems. Additionally, a high number of domains and subdomains makes this task even more complex. This often leads to standard methods falling short in terms of prediction accuracy.

Machine learning models, especially transformer architectures, are a promising alternative to manual estimation. A dataset created during earlier research aims to capture many aspects that make an AI project so complex by capturing project metadata using parameters like the project's core technical domain (Main), additional domains (Additional), development stage, and a project difficulty rating. This serves as a foundation for further research aiming to achieve the best performance possible.

Analysis of the latest research and publications

Loss functions are an important part of training supervised learning models, especially for regression tasks. Traditional metrics such as mean squared error (MSE) and mean absolute error (MAE) are popular due to their analytical simplicity and smooth optimization properties [1; 2]. However, they often make assumptions like equal importance across samples, which does not reflect real-world data distributions [3; 4].

Generic loss functions have been shown to lack robustness in the context of software engineering and AI project estimation. M. Jørgensen and M. Shepperd conducted a systematic review demonstrating the need to incorporate domain structure into forecasting methods [5]. B. A. Kitchenham et al. provided guidelines for empirical software engineering research, underscoring the necessity of context-aware modeling [6]. T. Menzies et al. emphasized the evolution of software analytics over 25 years, calling for models that align better with project risk and business outcomes [7].

Cost-sensitive and imbalanced data learning has been widely explored to address performance skew. N. Thai-Nghe et al. introduced comparative studies on cost-sensitive methods for imbalanced classification that should prove useful in designing domain-based losses [8], while C. Elkan provided important theory for adjusting models [9]. H. He and E. A. Garcia reviewed imbalanced data learning methods and showed the importance of weighting strategies connected to data structure [10].

In uncertainty modeling, A. Kendall and Y. Gal showed the integration of different uncertainties into deep learning, providing tools to manage prediction risk [11]. B. Lakshminarayanan et al. worked on deep ensembles for predictive uncertainty estimation, offering simplicity and high precision [12]. F. Xia et al. applied uncertainty modeling to text classification with multi-granularity strategies [13].

The aim of the study

Loss functions are the mathematical foundation of supervised learning. They quantify the penalty associated with prediction errors and guide the model toward better performance. In regression tasks like project time estimation, the most basic functions include mean absolute error (MAE) and mean squared error (MSE), each of which imposes different behavior on the model.

While MSE heavily penalizes large deviations, MAE treats all deviations equally. The loss function must reflect these priorities in domains where certain mistakes have disproportionately higher costs, such as underestimating AI projects.

Custom loss functions allow us to integrate domain-specific structure into training. This can include weighting schemes, variance modeling, and constraints that reshape the optimization surface. Domain-informed or uncertainty-aware losses can lead to dramatic improvements over standard MSE. Therefore, loss selection should not be treated as an additional configuration step but as a key modeling decision. This study will focus on improving loss functions for a transformer model, since it showed the best baseline results in previous studies [14].

Presentation of the main research material

Standard Mean Squared Error (MSE)

The mean squared error (MSE) is one of the most used loss functions in transformer models. It is mathematically defined as:

$$L = \left(\frac{1}{N}\right) \cdot \Sigma \left(y_i - \overline{y}_i\right)^2, \tag{1}$$

y – the true value of the project duration;

$\overline{y}$ – the predicted value of the project duration.

This loss function penalizes the square of the difference between each predicted value and the true value. Since this difference is squared, all larger errors will have a much bigger effect. This means that this loss function is more sensitive to outliers, making it both a more powerful and potentially risky tool for model training.

In project time estimation, large deviations from the initial estimation often lead to underestimations or overestimations that often have a very negative impact on planning and budgeting. This happens even more often in AI project time estimation [15]. Therefore, penalizing them more heavily should theoretically align well with business goals. However, some outliers are caused by data noise, inconsistent labeling, or more difficult and unpredictable edge-case projects.

Strengths:

• Simplicity and efficiency: Easy to calculate and differentiable, making it a good potential option for gradient-based optimization.

• Theoretical grounding: Often provides smooth optimization and is proven to be a good baseline approach.

Weaknesses:

• Lack of domain awareness: All prediction errors are treated equally, not taking into account project complexity or stage.

• Sensitivity to noise: Large errors from noisy data can heavily affect learning.

While MSE remains a strong baseline, its performance is often average in project forecasting contexts that involve domain-specific patterns.

Domain-Weighted Mean Squared Error

To overcome MSE's limitations in handling diverse projects, an alternative option is a domain-weighted MSE. This variation incorporates structured project metadata into the loss function through trainable weighted coefficients.

Each weight reflects the initial importance of a sample based on four key attributes: main domain, additional domain, project stage, and difficulty score. These metadata features are each mapped to numeric weights based on a combination of domain knowledge from industry experts. Also, these initial values are combined with trainable coefficients that will change as the model improves. This allows the incorporation of domain knowledge as a form of a baseline that will be improved upon during the training process.

It is mathematically expressed as:

$$L = \left(\frac{1}{N}\right) \cdot \Sigma (w_i \cdot (y_i - \overline{y}_i)^2), \tag{2}$$

https://doi.org/10.32782/mathematical-modelling/2026-9-1-13 ISSN 2618-0332 (print), 2618-0340 (online)

where:

- y – the true value of the project duration;
- $\bar{y}$ – the predicted value of the project duration;
- w_i – weight assigned to the i-th sample, representing its relative importance.

$$w_i = \alpha \cdot m_i + \beta \cdot a_i + \gamma \cdot s_i + \delta \cdot d_i, \quad (3)$$

where:

- m_i – numeric value for the main domain, for example, $CV = 1.0$, $ML = 1.3$;
- a_i – numeric value for additional domains, for example, $DL = 1.3$, $web = 0.8$;
- s_i – numeric value representing the project stage, for example, $PoC = 2.0$, $MVP = 4.0$;
- d_i – project difficulty score;
- α, β, γ, δ – scaling coefficients that determine the influence of each metadata component.

Main Domain Mapping:

- *CV* – 1.0: Computer vision projects are common, well-documented, and supported by well-developed tools, making them a solid baseline in terms of complexity.
- Generative *AI* – 1.1: These projects often involve newly designed architectures and complex data processing pipelines, slightly increasing implementation complexity.
- Augmented Reality – 1.2: AR typically requires integration with sensors, real-time tracking, and UX alignment, making it more demanding than standard CV tasks.
- Audio – 0.9: Audio tasks often involve smaller datasets and simpler pipelines compared to other domains.
- *NLP* – 1.0: Natural language processing benefits from many already pretrained models and libraries. This is another well-established field, making it another good baseline value.
- Machine Learning – 1.3: A broad and often foundational category, projects labeled as general ML may involve cutting-edge research or system-wide architecture challenges. Also, it often includes a high amount of data processing and architecture design.
- Signal Processing – 0.8: Traditional signal processing is considered lightweight in modern AI contexts, especially when it lacks deep learning components.
- 3D CV – 1.1: Involves additional geometric complexity and higher processing requirements.

Additional Domain Mapping:

- Image Processing – 1.0: Standardized techniques and strong community support make it a good baseline complexity domain.
- Mobile Development – 1.1: Implies client-side constraints, deployment issues, and UX integration, slightly increasing complexity.
- Machine Learning – 1.2: Even as an additional domain, ML is still complex and will often raise overall complexity.
- Signal Processing – 0.9: It tends to be less complex or already optimized.
- Web Development – 0.8: Mature domain with well-understood libraries and deployment processes.
- OCR – 1.0: Optical character recognition is highly structured and widely used, making it another good baseline for complexity.
- DL – 1.3: Deep learning implies high resource requirements and model tuning needs, elevating overall complexity.
- Backend Development – 1.0: Implementation focused and often well-scoped, therefore another good baseline.
- GIS – 1.0: Geographic information system work can be varied, but usually, it is thought of as an average difficulty.
- Data Visualization – 1.0: A support task that depends on existing data pipelines; scored neutrally.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-13 ISSN 2618-0332 (print), 2618-0340 (online)

• Data Processing – 1.0: Another standard auxiliary task that supports primary modeling efforts.

• 3D CV – 1.1: When it is an additional aspect, it increases complexity due to spatial reasoning and specialized rendering demands.

• "–" – 1.0: Default placeholder where no secondary domain is given. It should be treated as an average value.

Stage Mapping

• POC – 2.0: Proof of concept work is usually exploratory with uncertain requirements. This is the smallest project type out of all the options considered in this research.

• POC transitioning to MVP – 3.0: This stage is bigger than the POC stage, since it has to include additional tasks that would bring the project closer to the MVP level.

• MVP – 4.0: Represents a mature prototype suitable for real-world use. Additional requirements could include optimization, stability, and documentation, making it the most complex stage.

• Pilot – 2.5: An alternative and a bit more complex version of the POC with limited release features and some production expectations, but not full scalability.

• Research – 3.0: Tasks may involve high innovation but low structure, demanding abstract goal formulation. Since this stage is often the most cutting-edge, it can become quite difficult.

Difficulty is used as a direct multiplier on a 1–10 scale. A project marked as 8 on the difficulty scale will have its total weight scaled accordingly.

This detailed mapping ensures that all project characteristics – technical domain, software lifecycle stage, architectural scope, and subjective difficulty – are reflected in the weighted loss function. This strategy also accounts for potential errors in the initial mapping values and allows for mapping adjustments as the dataset evolves. Inclusion of this loss allowed the model to outperform the standard MSE loss.

Domain-weighted MSE shortcomings

While domain-weighted MSE should be capable of capturing domain knowledge and refining it during the training process, the formula used in this loss has some flaws. During training, the model adjusts its values to achieve minimum loss. Usually, this minimum value is achieved by making the most accurate prediction. In this case, if the model works perfectly, the loss would reach zero. However, in the case of this custom domain-weighted MSE, the model has complete control over the custom multiplier values since it controls the adjustment weights for the mapped values. The training process revealed a big flaw of this approach – the model eventually realizes that it can just make all the values of the adjustment weight equal zero, therefore making one of the multipliers that this custom loss consists of equal zero. Once this moment is reached, the model's prediction stops having any impact on the loss function, therefore making it useless.

This means that the next loss function should incorporate a method that would either not use the domain-weighted parameter or remove the option to just make all values zero by changing the reward function.

Uncertainty-Aware Loss

To first experiment with a mathematical way to address the previous issue, the uncertainty-aware loss should have its uncertainty value used both as a denominator and inside a logarithmic function – this doesn't allow the model to simply have an uncertainty equal zero and make the entire loss zero.

$$L = \left(\frac{1}{N}\right) \cdot \Sigma \left(\frac{(y - \overline{y})^2}{\sigma^2} + \log(\sigma^2) \right), \tag{4}$$

where:

• y – the true value of the project duration;

• $\overline{y}$ – the predicted value of the project duration;

• σ^2 – the predicted variance represents model uncertainty for that specific prediction.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-13 ISSN 2618-0332 (print), 2618-0340 (online)

It is important to note that in this case, the model predicts both the final value and the learned variance for each data point.

Strengths:

• Probabilistic modeling: Integrates uncertainty for each prediction to the final loss.

• Error calibration: Reduces overfitting to noisy data.

• Adaptivity: Improves generalization.

Weaknesses:

• Unclear reasoning: Uncertainty can be difficult to interpret, making results more difficult to understand.

• Higher computational costs: Dual-output model increases complexity, therefore making training slower.

Uncertainty-Aware loss improves resistance to noisy data. During training, it performed comparably to the domain-weighted MSE loss but didn't face the same issues in the later stages of training. This proves that this loss structure is suitable for the current model and dataset. The next step should be to replace the uncertainty coefficient with the custom mapping-based multiplier. Also, during the validation, the model showed that it sometimes predicts negative values. Since the predictions are directly correlated to hours and there can't be a negative time estimate for the project, the model's prediction range should be capped at only positive values.

Stabilized Loss with Domain Integration

The final loss function combines the structural benefits of the uncertainty-aware loss with the benefits of the mapping based on domain knowledge:

$$L = \left(\frac{1}{N}\right) \cdot \Sigma \left(\frac{(y - \overline{y})^2}{w_i^2} + \log(w_i^2) \right), \tag{5}$$

where:

• y – the true value of the project duration;

• $\overline{y}$ – the predicted value of the project duration;

• w_i – weight assigned to the i-th sample, representing its relative importance.

Strengths:

• Unified architecture: Combines weighted error and learned uncertainty.

• Domain-aware prioritization: Higher weights for more complex or risky projects focus the model on minimizing critical errors.

Weaknesses:

• Optimization sensitivity: Poorly scaled weights or learning rates can reduce training efficiency.

• Manual weight mapping: Depends on predefined domain knowledge.

Conclusions

The performance of each loss function was evaluated using standard metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics were calculated over the test set after training the model with each loss. The following table summarizes the results:

Table 1

Loss functions efficiency results

Loss Function	MAE	MSE	RMSE
Standard MSE	62.4	6543.2	80.9
Domain-Weighted MSE	46.2	4218.5	64.9
Uncertainty-Aware Loss	35.1	2987.6	54.6
Stabilized Loss with Domain Integration	32.8	2741.0	52.3

https://doi.org/10.32782/mathematical-modelling/2026-9-1-13 ISSN 2618-0332 (print), 2618-0340 (online)

The baseline MSE loss shows the highest error values, demonstrating that a standard approach is not a good fit for the current problem. Introducing domain-aware weighting provided a significant performance improvement, since domain knowledge allowed the model to have a better starting point. Uncertainty-Aware Loss, which didn't have the same issues with making trainable parameters zero, as in the previous loss, showed yet another significant improvement, proving that the structure of this loss can be used successfully. Finally, Stabilized Loss with Domain Integration, which combines domain weighting, improved structure, and task-based model limitations, achieved the best performance across all metrics.

Further research can use this study on loss functions for transformers in the AI project duration prediction domain by combining the final stabilized loss with domain integration with additional model improvement techniques. These could include methods like attention-based pooling to further focus on the difference in importance of different metadata values. A potential downside of this method is overfitting. A promising way to prevent overfitting could be activation dropout, which would randomly zero activations, therefore preventing overfitting. These methods could further improve the model's performance and allow it to become an assistant-type tool for people working on project estimations.

Bibliography

1. Bishop C. M. Pattern recognition and machine learning. Springer, 2006. URL: https://link.springer.com/book/10.1007/978-0-387-45528-0 (дата звернення: 08.03.2026).
2. Géron A. Hands-on machine learning with scikit-learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. 2nd ed. O'Reilly Media, 2019. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (дата звернення: 08.03.2026).
3. Gneiting T., Raftery A. E. Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association. 2007. Vol. 102, № 477. P. 359–378. DOI: https://doi.org/10.1198/016214506000001437
4. Khoshgoftaar T. M., Seliya N. Comparative assessment of software quality classification techniques: An empirical case study. Empirical Software Engineering. 2004. Vol. 9, № 3. P. 229–257. DOI: https://doi.org/10.1023/B:EMSE.0000027781.18360.9B
5. Jørgensen M., Shepperd M. A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering. 2007. Vol. 33, № 1. P. 33–53. DOI: https://doi.org/10.1109/TSE.2007.3
6. Kitchenham B. A., Pfleeger S. L., Pickard L. M., Jones P. W., Hoaglin D. C., El Emam K., Rosenberg J. Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering. 2002. Vol. 28, № 8. P. 721–734. DOI: https://doi.org/10.1109/TSE.2002.1027796
7. Menzies T., Zimmermann T. Software analytics: So what? IEEE Software. 2013. Vol. 30, № 4. P. 31–37. DOI: https://doi.org/10.1109/MS.2013.58 (дата звернення: 08.03.2026).
8. Thai-Nghe N., Gantner Z., Schmidt-Thieme L. Cost-sensitive learning methods for imbalanced data. Proceedings of the International Joint Conference on Neural Networks. 2010. P. 1–8. DOI: https://doi.org/10.1109/IJCNN.2010.5596486
9. Elkan C. The foundations of cost-sensitive learning. Proceedings of the 17th International Joint Conference on Artificial Intelligence. 2001. P. 973–978. URL: https://cseweb.ucsd.edu/~elkan/rescale.pdf (дата звернення: 28.02.2026).
10. He H., Garcia E. A. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 2009. Vol. 21, № 9. P. 1263–1284. DOI: https://doi.org/10.1109/TKDE.2008.239
11. Kendall A., Gal Y. What uncertainties do we need in Bayesian deep learning for computer vision? Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5574–5584. DOI: https://doi.org/10.48550/arXiv.1703.04977

12. Lakshminarayanan B., Pritzel A., Blundell C. Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 6402–6413. DOI: https://doi.org/10.48550/arXiv.1612.01474
13. Xia F., Liu Y., Wang Y., Liu Y. Multi-granularity uncertainty modeling for text classification. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019. P. 1056–1065. DOI: https://doi.org/10.18653/v1/P19-1101
14. Kachmar P. Comparative analysis of model performance for time estimations in AI projects. SCIENTIA: Collection of Scientific Papers. 2025. P. 207–212. URL: https://previous.scientia.report/index.php/archive/article/view/2300 (дата звернення: 18.03.2026).
15. Kachmar P. Statistical analysis of time estimation patterns in AI project timelines. SCIENTIA: Collection of Scientific Papers. 2024. P. 147–150. URL: https://previous.scientia.report/index.php/archive/article/view/2230 (дата звернення: 18.03.2026).

References

1. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer. https://link.springer.com/book/10.1007/978-0-387-45528-0 [in English].
2. Géron, A. (2019). Hands-on machine learning with scikit-learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ [in English].
3. Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477), 359–378. DOI: https://doi.org/10.1198/016214506000001437 [in English].
4. Khoshgoftaar, T. M., & Seliya, N. (2004). Comparative assessment of software quality classification techniques: An empirical case study. Empirical Software Engineering, 9(3), 229–257. DOI: https://doi.org/10.1023/B:EMSE.0000027781.18360.9B [in English].
5. Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering, 33(1), 33–53. DOI: https://doi.org/10.1109/TSE.2007.3 [in English].
6. Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering, 28(8), 721–734. DOI: https://doi.org/10.4224/8914084 [in English].
7. Menzies, T., & Zimmermann, T. (2013). Software analytics: So what? IEEE Software, 30(4), 31–37. DOI: https://doi.ieeecomputersociety.org/10.1109/MS.2013.86 [in English].
8. Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. Proceedings of the International Joint Conference on Neural Networks, 1–8. DOI: 10.1109/IJCNN.2010.5596486 [in English].
9. Elkan, C. (2001). The foundations of cost-sensitive learning. Proceedings of the 17th International Joint Conference on Artificial Intelligence, 973–978. Retrieved from https://www.researchgate.net/publication/2365611_The_Foundations_of_Cost-Sensitive_Learning [in English].
10. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. DOI: 10.35940/ijrte.E6286.018520 [in English].
11. Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? Advances in Neural Information Processing Systems, 30, 5574–5584. DOI: https://doi.org/10.48550/arXiv.1703.04977 [in English].
12. Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in Neural Information Processing Systems, 30, 6402–6413. DOI: https://doi.org/10.48550/arXiv.1612.01474 [in English].

13. Xia, F., Liu, Y., Wang, Y., & Liu, Y. (2019). Multi-granularity uncertainty modeling for text classification. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 1056–1065. DOI: https://doi.org/10.18653/v1/P19-1101 [in English].
14. Kachmar, P. (2025). Comparative analysis of model performance for time estimations in AI projects. SCIENTIA: Collection of Scientific Papers, 207–212. Retrieved from https://previous.scientia.report/index.php/archive/article/view/2300 [in English].
15. Kachmar, P. (2024). Statistical analysis of time estimation patterns in AI project timelines. SCIENTIA: Collection of Scientific Papers, 147–150. Retrieved from https://previous.scientia.report/index.php/archive/article/view/2230 [in English].

Kachmar Pavlo Olehovych – Postgraduate Student at the Department of Applied Mathematics of the Ivan Franko National University of Lviv. E-mail: pavlokach@gmail.com, ORCID: 0009-0001-3752-5332.

Качмар Павло Олегович – аспірант кафедри прикладної математики Львівського національного університету імені Івана Франка. E-mail: pavlokach@gmail.com, ORCID: 0009-0001-3752-5332.

Дата першого надходження статті до видання: 03.04.2026
Дата прийняття статті до друку після рецензування: 07.05.2026
Дата публікації (оприлюднення) статті: 01.07.2026