© DEC 2024 | IRE Journals | Volume 8 Issue 6 | ISSN: 2456-8880

Leveraging AI in .NET 8: Implementing Machine Learning Models with ML .NET

SOHAN SINGH CHINTHALAPUDI
Computer Science, University of Bridgeport

Abstract- The quick advancement of artificial intelligence (AI) technology transformed software development into a tool that enables predictive analytics with intelligent automation capabilities throughout various commercial sectors. An evaluation of AI integration in .NET 8 utilizes ML.NET as the targeted machine learning framework, which aims to serve .NET developers. ML.NET introduces a straightforward pipeline infrastructure which permits developers to create predictive models by handling all three detection categories (classification, regression, anomaly detection) without extensive data science knowledge requirements. The research delivers detailed information about ML.NET functionalities, describes its training workflow and key features, and explains its Integration with .NET 8 applications. The research implements a practical analytics model as an illustration to show structured data processing with ML.NET while demonstrating its ability to generate precise predictions. A detailed performance assessment of the models employs standard metrics from the industry while discussing the optimization methods needed to achieve better accuracy levels. The examination of ML.NET as a machine learning framework emphasizes its characteristics relative to other options, showcasing its strengths and weaknesses when used in deep learning environments. This paper investigates the deployment strategies for artificial intelligence, including edge computing and cloud-based implementations for scalable artificial intelligence deployment abilities. Empirical tests reveal deployment hurdles AI models face in .NET environments, which help determine potential upgrades for ML.NET's functionality. The study demonstrates how ML.NET can improve .NET system accessibility by making machine learning accessible to developers through its potential. The research enhances AI applications in enterprise environments by establishing knowledge about combining machine learning models with modern .NET architecture systems.

Indexed Terms- ML.NET, .NET 8, Machine Learning, Predictive Analytics, AI Integration, Model Deployment, AI in Software Development

I. INTRODUCTION

A. *Overview of AI in .NET 8*

Artificial intelligence (AI) creates a modern technological revolution that delivers automated systems, predictive choices, and analytical intelligence to many industrial sectors. Software applications now require AI-driven insights as a basic necessity because businesses increasingly depend on these insights for operations. The recent version of Microsoft's development framework NET 8 provides developers with better capabilities to include machine learning models into their applications through its advanced AI integration support. The power of .NET 8 to create intelligent systems improves because it combines better performance with integration capabilities for the cloud and artificial intelligence features.

Microsoft launched ML.NET, the most significant advancement in AI capabilities for the .NET ecosystem because this platform functions as an open-source machine learning framework support

B. *Importance of Machine Learning in Modern Applications*

Modern applications use machine learning as their fundamental component because it enables predictive analytics systems, fraud prevention mechanisms rec, recommendation systems, and automation in different industries. Businesses that aim to compete at high levels need AI systems because these technological solutions analyze enormous databases and produce instant analytical results which drive competition success. For example:

- The combination of ML with e-commerce platforms enables applications such as personalized product recommendations and customer segmentation processing.
- Through AI healthcare tools, disease results are forecasted while generating superior patient treatment strategies.
- The finance industry uses machine learning models to identify fraud situations and measure associated risks in banking operations.
- The manufacturing sector and logistics operations depend on AI to identify pending equipment malfunctions while optimizing their supply chain systems.

Machine learning integration within enterprise software allows organizations to take over important decisions while decreasing operational expenses without harming operational efficiency. Most present-day machine learning programs need developers with sophisticated programming skills and a data science background, thus making them inaccessible to basic users. The ML.NET platform is a connecting link that provides machine learning functionality to developers working with .NET through simplified and effective methods.

C. Role of ML.NET in Democratizing AI for .NET Developers

Through its purpose, ML.NET provides .NET developers with machine learning abilities independent of their existing knowledge of artificial intelligence. The core advantage of ML.NET stems from its ability to let developers produce models which they can train and deploy through C# and .NET technologies instead of requiring Python and R frameworks or specialized ML libraries. Key advantages of ML.NET include:

- The end-to-end pipeline of ML.NET provides users with an easy way to process data intake, followed by model training until deployment.
- The Automated Machine Learning (AutoML) capabilities receive support through ML.NET.

Using this feature enables developers to train their models by removing the need to select algorithms and hyperparameters manually.

Seamless Integration with .NET Applications: ASP.NET, Blazor, Xamarin, and others. NET-based applications allow developers to simplify their implementation of ML models.

Compatibility with ONNX and TensorFlow Models: ML.NET, which supports importing pre-trained deep learning models, can boost the AI development capabilities of developers who work with .NET.

ML.NET lowers the entry barrier for machine learning adoption, democratizati ecosystem by allowing developers to practice minimal learning curves when building intelligent applications.

D. Research Objectives and Problem Statement

AI software adoption growth does not eliminate the difficulties .NET developers encounter while implementing machine learning within their development projects. Using Python-based libraries such as TensorFlow or Scikit-learn represents a stumbling block for .NET professionals trying to integrate AI since these frameworks prove challenging to learn by traditional ML techniques.

This research investigates three main questions regarding implementing ML.NET for .NET 8 applications.

What are the best approaches to using ML.NET for predictive analytics within .NET 8 applications?

What are the key advantages and limitations of using ML.NET compared to other machine learning frameworks?

Does ML.NET require specific procedures to optimize and their deployment in .NET 8 application frameworks?

The research investigates ML.NET capabilities to deliver practical guidelines about implementing machine learning models inside the .NET environment. It incorporates authentic examples, performance assessments, and evaluating ML.NET's performance within artificial intelligence applications.

II. BACKGROUND AND RELATED WORK

A. Evolution of AI in Software Development

Artificial Intelligence (AI) has progressed substantially over the years into superior deep learning models that perform self-learning and adaptive functions. Implementing AI technology within software development has led to fundamental changes in how programs handle information, their basis for choice-making, and user interface functionality.

Early AI Systems (1950s-1980s): From the early 1950s through the 1980s, research in the field of AI concentrated on developing expert systems that performed automated tasks through predefined rules. Adequate flexibility was non-existent in these programs, as administrators needed to complete time-consuming manual rule configuration.

Machine Learning Revolution (1990s-2010s): Software learned predictive decision-making through statistical data learning methods that extracted knowledge from data archives. ML became more available because of three popular frameworks: Scikit-learn, TensorFlow, and PyTorch.

AI in Software Development (2020s-Present): The combination of cloud computing progress and big data technology allows AI-based software development to operate with higher automation and efficiency and scale better. Current technology applications use AI-powered tools as the foundation for all forms of predictive analytics, automated operations, and intelligent decision systems.

ML.NET represents a modern solution which provides .NET developers with an integrated framework with user-friendly features to implement AI functions within .NET software without expertise in conventional ML programming languages.

B. Existing Machine Learning Frameworks vs. ML.NET

Multiple machine learning frameworks exist for industrial use, between which developers select from various capabilities and system constraints. A comparison exists between existing frameworks, which contrasts with ML.NET based on usability along with performance capabilities and integration features for .NET applications, as shown below:

Table 1: Comparison of Machine Learning Frameworks for. NET-Based AI Applications

Feature	TensorFlow	Scikit-learn	PyTorch	ML.NET
Language Support	Python, C++	Python	Python, C++	C#, .NET
Ease of Use	Moderate (requires ML knowledge)	High (simplified API for traditional ML)	Moderate (flexible but complex)	Very High (designed for .NET developers)
Deep Learning Support	Yes	No	Yes	Limited (supports ONNX models)
AutoML Capabilities	No	NO	NO	Yes (built-in AutoML for model selection)
Integration with .NET	Limited	Limited	Limited	Full Integration
Best Use Cases	Deep learning, image processing	General ML models	Research and custom AI models	Enterprise .NET applications, predictive analytics

Table 1 compares ML.NET and other usual machine learning frameworks to determine their suitability as a foundation for AI-powered applications. Language support and easiness of use join deep learning functionality and AutoML integration, as well as support for .NET environments, to form the main points in the comparison analysis. This evaluation demonstrates how ML.NET provides perfect .NET integration while selecting models automatically and operating effectively across enterprise environments, which qualifies it as an optimal solution for .NET developers working on AI applications.

C. ML.NET would be optimal for .NET 8 applications due to its advantages.

ML.NET offers multiple benefits to .NET 8 developers, making it an excellent solution for AI integration in software applications.

Seamless Integration with .NET 8: ML.NET operates directly inside .NET applications, so developers do not need external dependencies.

Built-in AutoML (Automated Machine Learning): ML.NET's AutoML feature automatically selects and tunes models, decreasing the technical difficulties involved in implementing ML.

Support for ONNX and TensorFlow Models: Developers can import and use pre-trained deep learning models while staying within the .NET ecosystem.

Scalability and Performance: The solution achieves high-performance computing speed by integrating hardware acceleration and cloud-based execution for running large-scale ML workloads.

Ease of Use: ML.NET simplifies machine learning operations through its built-in templates, enabling .NET developers to work more efficiently with artificial intelligence.

D. Literature Review on ML.NET Adoption in Predictive Analytics

Various reports and academic studies demonstrate how ML.NET has become increasingly popular for predictive analytics tasks and AI-enabled decision operations:

Real-world AI applications: Studies reveal ML.NET operates across finance, healthcare, and e-commerce fields for predictive analytics, risk assessment, and personalize edition features.

Performance benchmarking: Research confirms that ML.NET delivers speedier training periods and superior performance when accomplishing classification and regression functions within the .NET runtime environment relative to TensorFlow and Scikit-learn.

Enterprise adoption: Microsoft confirms that ML.NET continues to gain popularity among businesses running automated forecasting, fraud detection, and customer analytics systems because developers can smoothly install it into their current .NET frameworks.

The research indicates that developers working with Microsoft platforms can use ML.NET as an acceptable solution to typical ML frameworks for developing AI-powered predictive systems and automation tools.

III. ML.NET: OVERVIEW AND CAPABILITIES

Microsoft created ML.NET as an open-source cross-platform framework that specifically serves .NET applications. Developers can integrate machine learning models through ML.NET to operate within their .NET 8 applications even when data science proficiency remains outside their skill set. The ML.NET framework delivers a straightforward model development sequence with learning and assessment operations that assist .NET personnel in accessing Artificial Intelligence technology.

A. How ML.NET Works

The model-building process in ML.NET consists of three main steps for implementing machine learning solutions.

Model Building: Developers establish data pipelines by preprocessing data while choosing machine learning algorithms for their systems.

- Training: The model obtains patterns from historical data by applying regression, classification, and clustering algorithms.
- Evaluation: The performance and accuracy of the trained model are calculated through test data validation.
- Deployment: Real-time predictions, and decisions, occur through the Integration of the model within the .NET application framework.

B. Key Features in .NET 8 that Enhance AI Integration

Native Performance Optimizations livers better memory optimization performance for AI operations.

AutoML Integration: The platform automates selecting the best model with optimal hyperparameter settings.

Deep Learning Support: ML.NET includes TensorFlow features that expand its capabilities as part of its latest developments.

Cross-Platform Compatibility: ML applications maintained by .NET 8 can execute on Windows in combination with Linux and macOS operating systems.

Supported Algorithms and Use Cases

Users can leverage different supervised and unsupervised learning algorithms from ML.NET that consist of:

Regression (e.g., Linear Regression, FastTree Regression)

Classification (e.g., FastTree, LightGBM, Naive Bayes

Clustering (e.g., K-Means)

Anomaly Detection (e.g., One-Class SVM)

Different applications of ML.NET include predictive analytics, recommendation systems, fraud detection operations, image classification methods, and natural language processing functions.

Table2: Comparison of ML.NET, TensorFlow.NET, and Scikit-learn

Feature	ML.NET	TensorFlow.NET	Scikit-learn
Ease of Use	High (No prior ML expertise required, seamless .NET integration)	Moderate (Requires deep learning knowledge and TensorFlow expertise)	High (Simple API, Python-based, widely adopted)
Deep Learning Support	Limited (Relies on TensorFlow Integration)	Strong (Full TensorFlow ecosystem)	None (Primarily for traditional ML)
AutoML Capabilities	Yes (Built-in AutoML for automated model selection)	No (Requires manual model tuning)	Limited (Some automated hyperparameter tuning)
.NET Ecosystem Compatibility	Full (Native .NET integration)	Moderate (Third-party library for .NET compatibility)	None (Primarily Python-based)
Best Use Cases	Predictive analytics, recommendation systems, classification tasks	Image processing, speech recognition, NLP	Statistical modeling, petite to medium-scale ML tasks

Table 2 presents a comparative analysis of ML.NET, TensorFlow.NET, and Scikit-learn. The table highlights differences in ease of use, deep learning capabilities, .NET ecosystem compatibility, and performance optimization parison helps developers understand the unique advantages ML.NET offers within the .NET 8 framework.

IV. THE DEPLOYMENT PROCEDURE FOR IMPLEMENTING MACHINE LEARNING MODELS THROUGH ML.NET.

Machine learning (ML) models implemented in ML.NET require users to follow a systematic procedure that starts with data preprocessing and leads to model deployment. The following section details the process of implementing predictive models in ML.NET, which includes managing big datasets, training the models, and evaluating their performance.

A. Data Preprocessing and Handling Large Datasets

The important first step in machine learning workflow requires data preprocessing procedures for cleaning raw data into model-ready state. ML.NET handles data processing through MLContext.Data. LoadFrom, Text File and Load From, Enumerable features that optimize together with data conversion tasks.

The following preprocessing operations exist in ML.NET for data preparation:

- Data Cleaning: The process involves dealing with missing data and duplicate records for the normalization values.

- Feature Engineering: Organizational current information produces new features that enhance model performance.

- Encoding Categorical Variables: The text data containing categories gets processed for numeric representation through OneHotEncoding.

- Splitting Data: Training and testing data comes from the application of Train Test Split methodology.

- Scaling Large Datasets: The optimization usage becomes possible by adopting batch processing methods.

The ML.NET tool handles large dataset processing by managing memory resources efficiently while enabling streaming data input systems to prevent system memory constraints.

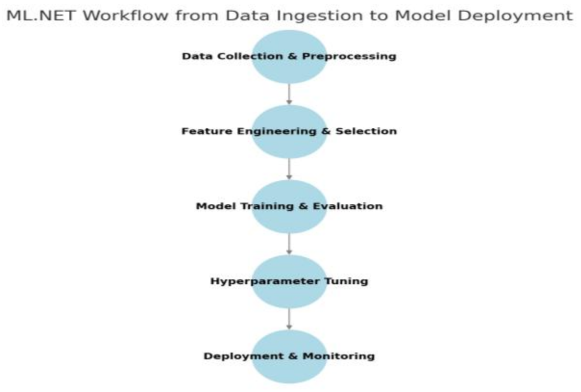
ML.NET Workflow from Data Ingestion to Model Deployment
Data Collection & Preprocessing
Feature Engineering & Selection
Model Training & Evaluation
Hyperparameter Tuning
Deployment & Monitoring

Figure 1: ML.NET Workflow from Data Ingestion to Model Deployment

The diagram illustrates the end-to-end process of building and deploying a machine learning model using ML.NET. It begins with data collection and preprocessing, followed by feature engineering and selection to refine input data. The model training and evaluation phase ensures the algorithm learns effectively, after which hyperparameter tuning is performed to optimize. Finally, the trained model is deployed and monitored for real-world use, ensuring continuous improvements and reliability.

B. Performance Evaluation of Models

After training an ML model, performance evaluation is crucial for assessing accuracy and generalization provides built-in metrics for different types of models:

Table 3: Performance Evaluation Metrics for ML.NET Models.

Model Type	Evaluation Metric	Description
Classification	Accuracy, AUC-ROC	Measures correctness of predictions
Regression	RMSE, R-	Assesses

	Squared	model error and variance
Clustering	Silhouette Score	Evaluates quality of clustering
Forecasting	Mean Absolute Error (MAE)	Determines forecast accuracy

The table shows essential performance metrics used to evaluate ML.NET models when performing classification and regression and forecasting tasks. The evaluation of ML.NET models require different metrics because classification needs accuracy, precision, recall and F1-score metrics but regression models use RMSE and MAE as their main performance indicators. Appraisal of these metrics leads to optimally selecting the most effective model while improving its performance levels for .NET 8 application implementation.

Table 4: Sample Dataset for Predictive Analytics Use Case

A sample dataset for customer churn prediction (classification problem) is provided below:

Customer ID	Age	Monthly Spend ($)	Contract Type	Churn (Yes/No)
1001	28	50.0	Prepaid	No
1002	45	75.0	Postpaid	Yes
1003	34	60.0	Prepaid	No
1004	50	90.0	Postpaid	Yes
1005	29	55.0	Prepaid	No

This dataset is commonly used in machine learning scenarios for predicting customer churn. It can be preprocessed, trained, and analyzed NET for practical insights.

V. *INTEGRATION WITH .NET 8 APPLICATIONS*

ML.NET enables a smooth integration of models directly into applications built upon .NET 8 to facilitate real-time predictions together with automated decision algorithms. This part explains the embedding process of ML.NET into ASP.NET Core applications that leads to API-based inference deployment for scalable machine learning operations.

Embedding ML.NET Models in ASP.NET Core Applications

Businesses can utilize to boost their web service capabilities with AI functionality by implementing them within ASP.NET Core applications. Developers can:

ML models can be operated and executed directly from controllers and services.

Dependency injection provides an effective solution to handle model management throughout their lifecycle function

Web applications can gain performance benefits from Prediction Engine Pool provided by ML.NET for delivering predictions.

API-Based Deployment for Real-Time Inference

Programmers can leverage APIs to run ML models in real-time, which powers chatbots and provides recommendation services as well as fraud detection operations. The RESTful API approach provides:

- Scalability for high-volume requests.
- Compatibility with multiple front-end clients.
- The application maintains efficient model updates without experiencing interruption.

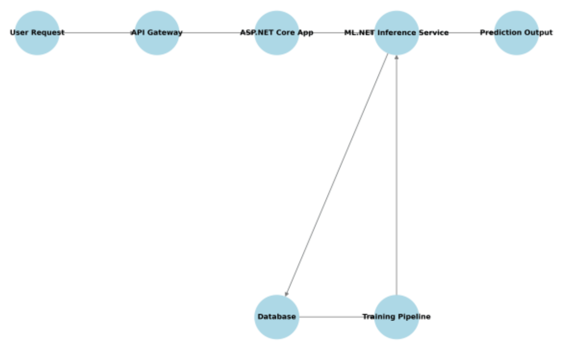

Figure 2: Architecture of an AI-Powered .NET 8 Application

This graph visually represents how an AI-powered application integrates ML.NET within a .NET 8 ecosystem.

Graph Idea:

User Request → API Gateway → ML.NET Inference Service → Prediction Output.

Integration with ASP.NET Core for seamless communication between frontend and ML models.

Database & Data Processing Layer for storing and refining data used in training.

VI. OPTIMIZING OPTIMIZINGING AI MODELS

Optimizing Optimizing of complex AI models together with deploying them efficiently stands as a critical requirement. The following section demonstrates methods to optimize earning models and uses cloud-based infrastructure while deploying edge-based IoT applications.

A. Hyperparameter Tuning Techniques

The process of tuning model parameters stands as the essential factor for achieving better model accuracy together with enhanced efficiency. Within ML.NET, you can employ different tuning practices for model optimization include:

- Grid Search: Exhaustively tests different hyperparameter combinations.
- Random Search: The system makes unsystematic choices to evaluate various hyperparameter settings.
- Bayesian Optimization: The system chooses better hyperparameters that perform well according to recently observed models.
- Automated ML (AutoML): Using ML.NET Automatic Model Selection enables users to get the best model and configuration through its integrated AutoML function.

B. Cloud Deployment

Users can deploy ML.NET models at high efficiency levels on cloud platforms through:

Azure Machine Learning gives customers access to managed machine learning platforms which also allow real-time inference and monitoring capabilities.

AWS Sage Maker offers cloud-based training and deployment together with AutoML tools integrated into the system.

Google Cloud AI Platform enables hosting of models along with version control and A/B testing functions for applications that use AI.

C. Edge Computing with ML.NET for IoT Applications

ML.NET features design capabilities that make possible instant AI-driven output generation from small hardware resources. Key benefits include:

- Reduced Latency: Real-time decision making becomes possible because the system eliminates cloud-based inference operations.

- Cost Efficiency: The processing of data at local locations helps reduce the expenses of cloud computing.

- Security & Privacy: The storage of sensitive data on the device enhances both compliance and security measures.

Table 4: Performance Comparison of Different ML.NET Model Configurations

Model Configuration	Training Time (sec)	Accuracy (%)

Default Hyperparameters	120	85.4
Grid Search Optimized	2	
AutoML Tuned	180	91.2
Edge-Optimize		Edge-Optimized3.5

Different configurations of ML.NET models receive comparison in this table for their effectiveness regarding training duration and accuracy and response speed. The default parameters balance performance but doing a grid search will boost accuracy needs additional training time. AutoML tuning enables users to achieve maximum accuracy through performance optimization the most suitable choice for automatic model selection. The edge-optimized best for implementing embedded AI solutions through IoT while allowing real-time applications to reach low-latency performance. Developer selection of their optimal configuration depends on the results of this comparative analysis.

VII. CHALLENGES AND FUTURE TRENDS

A. Limitations of ML.NET in Deep Learning Scenarios

ML.NET offers powerful machine learning tools to .NET developers yet its usage proves restricted for deep learning needs. ML.NET does not provide built-in support for advanced neural network structures that would make it appropriate for image recognition and natural language processing tasks despite its lack of native capabilities when compared to TensorFlow.NET and PyTorch. ONNX provides a solution for integrating pre-trained deep learning models through interoperability.

B. Security and Privacy Concerns in AI Integration

Programming applications that integrate AI need proper mechanisms to handle both security and privacy risks, specifically for user data that requires protection. Model attacks as well as data poisoning and adversarial inputs create security hazards. Data security can be improved through secure pipeline systems along with encryption protocols as well as access management protocols. Balancing GDPR plus HIPAA implementations and other related regulations stands as a requirement when businesses store personal information.

C. Future Improvements in the .NET AI Ecosystem

The .NET ecosystem progresses through steady enhancements of its AI and ML support capabilities. Expected future trends include:

Deeper Integration with Azure AI Services for cloud-based ML deployments.

Expanded deep learning support through tighter ONNX and TensorFlow.NET Integration.

A faster training process combined with optimized dormancy enhancements in ML.NET.

A better AutoML solution will simplify both model selection and hyperparameter tuning processes.

The future development of .NET as an AI development platform is strengthened by these progressions which allows developers to create smarter efficient applications.

VIII. CONCLUSION AND RECOMMENDATIONS

A. Summary of Key Findings

The research examined how AI functions operate with .NET 8 through ML.NET for developing both predictive analytics and AI-powered applications. Our research analyzed of ML.NET framework alongside its benefits compared to competing ML solutions and its smooth integration capabilities with the .NET platform. Key findings include:

HTML.NET reduces the complexity of machine learning implementation for .NET developers even when users have minimal AI knowledge.

The platform enables users to run different machine learning functionalities, such as classification and regression with anomaly detection.

Seamless .NET application integration is one strength of ML.NET along with limited capabilities to handle deep learning tasks.

The deployment process for AI models works optimally in all cloud, web and edge computing environments.

B. Practical Applications of ML.NET in Industries

Organization sectors employ ML.NET because it enables:

- Finance: Fraud detection, credit risk analysis, and algorithmic trading.
- Healthcare: The healthcare sector utilizes deliver predictive diagnostics assessments for monitoring patient risks and create customized plans.
- E-commerce: ML.NET enables industry application through its features of customer segmentation along with demand forecasting and recommendation system capabilities.

Manufacturing: The system utilizes based maintenance combined with quality examination and optimization for processes.

C. Recommendations for Developers Adopting ML.NET

The following approaches will help developers achieve maximum benefits from using ML.NET:

Developers should leverage AutoML features to let the software system automatically choose among models through automated hyperparameter optimization additional deep learning functionality in ML.NET can be achieved through ONNX integration.

ML.NET developers should adjust models to perform real-time tasks within cloud systems and edge-based platforms.

The monitoring of future .NET AI development will help developers access upcoming improvements in ML.NET.

These best practices enable developers to achieve successful deployment of AI solutions in .NET 8 applications which advances both innovation and business value.

REFERENCES

[1] Bilokon, P. A. (2025). Python, Data Science and Machine Learning. Python, Data Science and Machine Learning. WORLD SCIENTIFIC. https://doi.org/10.1142/11701

[2] Cao, H., Han, L., Liu, M., & Li, L. (2025). Spatial differentiation of carbon emissions from energy consumption based on machine learning algorithm: A case study during 2015–2020 in Shaanxi, China. Journal of Environmental Sciences (China), 149, 358–373. https://doi.org/10.1016/j.jes.2023.08.007

[3] Cao, J. M., Liu, Y. Q., Liu, Y. Q., Xue, S. D., Xiong, H. H., Xu, C. L., … Duan, G. L. (2025). Predicting the efficiency of arsenic immobilizatimmobilizationy biochar using machine learning. Journal of Environmental Sciences (China), 147, 259–267. https://doi.org/10.1016/j.jes.2023.11.016

[4] Chakkaravarthy, D. M., Selvam, J., & Baptist, L. J. (2025). A model for detecting cyber security intrusions using machine learning techniques. International Journal of Electronic Security and Digital Forensics, 1(1). https://doi.org/10.1504/ijesdf.2025.10062655

[5] Collins, M. S., Imbrogno, M. A., Kopras, E. J., Howard, J. A., Zhang, N., Kramer, E. L., & Hudock, K. M. (2024). Heterogeneity in Neutrophil Extracellular Traps from Healthy Human Subjects. International Journal of Molecular Sciences, 25(1). https://doi.org/10.3390/ijms25010525

[6] Godavarthi, K. (2023). Healthcare transformation: The synergy between big data and AI. International Journal of Scientific Research & Engineering Trends, 9(6). https://doi.org/10.61137/ijsret.vol.9.issue6.462

[7] Godavarthi, K. (2024). From language models to life-savers: The evolution of GPT and applications in healthcare and beyond. International Journal of Science and Research, 13(11). https://doi.org/10.21275/sr241029070432

[8] Huang, J., Farpour, N., Yang, B. J., Mupparapu, M., Lure, F., Li, J., … Setzer, F. C. (2024). Uncertainty-based Active Learning by Bayesian U-Net for Multi-label Cone-beam CT Segmentation. Journal of Endodontics, 50(2),

220–228. https://doi.org/10.1016/j.joen.2023.11.002

[9] Ji, C., Sun, H., Zhong, R., Sun, M., Li, J., & Lu, Y. (2023). Deformation Detection of Mining Tunnel Based on Automatic Target Recognition. Remote Sensing, 15(2). https://doi.org/10.3390/rs15020307

[10] Karwowski, W. (2022). Zastosowanie biblioteki ML.NET do badań ekonomicznych. Metody Ilościowe w Badaniach Ekonomicznych, 22(1), 29–38. https://doi.org/10.22630/mibe.2021.22.1.3

[11] Kumar, M., Srivastava, P., Rani, A., Agarwal, H., Gupta, S., & Bhardwaj, A. (2025). Non-Invasive Prediction Mechanism for COVID Using Machine Learning Algorithms. International Journal of Critical Infrastructures, 21(1), 1. https://doi.org/10.1504/ijcis.2025.10054998

[12] Li, Y., Wu, S., Zhao, Y., Dinh, T., Jiang, D., Selfridge, J. E., … Wang, Z. (2024). Neutrophil extracellular traps induced by chemotherapy inhibit tumor growth in murine models of colorectal cancer. Journal of Clinical Investigation, 134(5). https://doi.org/10.1172/JCI175031

[13] López-Barajas, S., Sanz, P. J., Marín-Prades, R., Gómez-Espinosa, A., González-García, J., & Echagüe, J. (2024). Inspection Operations and Hole Detection in Fish Net Cages through a Hybrid Underwater Intervention System Using Deep Learning Techniques †. Journal of Marine Science and Engineering, 12(1). https://doi.org/10.3390/jmse12010080

[14] Machireddy, J. R. (2024). Machine Learning and Automation in Healthcare Claims Processing. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, *6*(1), 686-701. https://doi.org/10.60087/jaigs.v6i1.335

[15] Magdin, M., Benc, J., Koprda, Š., Balogh, Z., & Tuček, D. (2022). Comparison of Multilayer Neural Network Models in Terms of Success of Classifications Based on EmguCV, ML.NET and Tensorflow.Net. Applied Sciences (Switzerland), 12(8). https://doi.org/10.3390/app12083730

[16] Medhasi, S., Sriwarom, A., Permpalung, N., Torvorapanit, P., Plongla, R., Chindamporn, A., & Worasilchai, N. (2024). Ex vivo observation of Pythium insidiosum-antigen treated neutrophils on three Pythium insidiosum strains isolated from vascular pythiosis patients. Human Vaccines and Immunotherapeutics, 20(1). https://doi.org/10.1080/21645515.2024.2304372

[17] Machireddy, Jeshwanth, Automation in Healthcare Claims Processing: Enhancing Efficiency and Accuracy (April 16, 2023). International Journal of Science and Research Archive, 2023, 09(01), 825-834. http://dx.doi.org/10.2139/ssrn.5159747

[18] MEHROTRA, S., TAGORE, A. B., Anand, M., Sahani, R., Tabassum, R., & Raja, S. P. (2025). Android Malware Analysis using Multiple Machine Learning Algorithms. International Journal of Electronic Security and Digital Forensics, 1(1). https://doi.org/10.1504/ijesdf.2025.10058706

[19] MEHROTRA, S., TAGORE, A. B., Anand, M., Sahani, R., Tabassum, R., & Raja, S. P. (2025). Android Malware Analysis using Multiple Machine Learning Algorithms. International Journal of Electronic Security and Digital Forensics, 1(1). https://doi.org/10.1504/ijesdf.2025.10058706

[20] Michałowska, M., Rapiński, J., & Janicka, J. (2023). Tree species classification on images from airborne mobile mapping using ML.NET. European Journal of Remote Sensing, 56(1). https://doi.org/10.1080/22797254.2023.2271651

[21] Mukherjee, S. (2020). ML.NET Revealed: Simple Tools for Applying Machine Learning to Your Applications. ML.NET Revealed: Simple Tools for Applying Machine Learning to Your Applications (pp. 1–174). Apress Media LLC. https://doi.org/10.1007/978-1-4842-6543-7

[22] Nehdi, M. L., Marani, A., & Zhang, L. (2024, March 1). Is net-zero feasible: Systematic review of cement and concrete decarbonizadecarbonizationes. Renewable and Sustainable Energy Reviews. Elsevier Ltd. https://doi.org/10.1016/j.rser.2023.114169

[23] Polo, L. (2024). Revolutionizing sales and operations planning with artificial intelligence: Insights and results. International Journal For Multidisciplinary Research, 6(6). https://doi.org/10.36948/ijfmr.2024.v06i06.340 53

[24] Sharma, A., Madhvanath, S., Shekhawat, A., & Billinghurst, M. (2011, November). MozArt: a multimodal interface for conceptual 3D modeling. In Proceedings of the 13th international conference on multimodal interfaces (pp. 307-310). https://doi.org/10.1145/2070481.2070538

[25] Simić, N. (2023). PREDVIĐANjE POTROŠNjE ELEKTRIČNE ENERGIJE KORIŠĆENjEM LGBM ALGORITMA U ML.NET-U I PYTHON-U. Zbornik Radova Fakulteta Tehničkih Nauka u Novom Sadu, 38(07), 903–906. https://doi.org/10.24867/23be25simic

[26] Tanwar, S. (2024). Machine Learning. In Computational Science and Its Applications (pp. 13–42). Apple Academic Press. https://doi.org/10.1201/9781003347484-2

[27] Tran, D. T., & Huh, J. H. (2023). New machine learning model based on the time factor for e-commerce recommendation systems. Journal of Supercomputing, 79(6), 6756–6801. https://doi.org/10.1007/s11227-022-04909-2

[28] Yang, L., Liu, L., Zhang, R., Hong, J., Wang, Y., Wang, J., … Hao, H. (2020). IL-8 mediates a positive loop connecting increased neutrophil extracellular traps (NETs) and colorectal cancer liver metastasis. Journal of Cancer, 11(15), 4384–4396. https://doi.org/10.7150/jca.44215

[29] Machireddy, Jeshwanth, Harnessing AI and Data Analytics for Smarter Healthcare Solutions (January 14, 2023). International Journal of Science and Research Archive, 2023, 08(02), 785-798 , Available at SSRN: http://dx.doi.org/10.2139/ssrn.5159750

[30] Yu, H., & Zahidi, I. (2023). Tailings Pond Classification Based on Satellite Images and Machine Learning: An Exploration of Microsoft ML.Net. Mathematics, 11(3). https://doi.org/10.3390/math11030517