XTab-CDS: A Lightweight Framework for Explainable Clinical Decision Support on Tabular Data

Adelia Manafov[1], Artem Mozharov[1], Khawla Elhadri[1,*], Ramona Simon[2], Jan Gohe[2], Max Blümer[2], Ben Williges[2], Jörg Schlötterer[1], Diana Arweiler-Harbeck[2], Benedikt Höing[2] and Christin Seifert[1]

[1]*Marburg University, Germany*
{manafov, mozharoa}@students.uni-marburg.de {khawla.elhadri, joerg.schloetterer, christin.seifert}@uni-marburg.de
[2]*University Hospital Essen, Germany*
{Ramona.Simon, Jan.Gohe, Max.Bluemer, Ben.Williges, Diana.Arweiler-Harbeck, Benedikt.Hoeing}@uk-essen.de

Abstract

Explainable clinical outcome prediction on structured patient data is increasingly important for healthcare applications, as clinicians require not only accurate predictions but also transparent and actionable explanations. The explainable tabular clinical decision support system (XTab-CDS) is a modular framework explicitly designed to support explainable AI (XAI) in clinical decision support systems. It separates dataset preprocessing, model integration, and explanation generation into configurable adapters, allowing researchers and clinicians to plug in different models and explanation methods without modifying the core infrastructure. With cochlear implant outcome prediction as demonstration, we show how domain-specific adapters and pluggable explainers such as SHapley Additive exPlanations (SHAP) can be integrated. The framework also supports alternative explainability methods, both model-specific and model-agnostic, and provides a web-based interface with interactive visualizations and clinician feedback. The source code is available at https://github.com/aix-group/hear-ui/.

Keywords

medical decision support, software framework, explainable AI, feature importance

1. Introduction

Machine learning–based clinical outcome prediction has the potential to improve patient care by providing accurate probability estimates for different outcomes. However, beyond predictive accuracy, clinicians require transparent, interpretable, and actionable explanations that can be directly used in their workflows [1]. This need has driven the development of explainable AI (XAI) methods, which show the reasoning behind model predictions.

Existing XAI solutions in healthcare are often tightly coupled to specific datasets, models, and explanation methods, which limits their reuse across different medical domains [2, 3]. XTab-CDS (explainable tabular clinical decision support system) addresses this limitation by providing a lightweight, modular, explainability-centered framework where dataset preprocessing, model integration, and explanation generation are separated into configurable adapters. This design enables seamless integration of different clinical prediction tasks and different explanation requirements using the same backend and frontend infrastructure. With cochlear implant outcome prediction as our demonstration use case, we show how domain-specific adapters and pluggable explainers such as SHAP [4] can be incorporated without modifying the core system. The framework is explanation-, task-, and model-agnostic, making it a flexible platform for structured clinical datasets. By emphasizing XAI from the start, XTab-CDS ensures that predictions are not only accurate but also transparent and interpretable for clinical users, fostering trust and actionable decision support. The reference implementation is available as an extensible Python package with modular explainers and a REST API.

Late-breaking work, Demos and Doctoral Consortium, colocated with the 4th World Conference on eXplainable Artificial Intelligence: July 01–03, 2026, Fortaleza, Brazil
*Corresponding author.

2. Related Work

As clinical decision support systems become increasingly data-driven, transparent and inspectable reasoning has been identified as essential for safe and responsible deployment [1]. Trust, accountability, and clinician oversight are central requirements for the integration of AI into medical practice [2, 3]. Effective clinical decision support systems must additionally integrate into existing workflows and provide recommendations at the time and location of decision-making [5, 6].

A wide range of explanation methods has been proposed to improve model interpretability. Feature-attribution techniques such as SHAP [4] and Local Interpretable Model-agnostic Explanations (LIME) [7] provide local explanations for individual predictions. Beyond individual methods, dedicated XAI frameworks facilitate the systematic application of explainability techniques across different model architectures. A recent benchmarking study by Le et al. [8] identifies Captum [9] and Quantus [10] as comparatively mature toolkits, highlighting their support for diverse data modalities and a broad range of explanation methods.

Most existing XAI methods and toolkits, however, primarily target model development and offline evaluation. In contrast, clinical decision support applications must also address maintainability, validation, and reproducibility considerations [3]. From a software engineering perspective, containerization technologies such as Docker [11] and modern Continuous Integration/Continuous Deployment (CI/CD) practices support reliable deployment and version control in regulated environments. Nevertheless, few systems explicitly decouple dataset preprocessing, model integration, and explanation generation in a manner that enables reuse across heterogeneous clinical prediction tasks.

In contrast to prior work that primarily advances explanation algorithms or develops task-specific intrinsically interpretable models, `XTab-CDS` focuses on operational integration. By separating data adapters, model adapters, and explainer adapters within a unified backend and frontend architecture, the framework enables systematic comparison of explanation methods while remaining reusable across different clinical datasets. Rather than proposing a new explainability technique, `XTab-CDS` provides a modular infrastructure for embedding existing XAI methods into standalone, deployable clinical decision support applications.

3. Framework Architecture

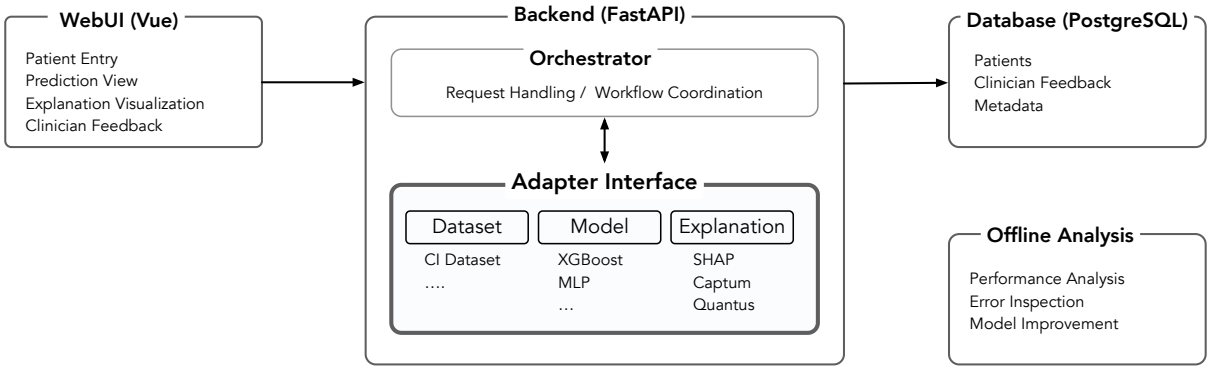

Figure 1: Architecture overview. `XTab-CDS` consists of frontend, backend, and a database tier. The backend contains adapter interfaces to datasets, machine learning models, and explanation methods, which allow i) to update models and exchange explanation methods and ii) to adapt to any clinical prediction tasks on tabular data. Predictions and explanations are computed on demand and not persistently stored. Clinician feedback is stored and used for offline evaluation and future model improvement.

To support reusable and modular integration of explainability in clinical decision support, `XTab-CDS` is designed as a modular framework for clinical outcome prediction that separates domain-specific concerns from reusable infrastructure (cf. Figure 1). The key architectural contribution is the explicit

decoupling of datasets, prediction models, and explanation methods via standardized adapter interfaces, enabling reuse across heterogeneous clinical prediction tasks. This separation allows the same core components to support different clinical prediction tasks with minimal reconfiguration.

XTab-CDS is a three-tier architecture and uses a Vue.js frontend with TypeScript, a FastAPI backend with Python, and a PostgreSQL database for persistent storage. All components are containerized using Docker to ensure reproducible deployments across development, staging, and production environments. We adopted a Representational State Transfer (REST) architecture and implemented a RESTful Application Programming Interface (API) to enable independent frontend and backend development, facilitate integration with hospital information systems, and support maintainability and future extensibility.

3.1. Modular Adapter Architecture

The adapter architecture consists of three primary components that encapsulate use-case-specific logic: dataset adapters, model adapters, and explainer adapters.

Dataset adapters implement a common interface for data preprocessing, feature engineering, and validation. They transform raw patient data from domain-specific formats into standardized feature vectors expected by the model. Each adapter defines its own feature schema, handles missing values according to domain conventions, and applies encoding strategies such as one-hot encoding for categorical variables or normalization for continuous measurements. Domain adaptability is achieved through a *schema-agnostic patient storage design*. Instead of storing each patient input feature in separate typed relational database columns, all input features are persisted within a single JSON column (`input_features`). This approach eliminates the need for database schema migrations when adapting the application to different clinical domains. Task-specific preprocessing and feature interpretation are delegated to a dedicated `DatasetAdapter`, which operates on the stored JSON structure and transforms it as required for the respective predictive model.

Model adapters wrap machine learning models behind a unified interface that exposes prediction and probability estimation methods. This abstraction allows support for linear models, tree-based ensembles, and neural networks without changes to the API layer. Model adapters handle model loading, input validation, and output formatting to ensure consistent prediction outputs.

Explainer adapters provide pluggable explanation methods, including individual approaches such as SHAP [4] as well as integration points for frameworks such as Captum [9] and Quantus [10].

3.2. System Implementation

The *backend* is organized into distinct modules that separate concerns and facilitate testing. The core module coordinates between dataset adapters, model adapters, and explainers, and manages feature configuration and metadata. The backend exposes *RESTful endpoints* for patient management (Create, Read, Update, Delete (CRUD)), individual and batch prediction with optional explanations, structured clinician feedback, and system monitoring. The database module uses SQLModel ORM with Alembic migrations, and the test module includes comprehensive coverage for unit, integration, and API tests.

The *frontend* provides an intuitive interface for clinicians, supporting validated patient data entry, real-time visualization of predictions with confidence indicators, interactive feature-attribution charts, structured clinician feedback submission, and bilingual operation (German and English; the current implementation supports left-to-right Latin-script locales, with further language extensions possible through the existing localization infrastructure).

3.3. Feedback and Continuous Improvement

Following recommendations for responsible medical AI [3], XTab-CDS includes a feedback mechanism: Clinicians can indicate binary agreement or disagreement with a prediction and optionally provide free-text comments. All feedback is persistently stored in the PostgreSQL database and linked to specific patients and predictions, creating an auditable trail for monitoring performance, identifying systematic errors, supporting retraining, and enabling continuous quality improvement.

4. Example Clinical Use Case

We exemplify our framework on a clinical task: outcome prediction for cochlear implantation. Cochlear implants (CIs) enable auditory perception in individuals with severe-to-profound sensorineural hearing loss [12], yet outcomes vary substantially across patients. Outcomes are influenced by a combination of clinical, demographic, and device-related factors. Known examples include age at implantation, duration of deafness, implant characteristics, and pre-operative speech performance [13]. Prior studies identify duration of deafness, age at implantation, and pre-operative speech scores as key prognostic variables, with most approaches relying on structured, tabular clinical data [13]. Recent studies increasingly apply supervised machine learning approaches to predict cochlear implant outcomes [14, 15, 16, 17]. For example, Crowson et al. [14] and Shew et al. [15] demonstrate the feasibility of predicting post-operative speech perception using structured clinical variables. Wang et al. [16] incorporate imaging-derived features to forecast language development in pediatric patients, while Zeitler et al. [17] apply machine learning to predict acoustic hearing preservation following cochlear implant surgery. Across these works, prediction is predominantly based on structured, tabular clinical data, further motivating the focus of `XTab-CDS` on explainable prediction models for tabular medical datasets.

4.1. Data and Machine Learning Models

Data. The dataset includes demographics, pre-operative symptoms, audiological measures, medical imaging, etiology, duration/onset of hearing loss, implant data, and pre-operative outcome measures. After preprocessing, each patient is represented by 39 tabular features. The prediction target is post-operative speech perception success (binary), measured 24 months after implantation.

Model. We implement models in Python using scikit-learn; for this demonstration we use a Random Forest classifier trained on $N = 235$ patients. With 5-fold cross-validation, the model has an average accuracy of 62% and an F1 score of 0.55. These results demonstrate technical feasibility; however, the model was trained on a relatively small dataset, and model optimization is not the focus of this work. The model outputs a probability estimate representing the probability of a successful outcome given the patient information. To mitigate overconfident predictions, probabilities are clipped to the range [1%, 99%]. This clipping constitutes a deliberate design choice rather than a standardized methodological requirement: it follows established practice in clinical decision support systems [1], which emphasizes explicit communication of residual uncertainty and discourages absolute certainty claims (0% or 100%) in medical decision-making. By avoiding extreme probability outputs, the system reinforces interpretation of predictions as supportive evidence rather than deterministic conclusions.

4.2. Framework Customization

Adapting `XTab-CDS` to a new clinical task on tabular data only requires implementing three domain-specific components: a `DatasetAdapter` defining the feature schema and preprocessing logic, a `ModelAdapter` providing the trained prediction model, and `configuration files` for the frontend specifying UI labels and input field definitions.

Data and Model. Two adapter interfaces customize the framework for a specific clinical task, and information about the machine learning model needs to be injected into the framework. The *DatasetAdapter* defines the feature schema, field names, data types, and preprocessing logic matched with a trained model. For the CI use case, the `RandomForestDatasetAdapter` implements the preprocessing pipeline, converting raw patient data with human-readable field names into the feature vector expected by the Random Forest model. This includes label encoding of categorical features, normalization of numeric features, and domain-specific feature engineering. The *ModelAdapter* provides a unified interface to the trained prediction model. The `SklearnModelAdapter` wraps the Random Forest classifier stored as a pickle file and exposes a standardized `predict()` method that accepts the

preprocessed feature vector and returns probability estimates. PyTorch, TensorFlow, or Open Neural Network Exchange (ONNX) models can be integrated using the same interface.

Model Cards are defined via structured JSON configuration files and capture essential model metadata, including model type, training data characteristics, performance metrics, intended use, limitations, and ethical considerations. The JSON file is validated on the backend using a `Pydantic` schema to ensure structural and semantic correctness before being exposed via a dedicated API endpoint. The backend renders the validated content into Markdown format and serves it to the frontend, where it is displayed within a dedicated `Vue` component. Model-card versioning is managed through a lightweight pointer mechanism using an `active_version.txt` file. Updating this pointer, different model-card versions can be activated without requiring code modifications or redeployment.

Frontend. The frontend is customized through configuration files rather than hard-coded UI logic, enabling reuse across different clinical tasks without modifying Vue.js components. *Feature Definitions* characterize the input field metadata including field names, data types (text, number, select, multi-select), validation rules, grouping (Demographics, Language & Communication, Family History, Pre-operative Symptoms, Imaging, Objective Measurements, Hearing Status, Treatment & Outcome), and available options for categorical fields. The frontend retrieves this configuration via `GET` and dynamically generates the patient entry form. Feature definitions are stored in `backend/app/config/feature_definitions.json`. *Localized Labels* provide translations for field names, section headers, and option labels in multiple languages (German/English). The frontend fetches localized strings via `GET` and applies them throughout the UI, enabling bilingual support without duplicating components. Adapting to a different clinical domain requires updating only these configuration files and the corresponding `DatasetAdapter`, the Vue components remain unchanged.

Explanation Adapter. The explanation method is configured via a backend explanation adapter (e.g., `ShapExplainer` for SHAP-based explanations). To integrate a different explanation method, developers must implement an explainer class exposing an `explain(model, x)` method that accepts (i) the trained model instance and (ii) a single preprocessed feature vector `x` (1D numeric array of length d), and returns feature attributions as structured data. The backend serializes the returned object as JSON and exposes it via a REST endpoint, while the frontend renders interactive visualizations (e.g., waterfall charts) directly from this structured response; no pre-generated images are required. If alternative visualization types are introduced, only the frontend rendering component must be adapted, while the backend interface remains unchanged. For the CI use case, we use SHAP [4] with TreeExplainer, enabling exact Shapley value computation for tree-based models in < 500 ms per patient on standard hardware, thereby supporting interactive clinical use. The returned object must follow a standardized, JSON-serializable schema with aligned, ordered lists to ensure correct frontend rendering. All lists must preserve identical ordering to guarantee correct feature–attribution alignment (e.g., using `wrapper.get_feature_names()` on the backend). The payload must not contain pre-rendered visual elements; optional visualization artifacts (e.g., base64-encoded images) may be supplied separately.

4.3. UI Walkthrough

The user interface (cf. Figure 2) guides clinicians through a structured workflow consisting of patient data entry, prediction generation, explanation exploration, and optional feedback submission. An entry page provides users with a brief usage guideline and access to the core functionalities of the system. After selecting the *Predictions* button, users can choose to search for an existing patient, create a new patient profile, or access the model card view.

Once a patient has been selected, clinicians can generate a prediction in the **Prediction View (A)**. The interface displays the predicted probability of post-operative speech perception success (here 41%), a qualitative interpretation (here "treatment success unlikely") and a visual decision-threshold chart that positions the individual patient's predicted probability along a 0–100% scale. A configurable decision threshold divides the scale into two regions: probabilities above the threshold are interpreted

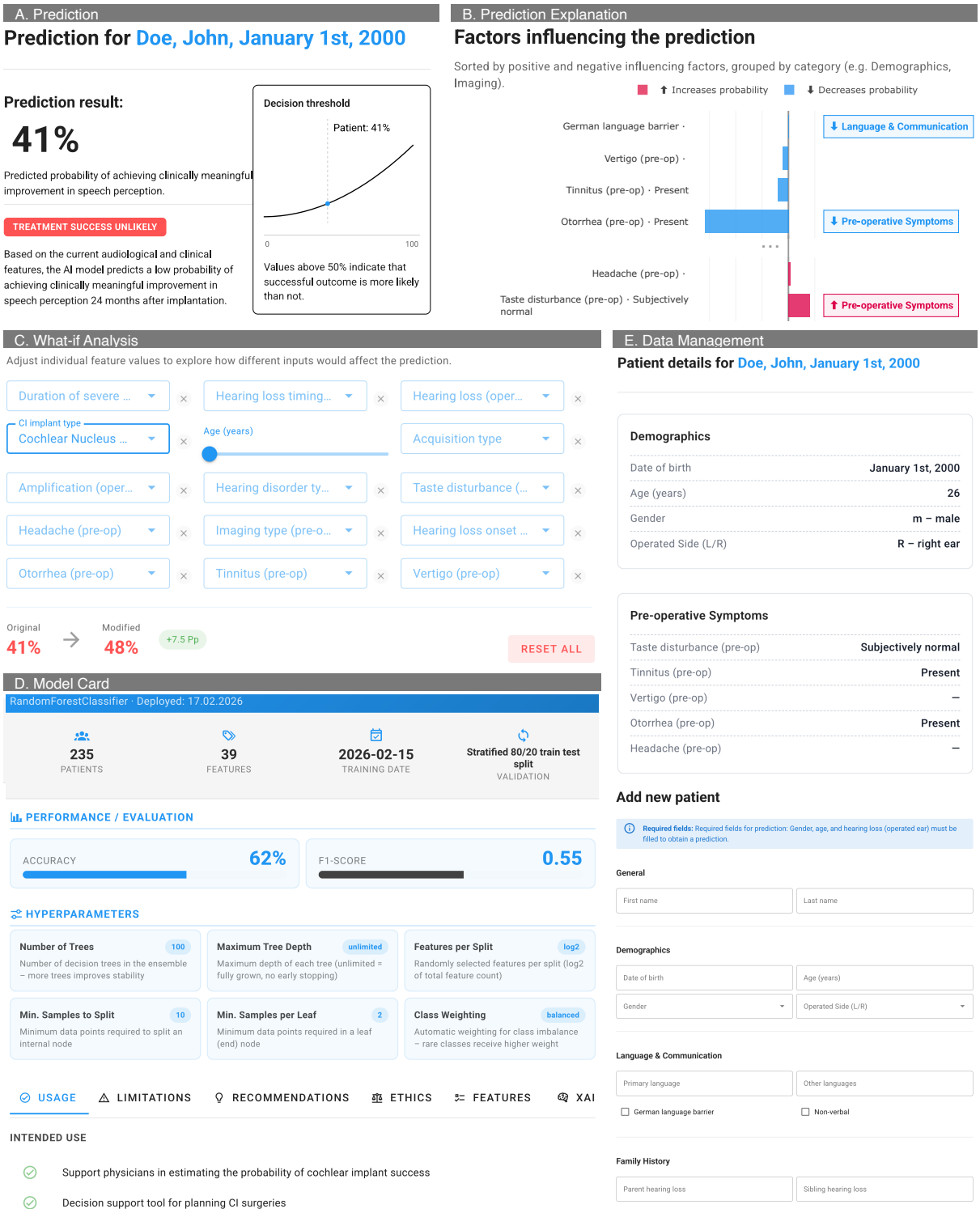

Figure 2: Overview of key UI components (output partially omitted). The prediction view includes a probability gauge, SHAP waterfall chart and feedback form. **(A)** Patient detail view with prediction, **(B)** Explanation chart, **(C)** integrated What-if analysis, **(D)** Model Card, and **(E)** patient data management.

as “treatment success likely”, and those below as “treatment success unlikely”. This visualization does not represent an empirical probability distribution; instead, it contextualizes the model output relative to a clinically meaningful decision boundary to support interpretation and decision-making. The **Explanation Chart (B)** shows the most influential features contributing to the prediction (here based on SHAP), including their positive or negative attributions. In addition, the interface integrates a **What-**

if Analysis (C), which allows clinicians to modify individual input features and dynamically observe how these adjustments affect the predicted probability. Unlike automated counterfactual explanation methods that algorithmically search for minimal feature perturbations to change a prediction [18, 19], but similar to [20], the What-if Analysis supports interactive, clinician-driven exploration guided by domain expertise. Clinicians may indicate agreement or disagreement with the prediction and optionally provide free-text comments via a feedback form. Submitted feedback is stored to facilitate ongoing model evaluation and iterative refinement. Furthermore, the system provides access to structured model documentation describing the model type, intended use, development context, performance characteristics, and known limitations by a **Model Card (D)** [21]. A Search Interface enables clinicians to locate existing patients by name or to enter information for patients not yet included in the database. All recorded patient data can be reviewed and edited. Clinicians may **create a new patient profile (E)** by completing a structured form that captures demographic, audiological, clinical, and medical imaging information. Upon submission, the patient record is stored in the system database.

5. Conclusion

XTab-CDS provides a lightweight, modular, and pluggable framework for explainable clinical outcome prediction on tabular data. By separating dataset preprocessing, model integration, and explanation generation into configurable adapters, the framework allows the same infrastructure to be reused across different medical domains. The adapter architecture allows developers to implement domain-specific components while leveraging a shared backend API, frontend interface, database schema, and deployment infrastructure. We show XTab-CDS's flexibility on a realistic clinical use case, exemplifying its easy adaptability without modifying core system code. Future work will focus on extending the framework to additional clinical domains and exploring data modalities beyond tabular data.

Acknowledgments

This research was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 536124560.

Declaration on Generative AI

Generative AI (GenAI) tools were used to refine language, improve clarity and style, correct grammar, and improve alignment with academic writing standards. Specifically, ChatGPT 5.2 Pro was used in temporary chat mode after completion of a full draft of the manuscript for the tasks "Paraphrase and Reword", "Improve Writing Style" and "Peer Review Simulation" according to the task taxonomy at https://ceur-ws.org/GenAI/Taxonomy.html. GenAI was also used to improve technical communication, such as improving code documentation. No GenAI tools were used for data analysis, statistical evaluation, model development, experimental design, or interpretation of results. All AI-assisted content was carefully reviewed, validated, and edited by the authors to ensure accuracy, originality, and compliance with academic integrity standards. The authors take full responsibility for the content of this publication.

References

[1] E. H. Shortliffe, M. J. Sepúlveda, Clinical decision support in the era of artificial intelligence, JAMA 320 (2018) 2199–2200.

[2] E. J. Topol, High-performance medicine: the convergence of human and artificial intelligence, Nature Medicine 25 (2019) 44–56.

[3] J. Wiens, S. Saria, M. Sendak, M. Ghassemi, V. X. Liu, F. Doshi-Velez, K. Jung, K. Heller, D. Kale, M. Saeed, et al., Do no harm: a roadmap for responsible machine learning for health care, Nature Medicine 25 (2019) 1337–1340.
[4] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems 30 (NeurIPS 2017), 2017, pp. 4765–4774.
[5] K. Kawamoto, C. A. Houlihan, E. A. Balas, D. F. Lobach, Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success, BMJ 330 (2005) 765.
[6] R. T. Sutton, D. Pincock, D. C. Baumgart, D. C. Sadowski, R. N. Fedorak, K. I. Kroeker, An overview of clinical decision support systems: benefits, risks, and strategies for success, npj Digital Medicine 3 (2020) 17.
[7] M. T. Ribeiro, S. Singh, C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in: 22nd Int. Conf. on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[8] P. Q. Le, M. Nauta, V. B. Nguyen, S. Pathak, J. Schlötterer, C. Seifert, Benchmarking explainable ai: A survey on available toolkits and open challenges, in: 32nd Int. Joint Conf. on Artificial Intelligence (IJCAI-23), 2023, pp. 6665–6673.
[9] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, O. Reblitz-Richardson, Captum: A unified and generic model interpretability library for PyTorch, 2020. Accessed April 2026.
[10] A. Hedström, L. Weber, D. Bareeva, F. Motzkus, W. Samek, S. Lapuschkin, M. M. C. Höhne, Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations, Journal of Machine Learning Research 24 (2023) 1–11.
[11] D. Merkel, Docker: lightweight linux containers for consistent development and deployment, Linux Journal 2014 (2014) 2.
[12] B. S. Wilson, M. F. Dorman, Cochlear implants: A remarkable past and a brilliant future, Hearing Research 242 (2008) 3–21.
[13] P. Blamey, F. Artieres, D. Başkent, F. Bergeron, A. Beynon, E. Burke, N. Dillier, R. Dowell, B. Fraysse, S. Gallégo, et al., Factors affecting auditory performance of postlingually deaf adults using cochlear implants: An update with 2251 patients, Audiology and Neurotology 18 (2013) 36–47.
[14] M. G. Crowson, P. Dixon, R. Mahmood, J. W. Lee, D. Shipp, T. Le, V. Lin, J. Chen, T. C. Y. Chan, Predicting postoperative cochlear implant performance using supervised machine learning, Otology & Neurotology 41 (2020) e1013–e1023.
[15] M. A. Shew, C. Pavelchek, A. Michelson, A. Ortmann, S. Lefler, A. Walia, N. Durakovic, A. Phillips, A. Rejepova, J. A. Herzog, P. Payne, J. F. Piccirillo, C. A. Buchman, Machine learning feasibility in cochlear implant speech perception outcomes—moving beyond single biomarkers for cochlear implant performance prediction, Ear & Hearing 46 (2025) 1266–1281.
[16] Y. Wang, D. Yuan, S. Dettman, D. Choo, E. S. Xu, D. Thomas, M. E. Ryan, P. C. M. Wong, N. M. Young, Forecasting spoken language development in children with cochlear implants using preimplant magnetic resonance imaging, JAMA Otolaryngology–Head & Neck Surgery (2025).
[17] D. M. Zeitler, Q. D. Buchlak, S. Ramasundara, F. Farrokhi, N. Esmaili, Predicting acoustic hearing preservation following cochlear implant surgery using machine learning, Laryngoscope 134 (2024) 926–936.
[18] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the GDPR, Harvard Journal of Law & Technology 31 (2018) 841–887.
[19] V. B. Nguyen, C. Seifert, J. Schlötterer, CEval: A benchmark for evaluating counterfactual text generation, in: 17th Int. Natural Language Generation Conf., 2024, pp. 55–69.
[20] E. Sciacca, G. Grasso, D. Verda, E. Ferrari, Deriving and visualizing predictive rules for disease risk: A transparent approach to medical ai, in: Joint xAI 2025 Late-breaking Work, Demos and Doctoral Consortium, 2025, pp. 273–280.
[21] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, T. Gebru, Model cards for model reporting, in: 2019 Conf. on Fairness, Accountability, and Transparency (FAT* '19), 2019, pp. 220–229.