

Demonstration: Converting your healthcare prediction model into the OHDSI framework

Jenna M Reps^{1,2}, John Xiang³, Sophia Y. Wang³, Cindy X. Cai^{4,5}, Michelle Hribar⁶, Kerry E. Ashby⁷, Egill A. Fridgeirsson²

¹Johnson & Johnson, Titusville, NJ, USA, ²Department of Medical Informatics, Erasmus University Medical Center, Rotterdam, the Netherlands. ³Department of Ophthalmology, Byers Eye Institute, Stanford University, Stanford, California, USA, ⁴Wilmer Eye Institute, Johns Hopkins School of Medicine, Baltimore, Maryland, ⁵Biomedical Informatics and Data Science, Division of General Internal Medicine, Department of Medicine, Johns Hopkins University School of Medicine, Baltimore, Maryland, ⁶Department of Ophthalmology and Visual Sciences, University of Illinois at Chicago, Chicago, IL, ⁷Topcon Healthcare Innovation Center, Topcon Healthcare, Inc. La Jolla, California.

Background

The OHDSI community has developed a flexible framework for building advanced healthcare prediction models [1]. The framework simplifies applying and externally validating models on new datasets formatted to the OMOP Common Data Model (CDM) [2]. Researchers can also integrate models developed outside the framework by plugging them into the OHDSI pipeline. Its flexible design supports complex methods such as deep learning models with autoencoder-derived features. In this demonstration, we show how an existing TensorFlow model [3-4] — which required 299 engineered features derived from hundreds of raw variables — was converted for use in the OHDSI framework, and we present a network-study package developed to validate the model across the OHDSI network.

Methods

The OHDSI Patient-Level Prediction (PLP) framework represents a model with a single `modelDesign` object that bundles all settings for cohort selection, feature extraction, preprocessing, splitting, training, and implementation. Key entries in `modelDesign`:

- `targetId`, `outcomeId` — cohort identifiers for the target population and the outcome.
- `restrictPlpData` — extraction-time filters (e.g., min prior observation, first-exposure, index date window, random sample).
- `studyPopulationSettings` — post-extraction inclusion/exclusion (e.g., exclude prior outcome, insufficient follow-up).
- `covariateSettings` — settings passed to `FeatureExtraction` [5] to generate SQL and extract covariates.
- `featureEngineeringSettings` — transformations learned on training data (e.g., selection, autoencoder); learned parameters are saved and reapplied at prediction time.
- `splitDataSettings`, `modelSettings`, `hyperParameterSettings` — data partitioning, algorithm choice, and hyperparameter search.
- `sampleSettings`, `preprocessSettings` — optional sampling and preprocessing (e.g., remove redundant features, scale by per-feature max); any removals/scalings are recorded for exact replication at prediction.

Output: the `plpModel` (R list, class "`plpModel`") minimally contains:

- `model` — fitted model object or pointer to a saved model.
- `modelDesign` — the design used to build the model.

- Preprocessing — includes requiresDenseMatrix (TRUE/FALSE) and featureEngineering steps (each with funct and settings capturing the learned transform).
- covariateImportance — data.frame mapping covariateId → columnId, which aligns FeatureExtraction outputs to model input columns.

Prediction function: plpModel must have a predictionFunction attribute that accepts (plpModel, data, cohort) and returns cohort with an added value column containing each person's predicted score.

Results

We converted an existing Python/TensorFlow glaucoma-screening model into a plpModel that implements the original model within the OHDSI PLP framework. The model repository is <https://github.com/ohdsi-studies/GlaucomaPrescreeningPrediction>. The model input combined 128 autoencoder features from drug concept ids, 128 autoencoder features from condition concept ids, demographic variables, and measurement-derived features.

Key steps

- **Covariates:** Created custom measurement covariates that extract the latest lab value in the prior 5 years and defined demographic and concept-based covariates using standard FeatureExtraction settings (see repository files measurementFeatures.R and plpModel.R).
- **Feature engineering:** Implemented feature-engineering functions (autoEncoders.R) that
 - load a saved ordered list of conceptIds,
 - build a sparse person × concept matrix (training data),
 - batch-convert to dense and apply the saved autoencoder to produce 128 compressed features per person,
 - append these autoencoder features to the training data; learned transformations and parameter locations are recorded in featureEngineeringSettings and referenced by the plpModel.
- **Model integration and prediction:** Added a prediction function that converts incoming data to the required sparse matrix, loads the Keras model, predicts in batches (converting batches to dense as needed), and returns scores.
- **Feature mapping:** Supplied covariateImportance (covariateId → columnId) so extracted covariates align with the model input columns.

Outcome

The resulting plpModel encapsulates the original Python/TensorFlow model, its feature engineering, and the mapping needed for consistent scoring within the OHDSI framework, enabling reuse and network-wide validation.

Conclusions

We successfully converted an external Python TensorFlow model into the OHDSI PLP framework by wrapping its feature extraction, learned feature-engineering, and prediction steps.

A network-study package (GlaucomaPrescreeningPrediction) automates deployment on any OMOP CDM-mapped database: after providing connection details, CDM and working schemas, the code

extracts covariates, applies the recorded feature transformations, scores each person, and returns evaluation metrics — enabling consistent reuse and network-wide external validation.

We believe this process can be implemented for most models developed outside of the OHDSI PLP framework.

References

- [1] Reps, J.M., Schuemie, M.J., Suchard, M.A., Ryan, P.B. and Rijnbeek, P.R., 2018. Design and implementation of a standardized framework to generate and evaluate patient-level prediction models using observational healthcare data. *Journal of the American Medical Informatics Association*, 25(8), pp.969-975.
- [2] Reps, J.M., Williams, R.D., You, S.C., Falconer, T., Minty, E., Callahan, A., Ryan, P.B., Park, R.W., Lim, H.S. and Rijnbeek, P., 2020. Feasibility and evaluation of a large-scale external validation approach for patient-level prediction in an international data network: validation of models predicting stroke in female patients newly diagnosed with atrial fibrillation. *BMC medical research methodology*, 20(1), p.102.
- [3] Ravindranath, R. and Wang, S. (2024) 'Artificial Intelligence Models to Identify Patients with High Probability of Glaucoma Using Electronic Health Records', *Ophthalmology Science*, 5
- [4] Xiang, J., Ravindranath, R. and Wang, S.Y. (2026) 'Validating a Deep Learning Algorithm to Identify Patients with Glaucoma using Systemic Electronic Health Records', arXiv preprint, arXiv:2604.20921. Available at: <https://arxiv.org/abs/2604.20921> (Accessed: 21 May 2026).
- [5] Observational Health Data Sciences and Informatics (OHDSI) FeatureExtraction. R package. Available at: <https://github.com/OHDSI/FeatureExtraction> (Accessed: 15 May 2026).