Stroke Prediction - Classification Problem

Tests AI and Data Science at Capgemini

Borja Serrano González

E-mail: borja.serranog@gmail.com

Contents

1 Introduction: Stroke prediction

Stroke is a major public health concern worldwide. According to the World Health Organization (WHO), it is the second leading cause of death globally, accounting for approximately 11% of total deaths. Early detection and prevention of strokes can significantly reduce mortality and improve quality of life, making predictive modeling an important tool in modern healthcare systems.

In this project, a binary classification problem is addressed, focusing on predicting whether a given patient will suffer a stroke based on a set of clinical and demographic characteristics. The data set used originates from a public source available on Kaggle and contains anonymized patient health records [1]. Each row in the dataset represents an individual patient, with multiple features that include age, gender, presence of hypertension or heart disease, smoking status, and other relevant information.

The objective is to develop a machine learning model capable of identifying patients at risk of stroke using the available features, representing a real-world challenge in the healthcare domain. The dataset includes the following attributes:

- **id**: Unique identifier for each patient.

- **gender**: Categorical feature with values `"Male"`, `"Female"`, or `"Other"`.
- **age**: Age of the patient (numerical).
- **hypertension**: Binary feature, 0 if the patient does not have hypertension, 1 otherwise.
- **heart_disease**: Binary feature, 0 if the patient does not have heart disease, 1 otherwise.
- **ever_married**: `"Yes"` or `"No"`.
- **work_type**: Type of employment, with values `"children"`, `"Govt_job"`, `"Never_worked"`, `"Private"`, or `"Self-employed"`.
- **Residence_type**: `"Rural"` or `"Urban"`.
- **avg_glucose_level**: Average glucose level in the blood (numerical).
- **bmi**: Body Mass Index (numerical).
- **smoking_status**: `"formerly smoked"`, `"never smoked"`, `"smokes"`, or `"Unknown"`.
- **stroke**: Target variable; 1 if the patient had a stroke, 0 otherwise.

In particular, the `smoking_status` feature may contain the value `"Unknown"`, indicating that the information is not available for certain patients. Handling such missing values appropriately is part of the preprocessing and modeling pipeline described in the following sections.

2 MLOps and CI/CD Integration

To ensure the reproducibility, maintainability, and scalability of the project, an MLOps workflow was implemented using GitHub Actions for continuous integration (CI) and continuous deployment (CD). This pipeline automates code quality checks, testing, and documentation deployment.

2.1 Environment and Dependency Management

The project environment is managed using `Poetry` (version 2.1.3), which handles package dependencies and virtual environments in a reproducible and declarative manner. The Python version used throughout the CI workflow is `3.12.11`, ensuring consistency across development, testing, and production environments.

2.2 Continuous Integration Pipeline

The CI pipeline is defined in a GitHub Actions workflow file (`.github/workflows/ci.yml`) and is automatically triggered on every push and merge of branches (see Figure 1).

It consists of three main tasks that are summarized in Figure 2.

Each of the steps in this pipeline can be described as follows.

- **Code Quality Checks:** This job performs static analysis using several tools:
  - `black` for code formatting,
  - `ruff` for linting,
  - `mypy` for static type checking,
  - `vulture` for detecting unused code.

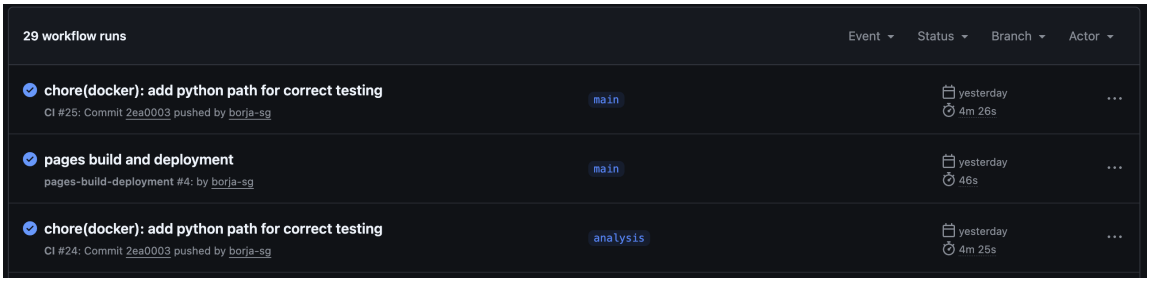

Figure 1: CI pipeline runs on GitHub actions.

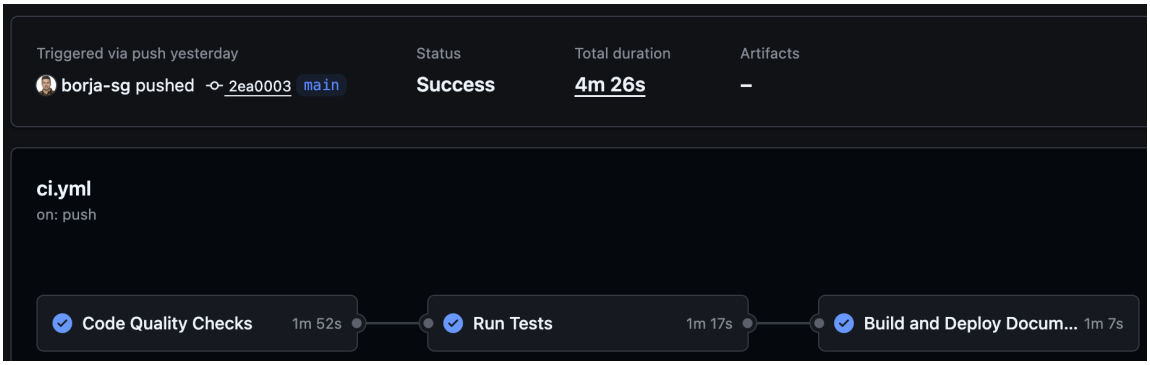

Figure 2: Continuous integration pipeline.

These tools are also integrated locally through `pre-commit` hooks, ensuring developers adhere to quality standards before committing changes.

- **Testing:** Unit tests are written using `pytest` (version `8.2.1`). Tests are executed automatically in the pipeline to validate correctness of the codebase before deployment.
- **Documentation Deployment:** Project documentation is written using `MkDocs`. During the final job of the pipeline, the documentation is built and automatically deployed to GitHub Pages using the `peaceiris/actions-gh-pages` action. The documentation is deployed at the following url: https://borja-sg.github.io/BSG-TestDataScience-1/.

2.3 Experiment Tracking and Optimization

Hyperparameter optimization is conducted using `Optuna` [2], a state-of-the-art framework for efficient sampling-based search. Model training runs and experiment metadata are tracked using `MLflow` [3], which provides logging, visualization, and reproducibility of results. Both tools are seamlessly integrated into the development cycle to ensure traceability of model improvements.

The combination of version control, CI/CD automation, code quality enforcement, and experiment tracking results in a robust MLOps pipeline. This approach enables collaborative and transparent development, supporting reliable deployment of machine learning workflows.

3 Exploratory Data Analysis (EDA)

A separate document containing the full Python notebook used for this analysis, including the code, visualizations, and outputs, is provided alongside this report (notebooks/eda.ipynb). This section summarizes the most relevant insights and processing steps obtained during the exploratory data analysis.

The dataset used in this analysis contains information for 5110 patients. The target variable is binary (`stroke` = 0 or 1), making this a supervised classification problem. The features include numerical and categorical variables such as age, sex, BMI, average glucose level, or smoking status.

3.1 Initial Data Inspection

After loading the dataset, it was verified that all columns except for `bmi` had complete entries. The `id` column was removed as it carries no predictive value. The dataset consisted of:

- **3 numerical features:** `age`, `avg_glucose_level`, and `bmi`.
- **4 binary features:** `hypertension`, `heart_disease`, `ever_married`, and `Residence_type`.
- **3 categorical features:** `gender`, `work_type`, and `smoking_status`.
- **1 target variable:** `stroke`.

3.2 Handling Missing and Ambiguous Data

Missing values: The `bmi` column had missing values for approximately 3.9% of the patients. These were imputed using a KNN imputer, after first encoding the categorical variables to numeric form. **Ambiguous entries:** Two categorical variables had problematic values:

- `gender` had a single instance labeled as `"Other"`, which was dropped to prevent bias.
- `smoking_status` had 1544 entries labeled as `"Unknown"`, which were converted to missing values and later imputed using KNN.

The distribution of the imputed `bmi` values was analyzed and found to be consistent with the original data, with imputed values concentrated around the mean. Similarly, the `smoking_status` variable was recovered by rounding the imputed values to the nearest valid class and restoring categorical labels.

3.3 Class Imbalance

The `stroke` variable is highly imbalanced, with 95.1% of the patients not having suffered a stroke and only 4.9% positive cases. This imbalance was addressed using stratified sampling for train/test split, ensuring the proportion of stroke cases remained consistent across both sets.

3.4 Feature Engineering

To enhance the model's predictive capability, several new categorical features were derived from continuous variables:

- **BMI Category:** Patients were classified into `underweight`, `normal`, `overweight`, and `obese` groups based on the World Health Organization established thresholds. As expected, a higher stroke prevalence was observed in the obese category.
- **Age Group:** Patients were binned into `young` (0–30), `adult` (30–45), `middle_aged` (45–60), `senior` (60–75), and `elderly` (75+). Stroke incidence increased with age, with the highest risk seen in the elderly group.
- **Glucose Category:** Based on the average glucose level, patients were labeled as `normal` (< 99), `prediabetic` (99–125), or `diabetic` (> 125). As expected, diabetic individuals had a higher probability of stroke.

- **Health Score and Risk Category:** A health risk score (0–5) was constructed by assigning 1 point each for the presence of the following risk factors: age > 60, high glucose, smoking, hypertension, and heart disease. Based on this score, patients were grouped into `low`, `moderate`, and `high` risk categories. This feature showed strong correlation with stroke incidence, confirming its utility in classification.

3.5 Correlation Analysis and Visualization

Pairwise scatter plots and a correlation heatmap were used to analyze relationships between continuous variables (`age`, `avg_glucose_level`, `bmi`) and the target. The correlation matrix revealed moderate relationships among some variables (e.g., age and glucose), but no strong multicollinearity.

3.6 Data Preparation for Modeling

After feature engineering and imputing missing values, all categorical variables were encoded numerically. The dataset was then split into training and test sets using a stratified sampling strategy to maintain the proportion of stroke cases. The final datasets were saved in HDF5 format for use in the modeling phase.

This exploratory data analysis confirmed known clinical risk factors for stroke and provided a richer feature space through domain-informed transformations. The processed data is now ready for model development.

4 Model Selection and Hyperparameter Optimization

To identify the most suitable model for the stroke prediction task, a systematic approach to model selection and hyperparameter optimization was implemented using the `Optuna` framework [2]. Given the highly imbalanced nature of the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) [4] was applied prior to training to balance the class distribution by generating synthetic samples for the minority class. This step was essential to mitigate the bias towards the majority class and improve the model's ability to detect positive cases.

4.1 Search Space and Candidate Models

The search space included a diverse set of classifiers, each with distinct learning principles and strengths:

- **Random Forest**: An ensemble method based on decision trees that uses bagging and feature randomness to improve predictive accuracy and control overfitting [5].
- **Logistic Regression**: A simple yet effective linear model for binary classification that estimates probabilities using the logistic function [6].
- **Support Vector Machine (SVC)**: A kernel-based method that constructs a maximum-margin hyperplane in high-dimensional space for classification tasks [7].
- **Multi-layer Perceptron (MLP)**: A feedforward neural network composed of multiple layers with nonlinear activation functions, trained using backpropagation [8].
- **Extreme Gradient Boosting (XGBoost)**: A highly efficient implementation of gradient-boosted decision trees designed for performance and scalability [9].

Each model was associated with a configuration dictionary defining its tunable hyperparameters. Optuna's dynamic sampling methods were used to explore this space efficiently. The parameters included the number of estimators, regularization coefficients, kernel types, learning rates, network architectures, among others.

4.2 Optimization Objective

The optimization objective was to maximize the F1-score via k-fold cross-validation with 3 folds. The F1-score is the harmonic mean of precision and recall, given by

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{4.1}$$

where precision is the proportion of true positive predictions among all positive predictions, and recall is the proportion of true positives detected among all actual positives. The F1-score was chosen as the primary metric because it provides a balanced measure of a model's accuracy on imbalanced datasets, unlike accuracy which can be misleading when the classes are unevenly distributed.

For each Optuna trial:

1. A classifier type was sampled from the candidate pool.
2. The corresponding hyperparameters were suggested according to predefined search spaces (categorical, float, or integer).
3. A pipeline was constructed with standardized features using `StandardScaler`, followed by the SMOTE oversampling step, and then the classifier.
4. The model was evaluated using 3-fold cross-validation.

The average F1-score across folds was returned as the trial's objective value. The optimization process was guided by Optuna's Tree-structured Parzen Estimator (TPE) sampler.

4.3 Experiment Tracking

Each trial was tracked using `MLflow`. Logged information included:

- Model type and hyperparameters.
- Cross-validation performance scores (F1-score).
- Optimization metric value.

MLflow's experiment tracking enabled systematic comparison and reproducibility of all trials. The optimization was run in a nested fashion so that each trial corresponds to a separate MLflow run.

4.4 Fittest Model

The best-performing model and its optimal hyperparameters were selected based on the highest mean F1-score across 3-fold cross-validation. Figure 3 shows the tracking of the entire experiment using `MLflow`, including the F1-score achieved by each evaluated model, the execution timeline of all trials, and the final outcome corresponding to the selected model.

The optimal model identified was a **Multi-layer Perceptron (MLP)** with a mean cross-validation F1-score of **0.972** (note that the cross validation set were also balanced using SMOTE). The corresponding hyperparameter configuration is summarized as follows:

- **Classifier:** MLP
- **Number of hidden layers:** 3
- **Neurons per layer:** [109, 90, 91]

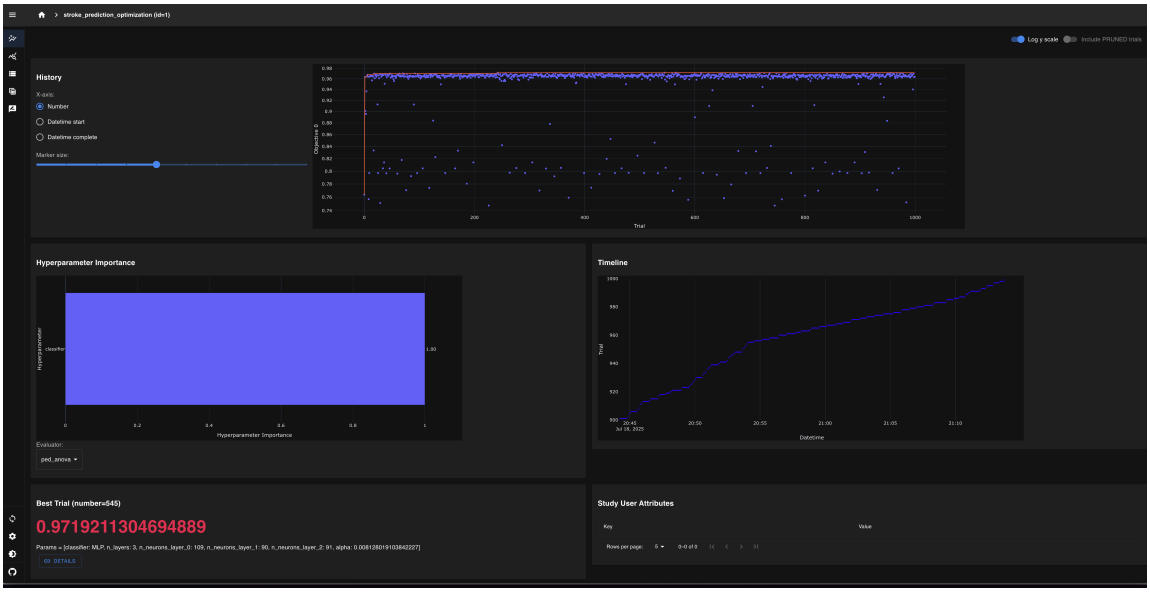

Figure 3: Visualization of the hyperparameter optimization results using `Optuna`, with full experiment tracking via `MLflow`. Each trial corresponds to a different model configuration, and the F1-score is used as the performance metric.

- **Regularization parameter α:** 0.0081

This architecture likely achieved superior performance due to its capacity to model complex nonlinear relationships in the feature space, which are essential for correctly distinguishing between stroke and non-stroke cases, especially in a dataset with imbalanced classes. The combination of SMOTE oversampling and the F1-score optimization criterion ensured that the model maintained both high precision and recall, thus minimizing both false positives and false negatives.

The final model, trained on the SMOTE-balanced dataset and standardized features, is now ready for deployment or further evaluation on a held-out test set or real-world data. The trained pipeline can be reused or exported for further evaluation or deployment. This approach allowed for a structured and automated way to balance model complexity, generalization capability, and computation cost, making the model selection phase both efficient and reproducible.

5 Model Evaluation

The model evaluation was performed in the notebook "*notebook/evaluation.ipynb*" (more details can be found there). The testing data was loaded and the MLP saved with MLFlow was tested. The analysis includes the confusion matrix, relevant metrics for classification, ROC curve, and AUC score to assess performance.

The confusion matrix is shown in Figure 4. From the confusion matrix, it could be seen that while the model is quite accurate in identifying patients who will not suffer the stroke, it struggles with the positive cases. This reflects the complex nature of the dataset, where most of the samples are negative.

We can also evaluate the model with other metrics relevant for imbalanced classification: precision, recall, and f1-score (Table 1). These results indicate that while the model is effective at identifying non-stroke individuals, it fails to reliably detect stroke cases. The low recall for the stroke class means that most actual stroke cases are missed (only 11 out of 75 correctly identified), and the low precision implies that many of the stroke predictions are false positives.

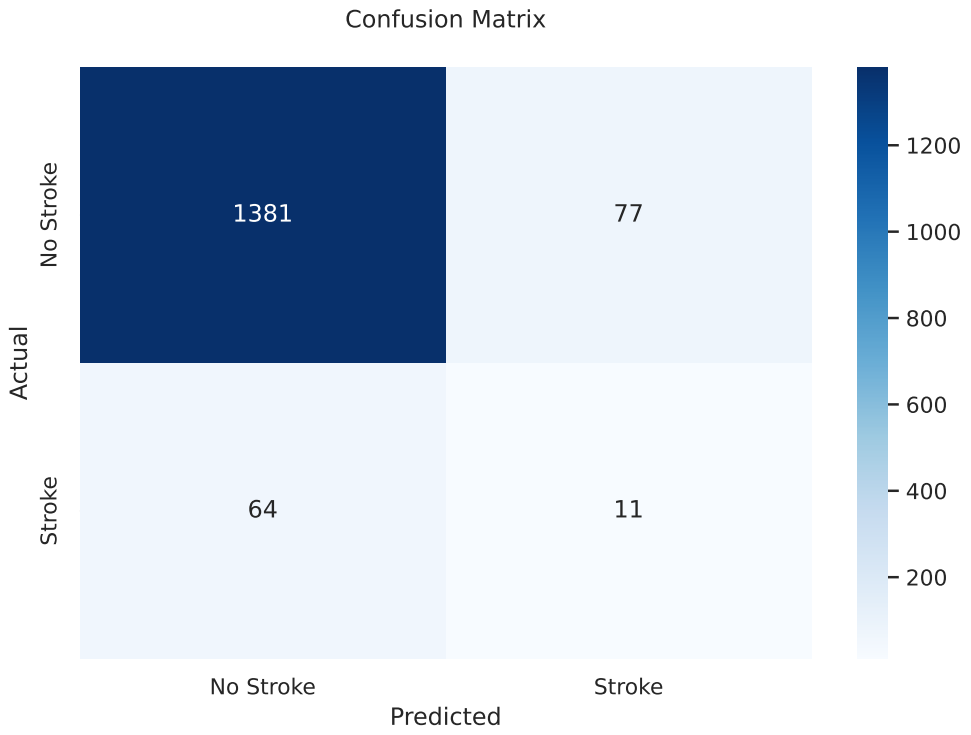

Figure 4: Confusion matrix using the MLP to predict the test dataset.

Class	Precision	Recall	F1-score
No Stroke	0.956	0.947	0.951
Stroke	0.125	0.147	0.135
Accuracy	0.908	0.908	0.908
Macro avg	0.540	0.547	0.543
Weighted avg	0.915	0.908	0.911

Table 1: Classification report metrics for the stroke prediction model.

This result shows that, even when balancing the dataset using SMOTE, the models potentially struggle generalizing data from the minority class, mainly because the data is not enough for the model to learn how to identify such events. Even though SMOTE was used to synthetically balance the dataset during training, the model remains biased toward the majority class. Future improvements could include enhanced resampling strategies, cost-sensitive learning, or specialized architectures better suited for imbalanced data.

Finally, the Receiver Operating Characteristic (ROC) curve is presented in Figure 5. The model achieved an Area Under the Curve (AUC) of 0.70, indicating a moderate ability to distinguish between stroke and non-stroke cases.

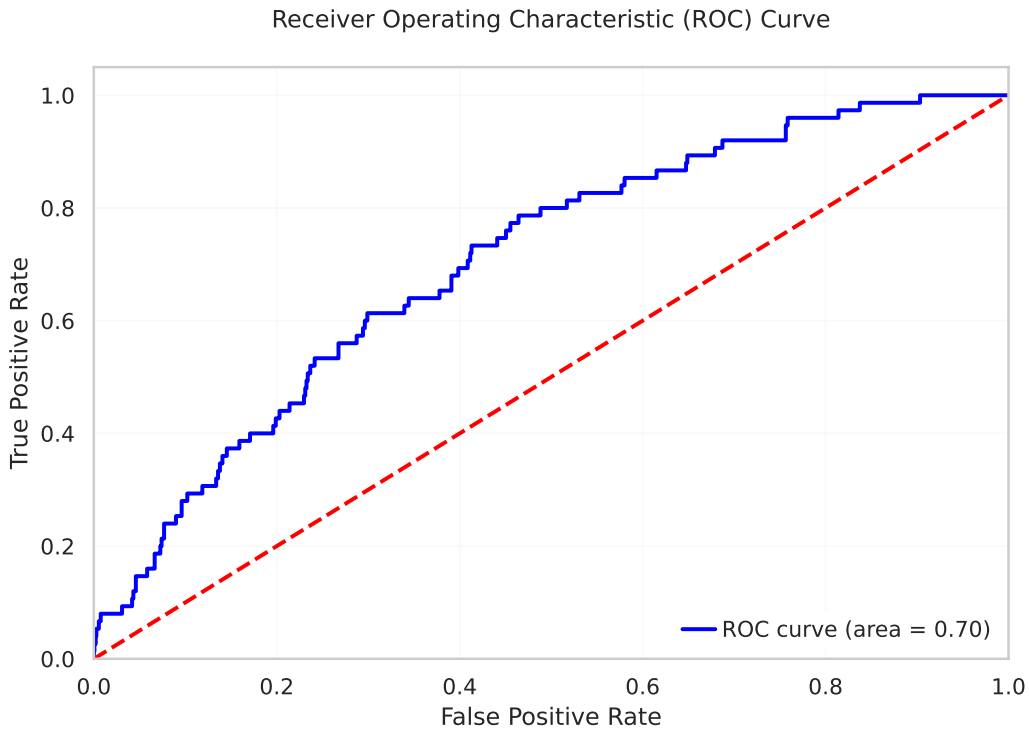

Figure 5: ROC curve using the MLP to predict the test dataset.

6 Model Deployment for a Real-Case Scenario

To bridge the gap between model development and clinical applicability, the trained stroke prediction model was deployed using a lightweight and accessible web service. This was done using the `FastAPI` framework, which allows for high-performance asynchronous APIs with automatic interactive documentation.

The application provides two main interfaces:

- A REST API endpoint (`/predict`) that receives structured data (in JSON format) and returns a stroke risk prediction along with its associated probability.
- A user-friendly web interface (`/web`) that enables medical professionals to enter patient data through a form and receive real-time feedback on stroke risk.

This deployment ensures that doctors can easily access the model without requiring any knowledge of the underlying machine learning infrastructure. Upon entering key patient features such as age, BMI, glucose levels, hypertension status, and lifestyle information (e.g., smoking habits), the system returns a probability estimate for stroke risk, helping to inform clinical judgment.

Figure 6 presents a case of an elderly individual with multiple risk factors including smoking, hypertension, and heart disease. The model yields a high stroke probability, as expected.

The backend system leverages a pre-trained model stored and tracked via `MLflow`, ensuring reproducibility and version control. The input data is preprocessed using the same transformations as

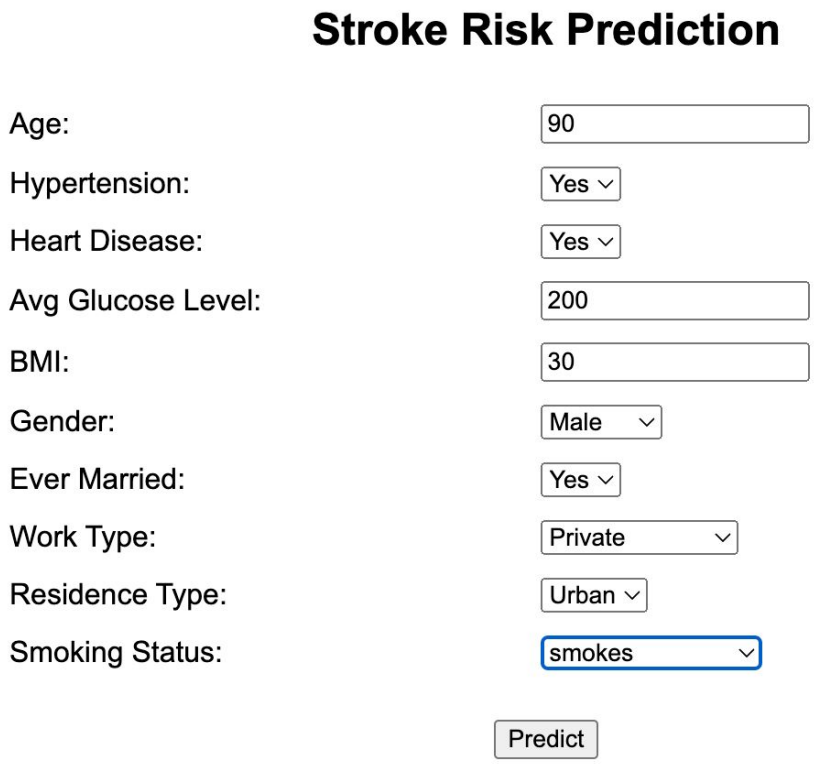

Figure 6: Prediction example for an older individual with multiple risk factors (e.g., smoking, hypertension). The model assigns a high stroke probability.

in the training pipeline (e.g., encoding categorical variables and applying SMOTE during training). This guarantees consistent behavior between development and production.

The use of FastAPI for deployment provides both flexibility and scalability, making it suitable for integration into broader medical systems or hospital intranets. This real-case deployment demonstrates how machine learning models can be effectively translated into actionable tools that support healthcare decision-making.

7 Conclusions

A supervised classification model was developed to predict the risk of stroke based on patient features, using a variety of machine learning algorithms and systematic hyperparameter optimization.

To address the class imbalance in the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) was applied during training. The F1-score was chosen as the optimization metric due to its suitability for imbalanced datasets, as it balances precision and recall.

The best-performing model was a Multi-layer Perceptron (MLP) with three hidden layers and tuned regularization. Its performance was evaluated using a confusion matrix and classification metrics.

These results highlight that the model performs very well on the majority class but struggles with detecting stroke cases, which are rare. Despite oversampling, many stroke cases are still missed (low recall), and most stroke predictions are incorrect (low precision).

The model was deployed using FastAPI, exposing both an API endpoint and a web interface tailored for medical use. This setup allows real-time stroke risk predictions based on patient data entered by healthcare professionals.

Although the system is functional and accessible, further work is needed to improve the sensitivity of the model to stroke cases before deployment in critical clinical environments. Future improvements may include ensemble methods, cost-sensitive training, or leveraging additional patient history features.

References

[1] "Stroke prediction dataset — kaggle.com." https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/data.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, *Optuna: A next-generation hyperparameter optimization framework*, in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[3] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S.A. Hong, A. Konwinski et al., *Accelerating the machine learning lifecycle with mlflow.*, *IEEE Data Eng. Bull.* **41** (2018) 39.

[4] N.V. Chawla, K.W. Bowyer, L.O. Hall and W.P. Kegelmeyer, *Smote: Synthetic minority over-sampling technique*, *J. Artif. Int. Res.* **16** (2002) 321–357.

[5] L. Breiman, *Random forests*, *Machine learning* **45** (2001) 5.

[6] D.W. Hosmer, S. Lemeshow and R.X. Sturdivant, *Applied logistic regression*, John Wiley & Sons (2013).

[7] C. Cortes and V. Vapnik, *Support-vector networks*, in *Machine learning*, vol. 20, pp. 273–297, Springer, 1995.

[8] D.E. Rumelhart, G.E. Hinton and R.J. Williams, *Learning representations by back-propagating errors*, *Nature* **323** (1986) 533.

[9] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2016.