Insurance Enrollment Prediction - Technical Report

---

1. Introduction

This project aims to predict whether an employee will opt in to a new voluntary insurance product based on demographic and employment-related features. We build a machine learning pipeline covering data preprocessing, model training, evaluation, and deployment via a REST API.

---

2. Dataset Overview

- **Source:** Synthetic employee census-style dataset
- **Size:** ~10,000 rows, representing employee records
- **Features:**
    - `employee_id` (unique identifier)
    - `age` (numeric)
    - `gender` (categorical)
    - `marital_status` (categorical)
    - `salary` (numeric)
    - `employment_type` (categorical)
    - `region` (categorical)
    - `has_dependents` (binary encoded as categorical)

  - `tenure_years` (numeric)
  - `enrolled` (target: 0 or 1)

---

3. Data Processing

- Handled missing values and ensured consistent data types.
- Encoded categorical variables using one-hot encoding for model compatibility.
- Scaled numerical features using `StandardScaler` to normalize data.
- Performed train-test split maintaining representative data distributions.
- Saved preprocessing pipeline using `joblib` for reuse during inference.

---

4. Model Development

- Chose **XGBoost** as the primary model due to its proven performance on tabular data and ability to handle mixed feature types.
- Trained the model on processed training data.
- Hyperparameter tuning was performed via MLflow to optimize model parameters and track experiments.

---

5. Model Evaluation

- Evaluation metrics computed on test set:
  - Accuracy

    - Precision
    - Recall
    - F1-score
    - ROC AUC
- Visualized results with interactive Plotly charts:
    - Bar chart of evaluation metrics
    - Confusion matrix heatmap
    - ROC curve
- Saved all plots in both HTML and PNG formats for reporting.

---

6. Model Persistence

- Saved trained model and preprocessing pipeline as pickle files (`model.pkl` and `preprocessor.pkl`) for reproducibility.
- Enabled easy model loading during serving to avoid retraining.

---

7. Deployment via REST API

- Developed a **FastAPI** REST endpoint `/predict` to serve real-time predictions.
- API accepts employee features as JSON, applies preprocessing, and returns enrollment probability.
- Tested with sample `curl` commands.
- Input schema validation implemented using Pydantic models for robust request handling.

---

8. Experiment Tracking

- Integrated **MLflow** for:
    - Logging hyperparameters
    - Recording evaluation metrics
    - Storing model artifacts
- Enables reproducibility and comparison across multiple training runs.

---

9. Key Takeaways

- The XGBoost model effectively predicts voluntary insurance enrollment using demographic and employment data.
- Proper data preprocessing and feature encoding are crucial for model performance.
- Interactive visualizations aid in interpreting model strengths and weaknesses.
- REST API deployment enables practical integration of ML predictions into business workflows.
- MLflow experiment tracking facilitates systematic hyperparameter tuning and model management.

---

10. What to Do Next with More Time

- **Data Enrichment:** Incorporate additional employee features like performance ratings, historical claims, or behavioral data for better predictions.
- **Feature Engineering:** Use domain knowledge to create composite features or embeddings to capture complex relationships.

- **Advanced Models:** Experiment with deep learning models or ensemble stacking to further improve accuracy.
- **Automated Hyperparameter Optimization:** Use Bayesian optimization frameworks (e.g., Optuna) for more efficient tuning.
- **API Enhancements:** Add authentication, rate limiting, and logging for production readiness.
- **CI/CD Pipeline:** Automate testing, deployment, and monitoring of the model with continuous integration/continuous deployment tools.
- **Scalability:** Containerize the API using Docker and deploy on cloud platforms with autoscaling.
- **User Interface:** Build a simple frontend to interact with the API and visualize predictions.
- **Explainability:** Integrate SHAP or LIME for model interpretability to explain individual predictions to stakeholders.

Appendix

- **Code repository:** **https://github.com/CSKacas/Insurance-Enrollment-Prediction.git**
- **Data file:** `employee_data.csv` (synthetic dataset)
- **Environment:** Python 3.9, XGBoost, FastAPI, MLflow, Plotly