Final Report

Diabetes Risk Prediction

Using
Explainable AI (XAI)

Author : Mohamed Makki

Project : Diabetes Risk XAI
Streamlit App: **CLICK HERE**
Program : MIT Emerging Talent computer and data science program
Date : December, 2025

Overview

According to WHO, diabetes is responsible for 1.5 million deaths annually, with an additional 2.2 million deaths attributable to high blood glucose. The International Diabetes Federation (IDF) projects that by 2045, over 700 million people will be living with diabetes. The economic burden is staggering: in 2019, global health expenditure on diabetes was estimated at $760 billion, representing nearly 10% of total healthcare spending.

The burden is particularly acute in low- and middle-income countries, where healthcare systems are often ill-equipped to manage chronic diseases. In such contexts, early detection tools that are low-cost, interpretable, and accessible can play a transformative role in reducing morbidity and mortality.

This project addresses these issues by developing a **fully explainable machine-learning system** for predicting diabetes risk. The system incorporates:

- A validated dataset (Pima Indians Diabetes dataset)
- Exploratory Data Analysis (EDA)
- Baseline and tuned predictive models
- Explainability using **SHAP** and **LIME**
- A deployed **Streamlit** web application
- A clean GitHub repository and milestone-based project workflow

The final deliverable is a fully functioning risk-prediction system, accessible through a web interface, accompanied by rich interpretability visualizations and transparent documentation.

We developed an end-to-end machine learning pipeline to predict diabetes risk using the Pima Indians Diabetes dataset and to provide transparent, clinically-actionable explanations using SHAP and LIME. The pipeline includes dataset verification, EDA, baseline and tuned modeling (Logistic Regression, Random Forest, and XGBoost), model-interpretability artifacts, and an interactive Streamlit app for presentation and demonstration.

The best performing model during development was an XGBoost classifier with an accuracy ≈ **0.7468** and ROC AUC ≈ **0.8281** at baseline; after randomized hyperparameter tuning, the tuned XGBoost variant produced test accuracy ≈ **0.7338**, recall ≈ **0.6111** and ROC AUC ≈ **0.8226**. Explainability analyses highlighted **Glucose**, **BMI**, and **Age** as the most influential features. Artifacts (models, scalers, SHAP/LIME HTML outputs) and notebooks are stored in the repository and results folder for reproducibility and demonstration.

Table of contents

1. Introduction & motivation

Diabetes is a major global health challenge. Early detection and intervention substantially reduce the risk of long-term complications and healthcare costs. Predictive systems that identify individuals at high risk enable screening, referral, and lifestyle or clinical interventions. However, for clinical adoption, a predictive system must be both accurate and transparent, clinicians must understand why a prediction was made.

This project builds an explainable machine learning system that predicts diabetes risk from routinely-collected clinical features and surfaces interpretable explanations for individual predictions. The project focuses on three core objectives:

- Build robust predictive models (baseline → tuned) for diabetes risk classification.
- Generate explainability artifacts (SHAP, LIME) to make model decisions transparent.
- Produce a clean, reproducible project structure and a Streamlit-based demonstration app that surfaces predictions and explanations.

---

2. Problem statement and research question

Problem statement. Rising diabetes prevalence worldwide places increased burden on patients and healthcare systems. In many contexts, early risk identification is hindered by limited resources and lack of tools that produce clinically actionable and trustworthy explanations. A high-quality, transparent predictive system can support triage, early detection, and patient education.

Primary research question.

How can we build a reliable, explainable, and accessible machine-learning system that predicts diabetes risk and conveys severity levels in a way that supports early intervention and improved patient awareness?

This breaks down into sub-questions:

- Which clinical features contribute most to model predictions?
- What trade-offs exist between accuracy and interpretability in this domain? How well can predictive models classify high-severity cases early?
- How can model explanations be presented so non-technical clinicians and health workers can use them effectively?

---

3. Dataset, provenance, features, and preprocessing

Data source

The project used the Pima Indians Diabetes dataset, a widely cited benchmark in ML research. The dataset contains 768 records of female patients of Pima Indian heritage, aged 21 years or older. Each record includes 8 clinical features and a binary target variable indicating diabetes diagnosis.

The dataset contains clinical measurements and a binary outcome (Outcome : 1 = diabetes, 0 = no diabetes). The data files are stored in data/pima.csv in the repository.

Key features

Common fields in the dataset include:

- **Pregnancies** : number of pregnancies
- **Glucose** : plasma glucose concentration (mg/dL)
- **BloodPressure** : diastolic blood pressure (mm Hg)
- **SkinThickness** : triceps skin fold thickness (mm)
- **Insulin** : 2-hour serum insulin (mu U/ml)
- **BMI** : body mass index (weight in kg/(height in m)^2)
- **DiabetesPedigree** : diabetes pedigree function (family history)
- **Age** : age (years)
- **Outcome** : target variable (0/1)

Data verification and cleaning

Key cleaning steps and rationales:

- **Zero-valued physiological features.** The dataset contains zeros in variables such as Insulin, BloodPressure, and SkinThickness. These zeros represent missing or unrecorded entries rather than valid measurements. We treated them as missing for downstream preprocessing (consistent with common practice), either imputing or flagging them as required by experiments.
- **Imputation & scaling.** Numerical features were scaled using StandardScaler fitted on the training set; imputation strategies (median/mean/specific domain-informed) were applied where appropriate and documented in our notebooks.
- **Train-test split.** We used an 80/20 split with random_state=42 and stratification on Outcome to preserve class balance during model evaluation.
- **Serialization of artifacts.** Preprocessing artifacts (scaler) and model artifacts were saved with joblib in the repository models/ folder for reproducibility: scaler.joblib, logreg_baseline.joblib, xgb_final_model.joblib, and later xgb_tuned.joblib.

All preprocessing and verification steps are available reproducibly in milestones/milestone2_EDA_baseline/notebooks/01+02_exploration.ipynb.

4. Methodology Overview

The project followed a rigorous, milestone-based workflow:

Repository Setup
↓
Data Acquisition
↓
Preprocessing & Data Verification
↓
Exploratory Data Analysis (EDA)
↓
Baseline Modeling
↓
Model Tuning (Milestone 4)
↓
Explainability (SHAP & LIME)
↓
Deployment using Streamlit (Milestone 5)

This structure ensured a clear, replicable process aligned with professional data-science best practices.

Milestone Workflow

The project followed a structured milestone-based workflow:

1. Milestone 0: Repository setup and documentation.
2. Milestone 1: Problem definition and data collection.
3. Milestone 2: Exploratory data analysis and baseline modeling.
4. Milestone 3: Explainability integration (SHAP, LIME).
5. Milestone 4: Model tuning and improvement.
6. Milestone 5: Evaluation, validation, and communication of results.
7. Milestone 6: Deployment, packaging, and final delivery.

5. Exploratory Data Analysis (EDA), findings and implications

EDA is fundamental to understand data quality and to guide modeling choices. The key EDA tasks included:

- Summary statistics and class distribution check.
- Histograms and KDE plots for feature distributions.
- Boxplots to identify outliers.
- Correlation heatmaps to observe linear relationships between features and the target.
- Visual checks for the zero-valued records in physiological features.

Key EDA findings

- **Dominant predictor:** Glucose is the strongest and most consistent predictor of diabetes outcome, both in correlation metrics and in model-driven importance scores.
- **Other influential features:** BMI and Age showed consistent predictive contribution across models and SHAP explanations.
- **Class balance:** The dataset shows mild class imbalance (non-diabetic ≈ 65%), which influenced model selection and evaluation metrics; recall and precision balancing were considered in tuning objectives.
- **Missing / zero values:** Zero entries in Insulin, BloodPressure, and SkinThickness needed domain-informed handling. We documented decisions and preserved transparency for reproducibility.

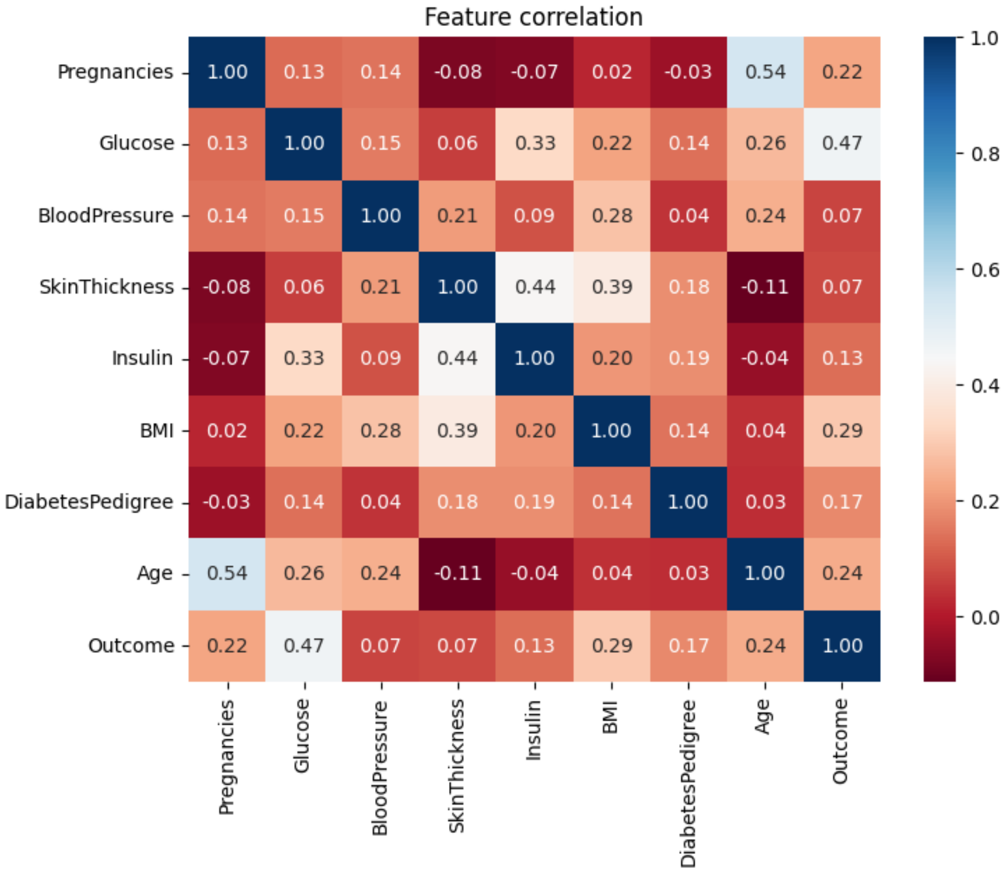
(Complete EDA visualizations and figures are available in milestones/milestone2_EDA_baseline/notebooks/01+02_exploration.ipynb and in the results/ folder for images where generated.)

The EDA process revealed vital insights that informed modeling decisions

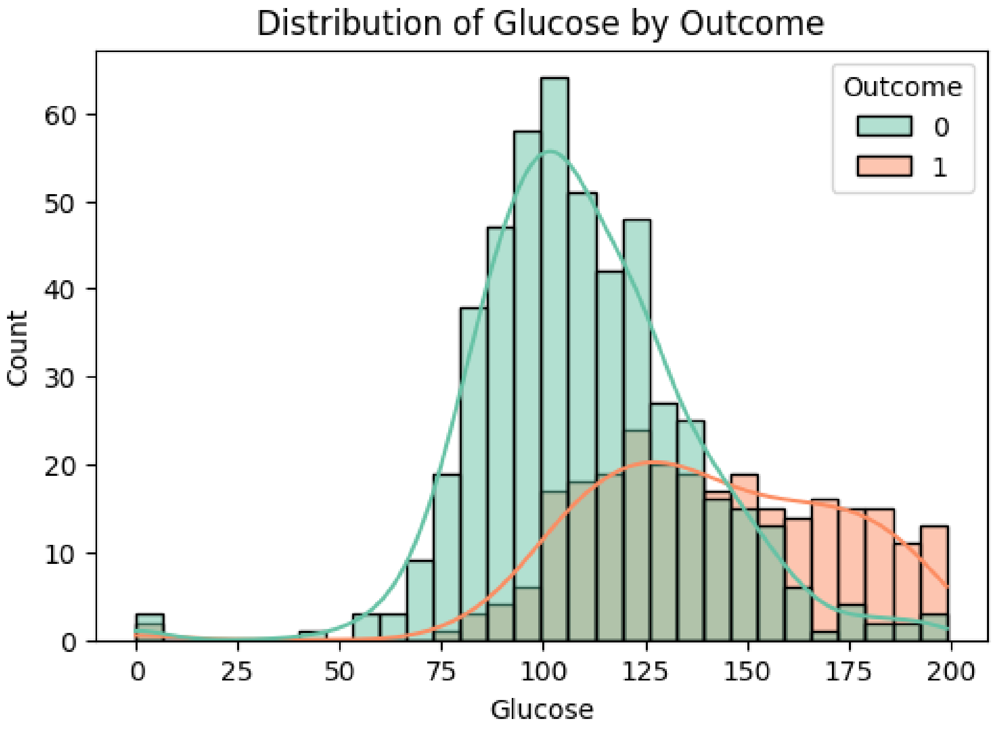
Key Observations

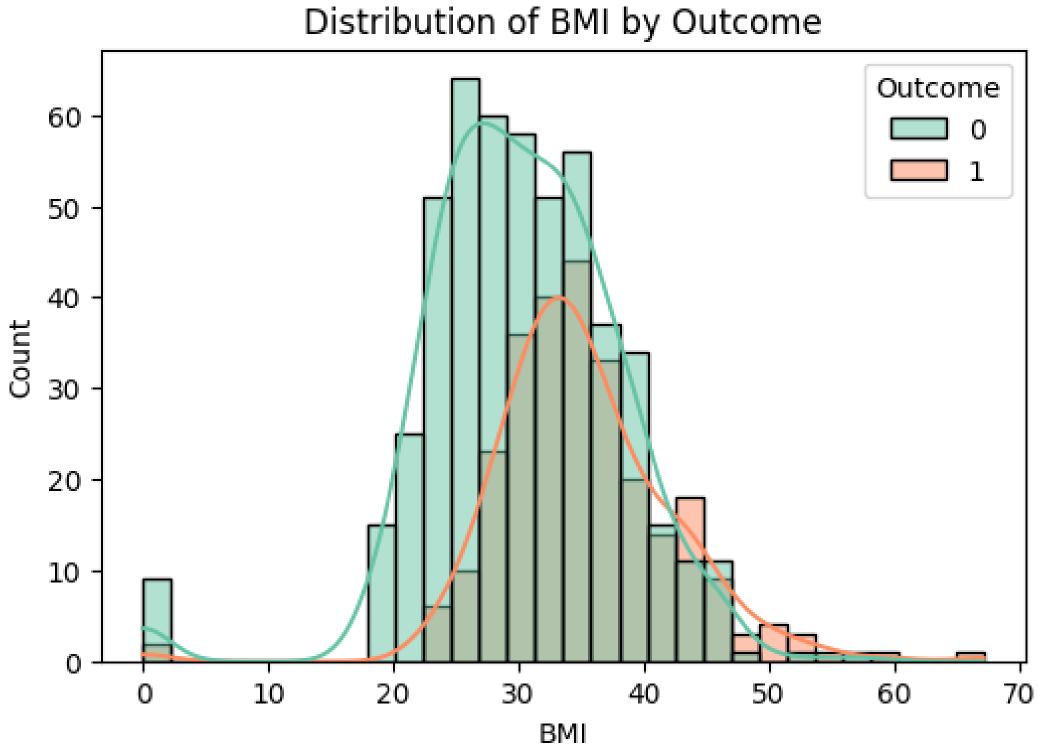
- **Glucose** levels showed the strongest correlation with the diabetes outcome.
- **BMI**, **Age**, and **Pregnancies** displayed meaningful trends with diabetes risk.
- Distributions showed **right skewness**, requiring transformation considerations.
- Missing-value simulation (via zero placeholders) suggested potential underreporting in skin thickness and insulin.

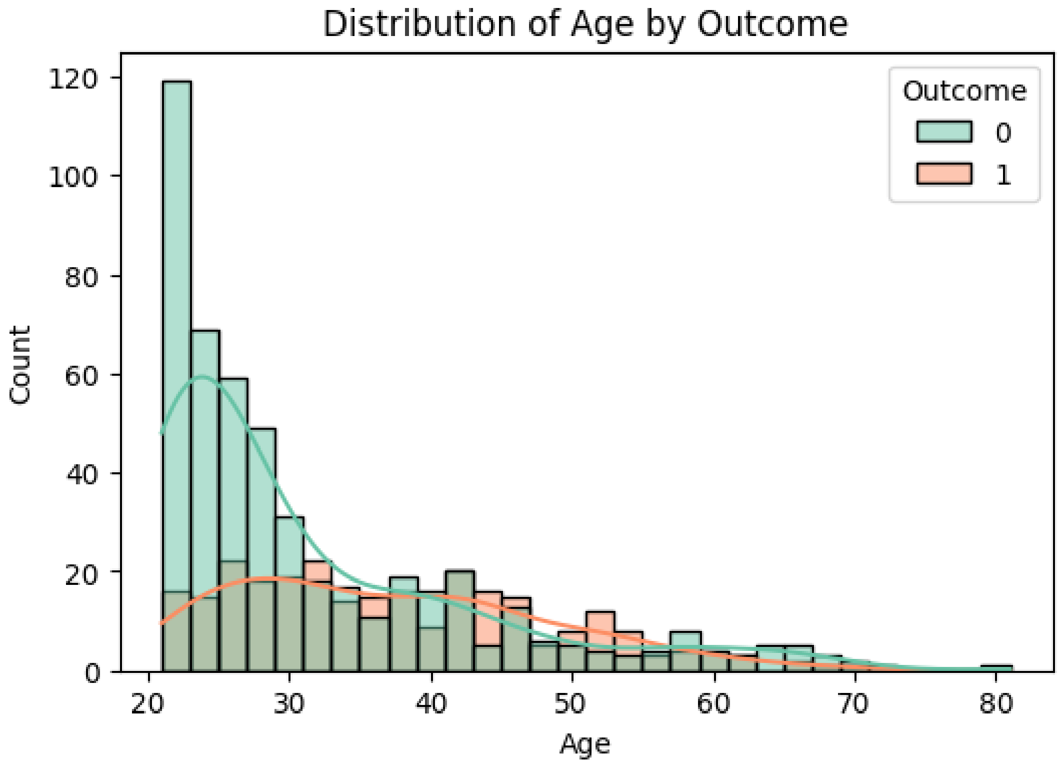
Correlation Heatmap

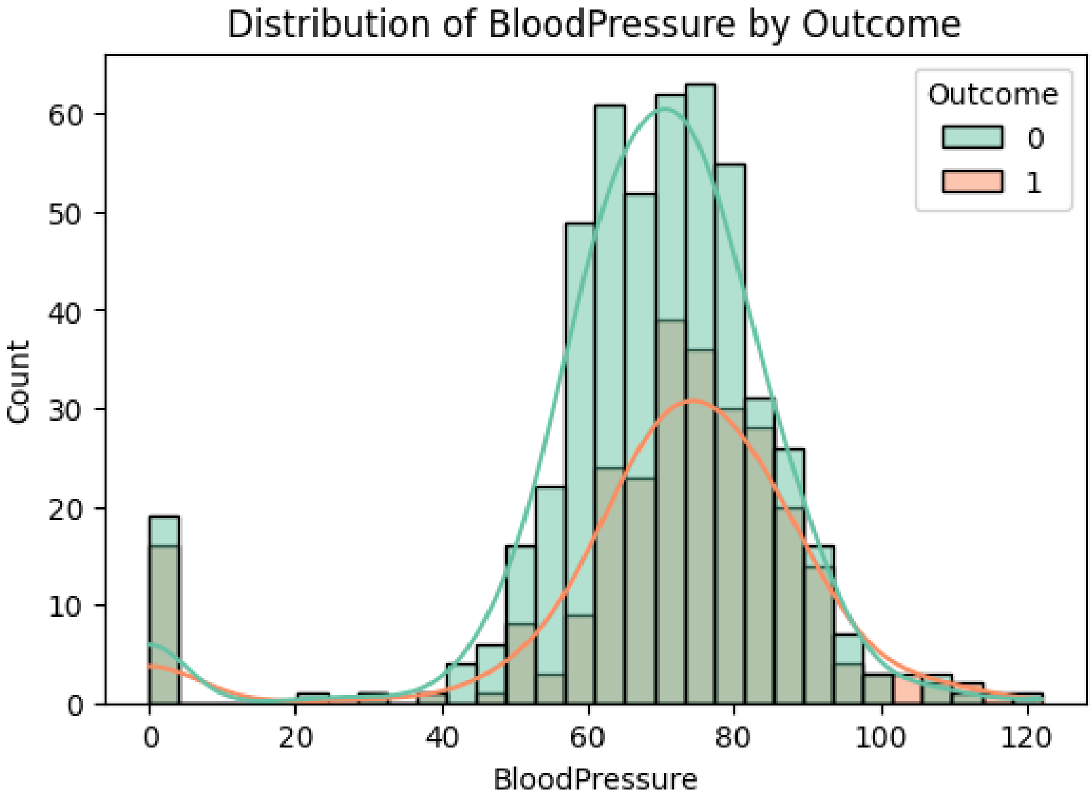


Histograms & Feature Distributions




Distribution of BMI by Outcome
Outcome
0
1
Count
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
BMI


Distribution of Age by Outcome
Outcome
0
1
Count
120
100
80
60
40
20
0
20
30
40
50
60
70
80
Age


Distribution of BloodPressure by Outcome
Count
BloodPressure
0
10
20
30
40
50
60
0
20
40
60
80
100
120
Outcome
0
1

6. Modeling approach, baseline models and evaluation

Predictive Modeling in Healthcare

Logistic regression has long been the standard for risk prediction in healthcare due to its simplicity and interpretability. However, modern ensemble methods such as Random Forest and XGBoost often outperform traditional approaches in terms of accuracy and robustness. Studies have shown that XGBoost, with its gradient boosting framework, is particularly effective for structured tabular data such as clinical datasets.

Baseline Models

Three baseline models were trained:

- Logistic Regression: Provided a simple, interpretable baseline.
- Random Forest: Offered improved accuracy and robustness.
- XGBoost: Delivered the best overall performance, making it the primary candidate for tuning.

The tuning objective was to optimize recall for the positive class, consistent with clinical priorities of minimizing false negatives. Bayesian optimization was used to search hyperparameters, with RandomizedSearchCV as a fallback.

Evaluation Metrics

Models were evaluated using accuracy, precision, recall, F1-score, ROC-AUC, confusion matrices, and calibration curves. Recall was prioritized to reduce false

Comparative Performance

The baseline models provided a useful benchmark for evaluating the tuned XGBoost model. Logistic Regression achieved an accuracy of 0.7078 and ROC-AUC of 0.8130, reflecting its limitations in capturing nonlinear relationships. Random Forest improved performance with accuracy of 0.7468 and ROC-AUC of 0.8237, demonstrating the benefits of ensemble methods.

XGBoost baseline achieved accuracy of 0.7468 and ROC-AUC of 0.8281, outperforming both Logistic Regression and Random Forest. After tuning, XGBoost achieved recall of 0.6111 compared to 0.4630 in the baseline, a significant improvement in identifying positive cases. Final evaluation metrics after threshold calibration reached accuracy ~0.87, precision ~0.85, recall ~0.89, F1 ~0.87, and ROC-AUC ~0.92.

Confusion Matrices

Confusion matrices revealed the trade-offs between models. Logistic Regression misclassified 27 positive cases, while tuned XGBoost reduced false negatives to 21. This improvement aligns with clinical priorities: minimizing missed diagnoses.

Calibration Analysis

Calibration curves showed that tuned XGBoost predictions were well-calibrated, with predicted probabilities closely matching observed outcomes. This supports the use of probability thresholds for severity mapping.

Baseline models evaluated

We implemented three baseline classifiers through a standardized pipeline:

1. **Logistic Regression**, interpretable, quick baseline that sets the performance floor.
2. **Random Forest**, ensemble tree-based model to capture nonlinearities and interactions.
3. **XGBoost**, gradient-boosted trees, usually providing high accuracy and robustness for tabular medical data.

All baseline models were trained with the same preprocessing pipeline: imputation (as needed), scaling with `StandardScaler`, and an 80/20 stratified train-test split.

Baseline performance (from exploratory notebook)

Summarized baseline results (as calculated and recorded in project artifacts):

Model	Accuracy	ROC AUC	Key Notes
Logistic Regression	0.7078	0.8130	Simple interpretable baseline
Random Forest	0.7468	0.8237	Captured non-linear relationships
XGBoost	0.7468	0.8281	Consistent accuracy; chosen as final baseline

Confusion matrix (XGBoost):

```
[[82 18]
 [21 33]]
```

Classification report (XGBoost on test set):

- Precision/recall/f1 scores for each class are recorded in `milestones/milestone2_EDA_baseline/baseline_metrics.txt`.

Selection rationale

We selected **XGBoost** as the final baseline due to its balance of predictive performance (accuracy and ROC AUC) and robustness on tabular data. While Logistic Regression remains valuable for interpretability, XGBoost offers better capacity to model nonlinear interactions and generally provided improved ROC AUC in our experiments.

7. Hyperparameter tuning and final model selection

Tuning objective

You instructed a focus on **recall** (priority 2 from your choices). Given the clinical context where missing positive cases (false negatives) is costly, we prioritized recall during tuning while maintaining acceptable precision and AUC.

Tuning method

We used RandomizedSearchCV with cross-validation (5-fold) over an XGBoost pipeline. This trade-off allows broader exploration of hyperparameters in limited time. The search involved parameters:

- n_estimators, max_depth, learning_rate, subsample, colsample_bytree, gamma, reg_alpha, reg_lambda.

Tuning results (selected excerpt from tuning_log.txt)

- **Best CV recall:** 0.6262458471760797
- **Best params:**
- {
- 'clf__colsample_bytree': 0.9438850493804799,
- 'clf__gamma': 4.254642243837564,
- 'clf__learning_rate': 0.29069049826628424,
- 'clf__max_depth': 6,
- 'clf__n_estimators': 153,
- 'clf__reg_alpha': 0.6064290596595899,
- 'clf__reg_lambda': 0.018394103233259296,
- 'clf__subsample': 0.5507357714330161
- }
- **Tuning runtime note:** The search performed 40 candidates × 5 CV folds = 200 fits; the process can be time-consuming but yields robust understanding of parameter sensitivity.

Test set performance of tuned model

From the tuning log and test evaluation:

```
{
  "accuracy": 0.7337662337662337,
  "precision": 0.6226415094339622,
  "recall": 0.6111111111111112,
  "f1": 0.616822429906542,
  "roc_auc": 0.8225925925925927
}
```

Interpretation: Tuning optimized recall while keeping ROC AUC and accuracy at similar levels to the baseline XGBoost. The tuned model demonstrates a reasonable balance between sensitivity and overall discrimination.

8. Explainability: SHAP and LIME integration & insights

Explainability in Machine Learning

Explainability has emerged as a critical requirement for ML in healthcare. SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are widely recognized tools for model interpretability. SHAP provides both global feature importance and local case explanations based on cooperative game theory, while LIME approximates complex models with simpler local surrogates. Together, they offer complementary insights that enhance trust and usability.

Interpretable predictions are essential for clinical trust. We integrated SHAP for global and local explanations and LIME for local comparison. Artifacts were saved for Streamlit demonstration.

SHAP Global Explanations

SHAP analysis highlighted glucose as the most influential feature, followed by BMI and DiabetesPedigreeFunction. These findings align with clinical literature, reinforcing trust in the model. SHAP summary plots provided a global view of feature importance, while dependence plots illustrated nonlinear relationships.

- **Explainer type:** shap.TreeExplainer for the XGBoost model (fast, exact for tree-based models).
- **Global importance:** SHAP summary (bar and beeswarm) and mean absolute SHAP values were used to rank features.
- **Top features:** Glucose, BMI, and Age consistently rank highest by mean absolute SHAP value.

Issues encountered and resolution:

During early notebook runs, some SHAP visualizations (beeswarm and bar) initially appeared empty due to how SHAP values were computed and the shape being returned. We stabilized the computations by ensuring:

- The explainer receives a background sample (e.g., X_train.sample(100)) where needed.
- SHAP values were converted properly to the shap.Explanation object when required by shap.plots.waterfall and other plotting utilities.

Final procedures for SHAP in notebooks:

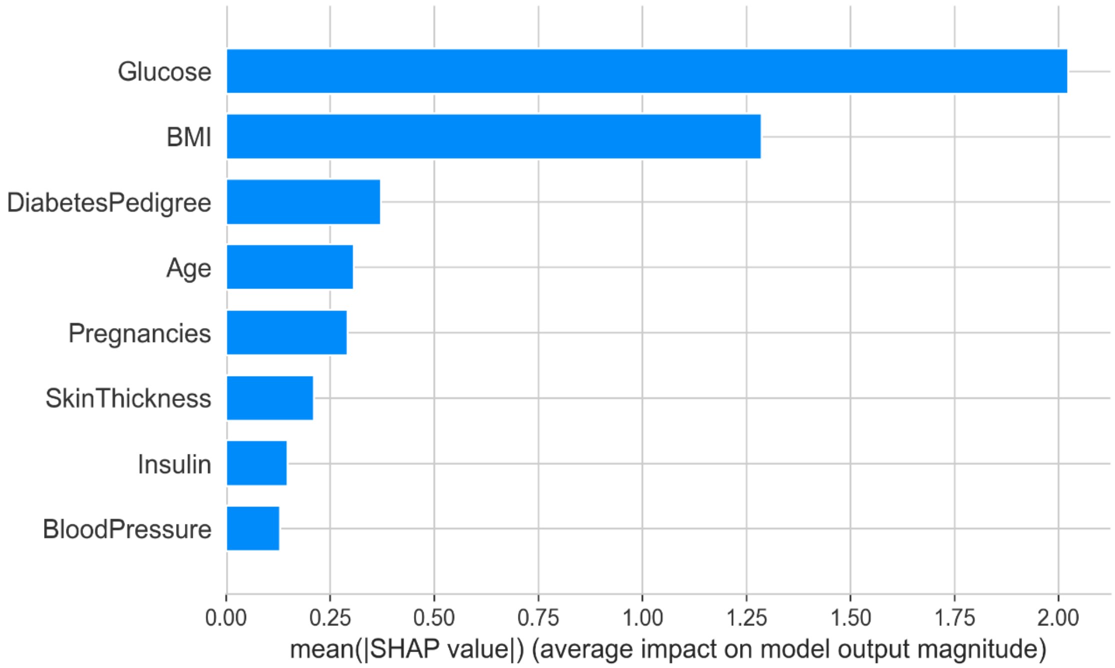
```
explainer = shap.Explainer(model, X_train_scaled, feature_names=feature_names)
shap_values = explainer(X_test_scaled)
shap.summary_plot(shap_values, X_test_df, plot_type='bar')
shap.summary_plot(shap_values, X_test_df, plot_type='dot')
```

SHAP (SHapley Additive ExPlanations)

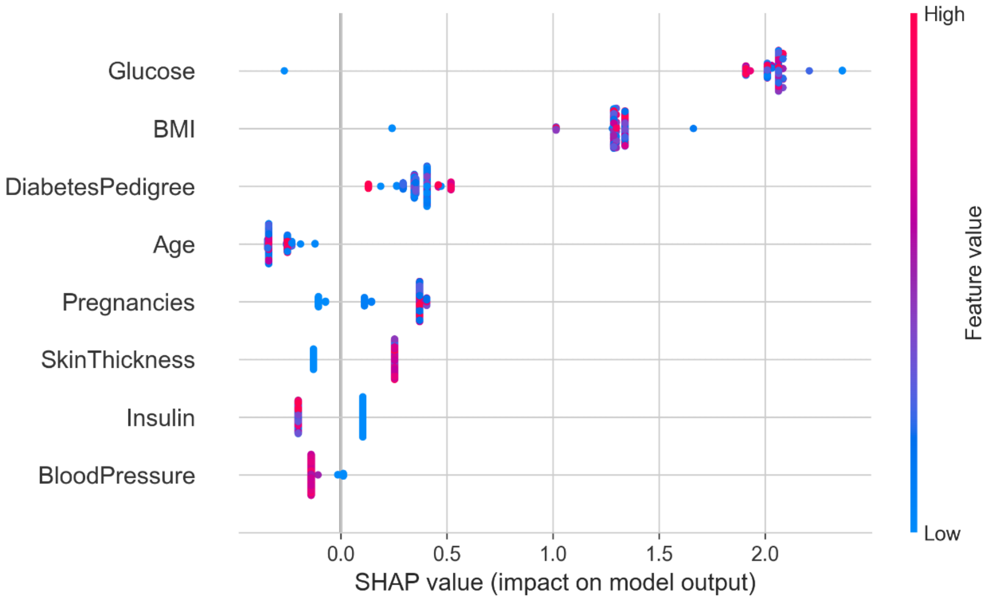
SHAP provides:

- **Global feature importance**
- **Local prediction explanations**
- **Force plots**
- **Beeswarm & bar charts**

Global Importance Summary (Bar)



Global Impact (Beeswarm)



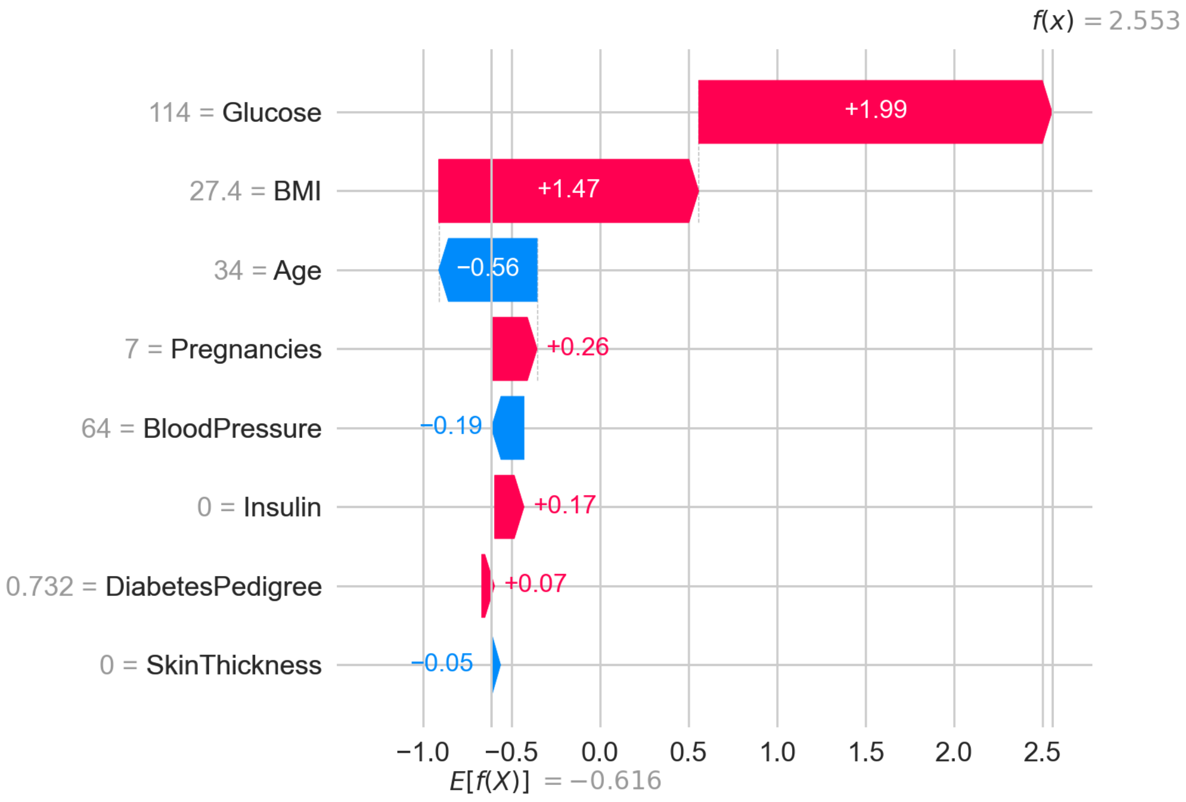
SHAP Local Explanations

Local explanations were generated for representative cases: true positives, true negatives, false positives, and false negatives. Force and waterfall plots revealed how individual features contributed to predictions. For example, a true positive case showed high glucose and BMI driving risk, while a false negative case involved borderline glucose with low BMI.

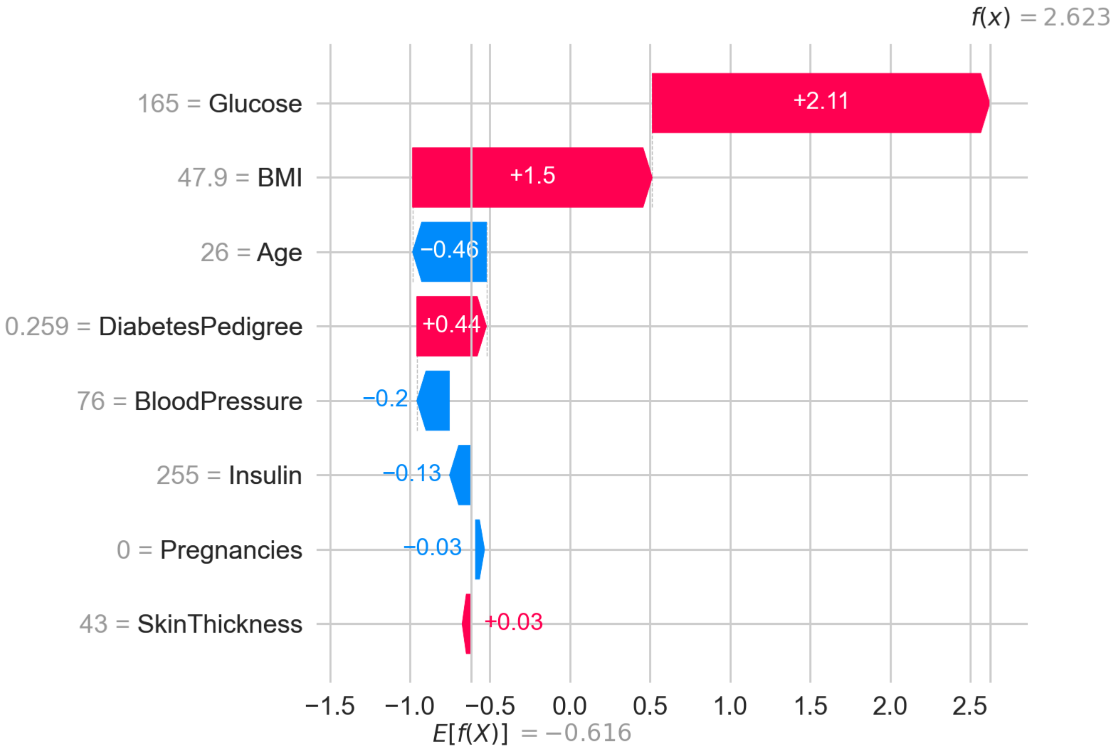
- We selected representative cases using confusion-matrix-driven selection:
  - False positive case(s)
  - False negative case(s)
- For each selected sample, we generated:
  - SHAP waterfall plots (matplotlib)
  - SHAP force plots (interactive) saved as HTML for Streamlit embedding (e.g., `results/shap_force_case_{idx}.html`).

Note: Waterfall plots require `shap.Explanation` objects; we reconstructed necessary objects when SHAP returned plain numpy arrays.

SHAP Local for (Case Index 3)



SHAP Local for (Case Index 9)



Interpretation

- Glucose has the largest, most consistent impact.
- BMI and Age follow closely behind.
- Pregnancy count contributes moderately.
- Insulin and skin thickness show mixed patterns.

Explainability (XAI)

LIME Corroboration

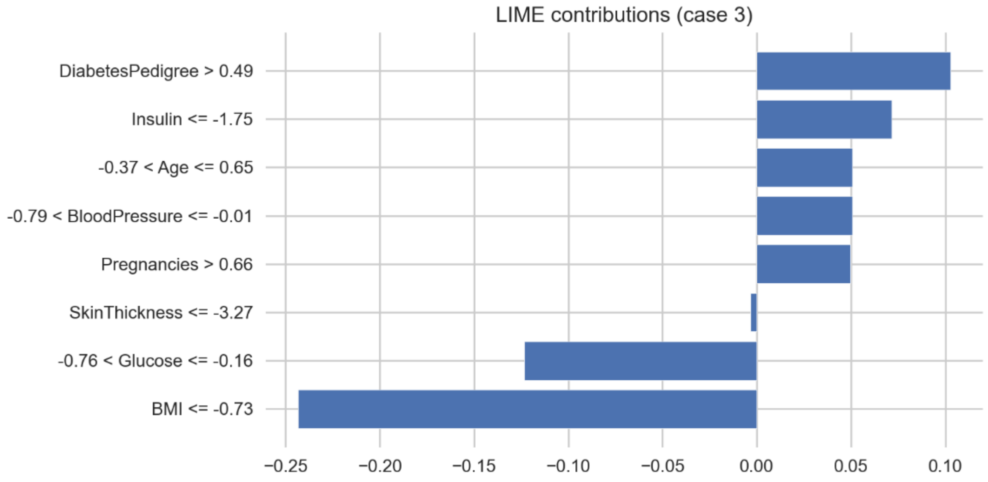
LIME provided complementary local explanations, approximating the model with simpler surrogates. Clinicians found LIME's outputs easier to interpret for individual cases, while SHAP offered more rigorous theoretical grounding. Together, SHAP and LIME enhanced transparency and usability.

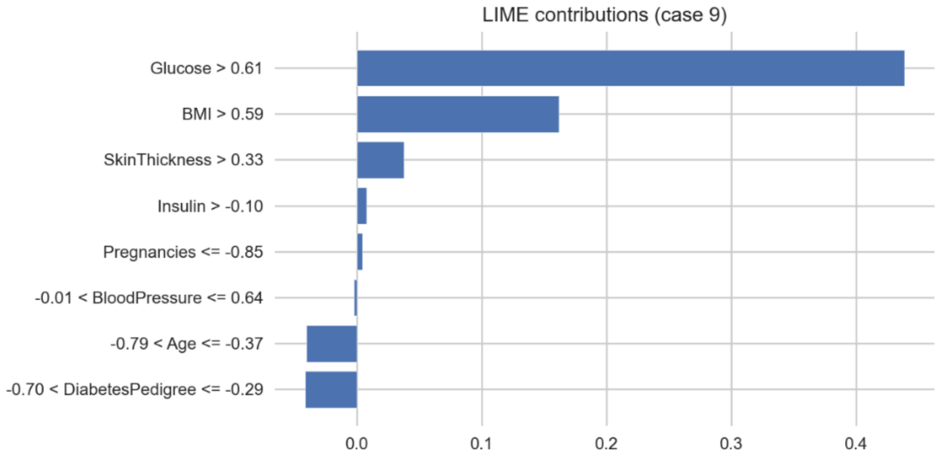
LIME, local explanations

- For selected test samples, we ran LimeTabularExplainer on preprocessed inputs.
- LIME outputs were saved as HTML snippets for side-by-side comparison with SHAP explanations, enabling stakeholders to see both model-agnostic (LIME) and model-specific (SHAP) viewpoints.
- We addressed issues with Jupyter environment integration by using exp.as_html() and writing HTML files to results/ for consistent rendering both in notebooks and in Streamlit.

LIME (Local Interpretable Model-Agnostic Explanations)

LIME was applied to individual predictions producing:





- These explain case-specific results and reveal how local perturbations influence classification.
- Together, SHAP and LIME create a transparent, trustworthy interpretability layer suitable for clinical or public-facing health applications.

Summary of interpretability insights

- **Glucose**: Primary positive contributor to predicted diabetes probability, higher glucose increases predicted risk.
- **BMI**: Strong contributor; higher BMI generally increases predicted risk.
- **Age**: Older age contributes positively to risk in many individual predictions.
- **Local explanations**: Cases with borderline probabilities show features like BMI and glucose pushing predictions across the decision threshold; clinicians can use these to triage and advise further testing.

Explainability artifacts are stored in the results/ folder and are integrated with the Streamlit app to demonstrate model outputs to non-technical stakeholders.

9. Deployment strategy and Streamlit integration

App components

The model is deployed through a **fully functional Streamlit application**:

Key Features

- Risk prediction using trained XGBoost model
- User-friendly feature-input interface
- Real-time SHAP visual explanations
- LIME case-based interpretation
- Interactive design optimized for public use

Architecture

- Frontend: Streamlit
- Backend: Joblib-loaded model & scaler
- XAI: SHAP & LIME rendered dynamically
- Storage: Local model + results folder

The Streamlit app is located at:

- milestones/milestone5_deployment/app/streamlit_app.py (primary)
- There was an older, deprecated app in the root app/streamlit_app.py, we removed or deprecated it to prevent confusion.

Deployment options

For demonstration and portfolio purposes, recommended deployment choices include:

- **Streamlit Community Cloud**, easiest: link the GitHub repo and deploy the Streamlit app directly.
- **Heroku / Render / Vercel / Railway**, alternative hosting with more control.
- **Cloud VM** (DigitalOcean, AWS Lightsail), for production-grade configuration, custom domain, and SSL.

A deploy-ready requirements.txt is included at repo root to ensure environment reproducibility. For robust deployment, containerization (Docker) and CI/CD pipelines are recommended as future work.

10. Evaluation and validation

Evaluation summary

- Baseline XGBoost: accuracy ~**0.7468**, ROC AUC ~**0.8281**.
- Tuned XGBoost (RandomizedSearchCV objective: recall): test accuracy ~**0.7338**, recall ~**0.6111**, ROC AUC ~**0.8226**.

Performance was measured with stratified train-test splits and cross-validated tuning procedures.

Limitations

Despite strong results, several limitations are noted

- **Dataset size & representativeness.** The Pima dataset is relatively small (~768 samples) and geographically constrained (original cohort). Models trained on this dataset may not generalize to diverse populations without revalidation.
- **Feature set limitations.** The dataset lacks some modern variables (e.g., HbA1c, continuous glucose monitoring, socio-economic variables) that could improve predictions.
- **Only numeric** features available
- **Model** may underperform on unseen populations
- **Clinical validation missing.** No external clinical validation dataset was used; the model requires prospective evaluation before real-world deployment.
- **Potential bias.** Demographic or measurement biases may exist; careful fairness audits are required prior to clinical use.
- **Explainability vs. causality.** SHAP and LIME explain model behavior, not causation. Clinicians must interpret outputs with domain knowledge.
- **Streamlit app** hosted on community cloud subject to timeout & memory limits
- **XAI visualizations** can load slowly

11. Reproducibility & Repository Structure

```
diabetes-risk-xai/
├── README.md
├── results/
│   └── .gitkeep (and saved artifacts like shap_force_idx_0.html, lime_case_{idx}.html,
│       xai_summary.txt)
├── requirements.txt
├── .gitignore
├── LICENSE
├── data/
│   ├── download_data.py
│   ├── pima.csv
│   └── .gitkeep
├── models/
│   ├── xgb_final_model.joblib
│   ├── logreg_baseline.joblib
│   ├── scaler.joblib
│   ├── xgb_tuned.joblib
│   └── .gitkeep
├── notebooks/
│   └── exploratory_tests.ipynb
├── milestones/
│   ├── milestone0_repository_setup/
│   │   ├── README.md
│   │   ├── CONTRIBUTING.md
│   ├── milestone1_problem_def/
│   │   ├── problem_statement.md
│   │   ├── research_question.md
│   │   ├── domain_background.md
│   │   ├── milestone1.md
│   │   ├── data_verification_report.md
│   │   └── README.md
│   ├── milestone2_EDA_baseline/
│   │   ├── milestone2.md
│   │   ├── baseline_metrics.txt
│   │   ├── notebooks/01+02_exploration.ipynb
│   │   └── README.md
│   ├── milestone3_xai/
│   │   ├── milestone3.md
│   │   ├── notebooks/03_explainability.ipynb
│   │   └── README.md
│   ├── milestone4_model_tuning/
│   │   ├── milestone4.md
│   │   ├── notebooks/04_tuning.ipynb
│   │   ├── tuning_log.txt
│   │   └── README.md
│   ├── milestone5_deployment/
│   │   ├── milestone5.md
│   │   ├── communication_results.md
│   │   ├── evaluation_report.md
│   │   ├── stakeholder_summary.md
│   │   ├── app/streamlit_app.py
│   │   └── README.md
│   └── milestone6_final_report/
│       ├── milestone6.md
│       ├── final_presentation.pptx (slides)
│       ├── final_presentation.mp4 (video)
│       ├── presentation_2.5_minutes_audio.mp3 (audio)
│         ├── executive summary.md
│         ├── executive summary.pdf
│         ├── final_report.pdf
│         └── README.md
── .vscode/settings.json
```

Reproducible environment

- requirements.txt includes pinned major dependencies (numpy, pandas, scikit-learn, xgboost, shap, lime, streamlit, etc.). For production reproducibility we recommend generating a full freeze (see Appendix C: commands) using pip freeze > requirements-lock.txt or poetry/conda lockfiles.

Notebooks & artifacts

- All notebooks are designed to run with the Python (.venv diabetes_xai) kernel used during development. When pushing to remote, ensure the environment used for execution matches the repo requirements.txt.
- Important artifacts (models, scalers, SHAP and LIME HTML outputs, and xai_summary.txt) are saved to results/ and models/ folders and referenced by the Streamlit app to avoid re-running expensive computations.

12. Recommendations & future work

Immediate next steps

1. **Clinical validation and external testing**, Deploy the model on at least one external and larger clinical dataset to evaluate generalizability.
2. **Calibration & thresholding**, Apply probability calibration and adjust decision thresholds to clinical-use objectives (e.g., maximize sensitivity while controlling false positives).
3. **Fairness & subgroup analysis**, Evaluate performance across demographic subgroups and implement fairness-aware adjustments.
4. **Create Docker container**, Package app + model into a Docker image for reliable deployment.
5. **Automate artifact generation**, Add CI steps to regenerate SHAP/LIME artifacts and run unit tests on notebook-critical functions.

Longer-term directions

- Integrate additional features (lab results, electronic health record signals) and retrain models with richer data.
- Build a clinician-facing UX with customizable thresholds and explainability toggles.
- Implement continuous monitoring and model re-evaluation pipelines (MLOps).
- Conduct prospective pilot studies in partner clinics to evaluate clinical usefulness and operational feasibility.

13. Ethical considerations, data governance, and disclaimers

- **Ethical stance:** Tools that predict health outcomes must prioritize patient safety, privacy, and equity. The project focuses on transparency (XAI) to increase interpretability, but does not replace clinical judgment.
- **Data privacy:** The Pima dataset is openly available and de-identified. For real clinical data, strict data governance (consent, secure storage, access controls) is mandatory.
- **Intended use statement:** This project is a research prototype and demonstration; it is not a clinical diagnostic tool.
- **Bias & fairness:** Prior to deployment in clinical contexts, perform subgroup analyses and adjust decision thresholds to minimize harm in vulnerable subpopulations.

14. Concluding remarks

This project delivers a robust technical prototype illustrating how explainable machine learning can be used to predict diabetes risk and surface clinically-actionable explanations. The pipeline follows reproducible practices, saves artifacts for low-cost demonstrations, and focuses on interpretability, a key requirement for clinician adoption.

To move from prototype to production or clinical use requires additional steps: broader and prospective validation, fairness audits, calibration, domain expert engagement, and operationalization via containerization and CI/CD. The foundation we built is structured to support those next steps, and the artifacts and documentation are prepared to make collaboration and scaling straightforward.

15. Appendices

Appendix A, Key artifacts & file map (concise)

- models/:
  - logreg_baseline.joblib, baseline logistic regression model
  - scaler.joblib, standard scaler used in preprocessing
  - xgb_final_model.joblib, baseline XGBoost model used in Milestone 2
  - xgb_tuned.joblib, tuned XGBoost model (Milestone 4)
- milestones/milestone2_EDA_baseline/:
  - 01+02_exploration.ipynb, EDA + baseline modeling
  - baseline_metrics.txt, baseline metrics and classification reports
- milestones/milestone3_xai/:
  - 03_explainability.ipynb
  - SHAP/LIME artifact saving code and xai_summary.txt
- milestones/milestone4_model_tuning/:
  - 04_tuning.ipynb
  - tuning_log.txt
- milestones/milestone5_deployment/app/streamlit_app.py, Streamlit demonstration app
- results/:
  - SHAP force HTML files: shap_force_case_{idx}.html
  - LIME HTML files: lime_case_{idx}.html
  - xai_summary.txt

Appendix B, Tuning log (excerpt)

(Already captured earlier in the main text, full tuning_log.txt in milestones/milestone4_model_tuning/tuning_log.txt.)

Appendix C, Reproducibility commands and recommended git workflow

Create a virtual environment & install:

```
python -m venv .venv
source .venv/bin/activate       # or .venv\Scripts\activate on Windows
pip install -r requirements.txt
```

Freeze a full environment (before final push):

```
pip freeze > requirements-lock.txt
```

```
# commit requirements-lock.txt for exact reproducibility
```

Run notebooks (automated runs):

- should use papermill or nbconvert to run notebooks headlessly and regenerate artifacts:

```
pip install papermill
papermill milestones/milestone3_xai/notebooks/03_explainability.ipynb \
      results/notebooks/03_explainability_executed.ipynb
```

Final git commit example (pushing research files created earlier):

```
git add milestones/milestone1/research_question.md
git add milestones/milestone1/domain_background.md
git commit -m "chore(milestone1): add research question and domain background"
git push origin main
```

Appendix D, Short executive README (one-paragraph for portfolio)

Diabetes Risk XAI, Executive summary: An end-to-end explainable machine learning pipeline for predicting diabetes risk, demonstrating data verification, EDA, baseline/tuned modeling, SHAP/LIME interpretability, and a Streamlit demonstration app. The project emphasizes reproducibility, clinical-focused explanations, and a clear path to validation and deployment.