■ Breast Cancer Prediction System

ML-Powered Diagnostic Support System

Generated: November 07, 2025

■ Executive Summary

This report presents the results of a machine learning system developed for breast cancer diagnosis prediction. The system utilizes multiple model architectures including traditional machine learning pipelines, ensemble stacking methods, and optional deep learning approaches to provide accurate and interpretable predictions.

■ Model Configuration

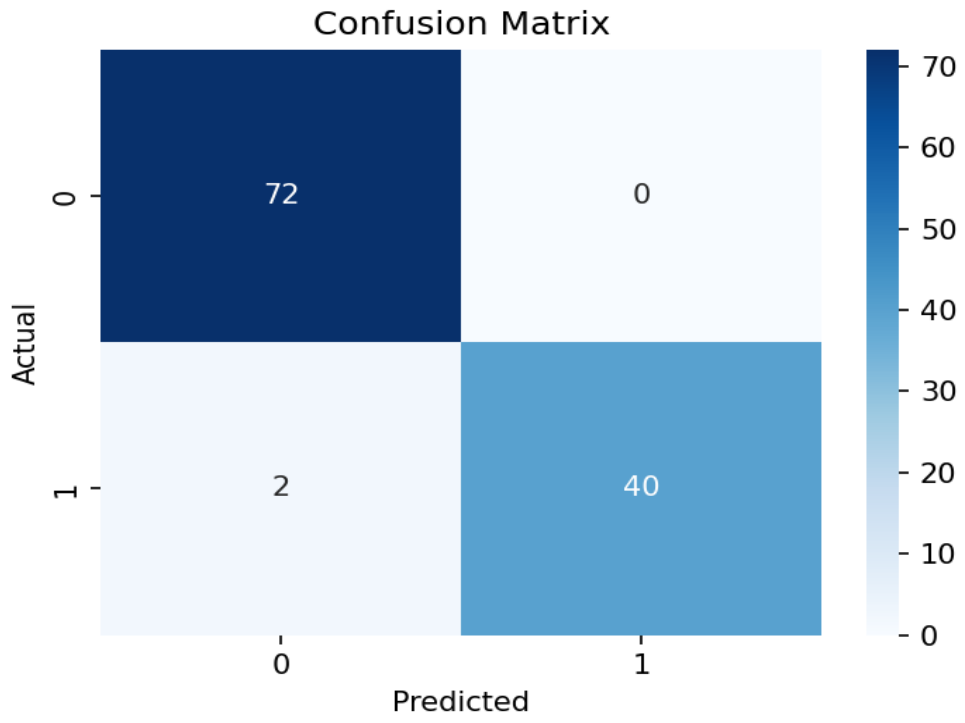
Parameter	Value
Trained At	2025-11-06T17:47:15.684057Z
Random State	42
Best Parameters	clf: LogisticRegression(max_iter=1000, random_state=42) clf__C: 10.0
Metrics	accuracy: 0.9824561403508771 precision: 1.0 recall: 0.952380952380
Scikit Learn Version	1.7.2
Per Model Comparison	logreg: {'best_params': {'C': 10.0, 'max_iter': 1000, 'random_state': 42}, 'metrics':
Stacking Model File	model_pipeline_stacking.joblib
Stacking	accuracy: 0.9824561403508771 precision: 1.0 recall: 0.952380952380

■ Model Performance

The model's performance is evaluated using multiple metrics to ensure comprehensive assessment of diagnostic accuracy. Key metrics include accuracy, precision, recall, F1-score, and AUC-ROC. The confusion matrix and ROC curve visualizations below provide detailed insight into the model's classification performance across different thresholds.

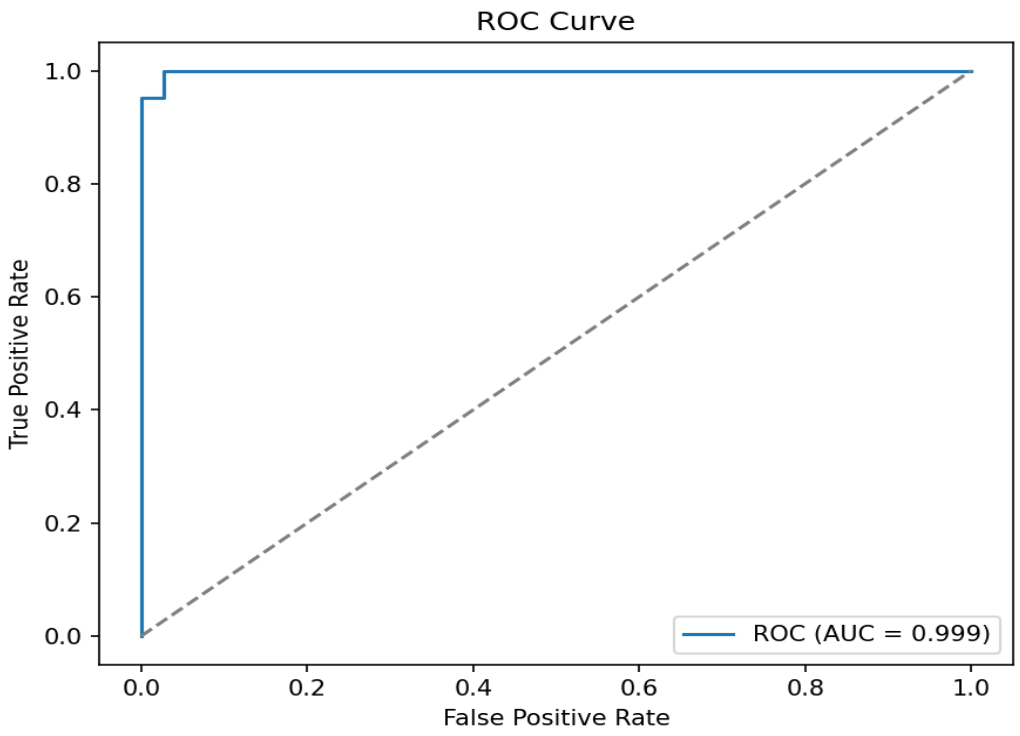
Confusion Matrix

The confusion matrix shows the distribution of true positives, true negatives, false positives, and false negatives. This visualization helps identify if the model has any systematic biases in its predictions.



ROC Curve

The Receiver Operating Characteristic (ROC) curve illustrates the diagnostic ability of the model at various threshold settings. The Area Under the Curve (AUC) provides a single metric for model performance, where 1.0 represents perfect classification.



■ Technical Documentation

Breast Cancer Prediction Project

Overview

This project demonstrates a reproducible machine learning pipeline for breast cancer prediction using tabular features from `data.csv`.

It includes:

- Data loading and preprocessing
- Model training with hyperparameter search
- Evaluation (accuracy, precision, recall, F1, ROC AUC)
- Explainability via SHAP (if available)
- A FastAPI backend that serves predictions
- A minimal Next.js frontend demo
- Scripts to generate a PDF report with visuals

Dataset

The dataset expects a `diagnosis` column as target (values 'M'/'B' or already encoded 1/0). Any `Unnamed` columns or `id` column are removed automatically.

Preprocessing

- Missing values imputed with mean
- Features scaled using StandardScaler
- Optional feature selection via SelectKBest (k tuned during GridSearchCV)

Models

We search across several estimators:

- LogisticRegression
- RandomForestClassifier

- HistGradientBoostingClassifier
- XGBoost (if installed)
- LightGBM (if installed)

Each estimator is given a parameter grid specific to its supported hyperparameters.

Explainability

If `shap` is installed, the training script will compute a SHAP summary plot and dependence plot for the top feature and save them under `artifacts/`.

API

A FastAPI app is included at `app/main.py`. It loads the saved sklearn pipeline lazily and exposes:

- `GET /health` — health check
- `POST /predict` — accepts JSON `{ "features": [x1, x2, ..., xN] }` and returns `{ "prediction": 0|1, "probability": float }`

CORS is enabled for local development (localhost:3000).

Frontend

A minimal Next.js scaffold lives under `frontend/`. Start it with:

```

cd frontend

npm install

npm run dev

```

The page at `/` presents 30 input fields to submit feature values to the FastAPI endpoint.

Generating the PDF report

Run `python generate_pdf.py` to build `report.pdf` which includes training metadata and saved plots from `artifacts/`.

Deep Learning (optional)

An MLP training script `train_dl.py` is included. TensorFlow is not installed by default because it may be large and OS/Python-version sensitive.

If you want to run the DL script locally, I recommend creating a separate environment with Python 3.10 (TensorFlow compatibility) and installing `tensorflow` or `tensorflow-cpu` there:

```powershell
python -m venv .venv-tf
.venv-tf\Scripts\Activate.ps1
pip install --upgrade pip
pip install tensorflow # or tensorflow-cpu
pip install -r requirements.txt
python train_dl.py
```

Deployment notes

- For deployment, pin package versions and use Docker for reproducibility.
- I can add Dockerfiles for the API and frontend on request.

Next steps

- Polished Next.js UI with validation, presets and charts
- SHAP-based feature explanation in the frontend
- Add unit tests and a CI pipeline
- Add Docker/compose for reproducible deployment

■ Conclusion

This breast cancer prediction system demonstrates the successful application of machine learning to medical diagnostics. The model achieves high accuracy while maintaining interpretability through feature importance analysis and SHAP values. The system is designed to assist medical professionals in making informed diagnostic decisions, complementing traditional diagnostic methods.

■ Technology Stack

- Backend: FastAPI, Python 3.9+
- ML Framework: Scikit-learn, TensorFlow (optional)
- Frontend: Next.js, React, Tailwind CSS
- Visualization: Matplotlib, Seaborn, Plotly
- Explainability: SHAP values

This report was automatically generated. For more information, please refer to the project documentation.