



# Diabetes Progression Prediction Service

ML service for predicting short-term diabetes progression risk to help triage nurses prioritize patient follow-ups.

[Show Image](#)



## Overview

**Context:** A hospital runs a virtual diabetes clinic where nurses review hundreds of patient check-ins weekly to decide who needs follow-up calls. Manual reviews are time-consuming.

**Solution:** This ML service predicts disease progression and returns a continuous risk score, enabling nurses to prioritize high-risk patients first.

**Dataset:** Uses scikit-learn's diabetes dataset as a stand-in for de-identified EHR features (age, BMI, blood pressure, cholesterol, blood sugar, etc.).



## Quick Start

### Using Docker (Recommended)

```
bash

# Pull pre-built image
docker pull ghcr.io/YOUR_USERNAME/YOUR_REPO/diabetes-predictor:latest

# Run service
docker run -p 8000:8000 ghcr.io/YOUR_USERNAME/YOUR_REPO/diabetes-predictor:latest

# Test prediction
curl -X POST http://localhost:8000/predict/single \
  -H "Content-Type: application/json" \
  -d '{
    "age": 0.05, "sex": 0.05, "bmi": 0.06, "bp": 0.02,
    "s1": -0.04, "s2": -0.03, "s3": -0.04, "s4": 0.0,
    "s5": 0.02, "s6": -0.03
  }'
```

## Local Development

```
bash
```

```
# Install dependencies
```

```
pip install -r requirements.txt
```

```
# Train model
```

```
python train.py --version v0.1 --model-type linear
```

```
# Run API
```

```
uvicorn app:app --reload --port 8000
```

```
# Open API docs
```

```
open http://localhost:8000/docs
```



## API Endpoints

### Health Check

```
bash
```

```
GET /health
```

### Response:

```
json
```

```
{  
  "status": "healthy",  
  "model": "LinearRegression",  
  "scaler": "StandardScaler"  
}
```

### Single Patient Prediction

```
bash
```

```
POST /predict/single
```

### Request Body:

```
json
```

```
{
  "age": 0.05,
  "sex": 0.05,
  "bmi": 0.06,
  "bp": 0.02,
  "s1": -0.04,
  "s2": -0.03,
  "s3": -0.04,
  "s4": -0.00,
  "s5": 0.02,
  "s6": -0.03
}
```

## Response:

```
json

{
  "patient_id": 0,
  "progression_score": 152.5,
  "risk_level": "HIGH",
  "high_risk": true
}
```

## Batch Prediction (Triage Dashboard)

bash

POST /predict

## Request Body:

```
json
```

```
{
  "patients": [
    {
      "age": 0.05, "sex": 0.05, "bmi": 0.06, "bp": 0.02,
      "s1": -0.04, "s2": -0.03, "s3": -0.04, "s4": 0.0,
      "s5": 0.02, "s6": -0.03
    },
    {
      "age": -0.01, "sex": -0.04, "bmi": -0.03, "bp": 0.0,
      "s1": 0.01, "s2": 0.02, "s3": 0.03, "s4": 0.02,
      "s5": 0.0, "s6": 0.01
    }
  ]
}
```

**Response:** (sorted by risk, descending)

```
json

{
  "predictions": [
    {
      "patient_id": 0,
      "progression_score": 152.5,
      "risk_level": "HIGH",
      "high_risk": true
    },
    {
      "patient_id": 1,
      "progression_score": 98.3,
      "risk_level": "LOW",
      "high_risk": false
    }
  ],
  "model_version": "v0.1"
}
```







## Architecture





## CI/CD Pipeline

GitHub Actions automatically:

1.  Trains model on every push to `main`
2.  Runs unit tests
3.  Builds Docker image
4.  Pushes to GitHub Container Registry
5.  Runs smoke tests
6.  Reports metrics in Actions summary

## Manual Model Training:

```
bash

# Trigger workflow with custom parameters
gh workflow run train-and-deploy.yml \
  -f version=v0.2 \
  -f model_type=ridge
```

## Model Performance

### v0.1 (Baseline - Linear Regression)

| Metric         | Value      |
|----------------|------------|
| RMSE           | ~55-60     |
| MAE            | ~45-50     |
| R <sup>2</sup> | ~0.45-0.52 |

### Risk Classification (75th percentile threshold):

| Metric    | Value      |
|-----------|------------|
| Precision | ~0.70-0.80 |
| Recall    | ~0.65-0.75 |
| F1 Score  | ~0.68-0.77 |

See [CHANGELOG.md](#) for detailed metrics and version history.

## Training New Models

### Baseline (v0.1)

```
bash

python train.py --version v0.1 --model-type linear
```

### Ridge Regression (v0.2)

```
bash

python train.py --version v0.2 --model-type ridge --alpha 1.0
```

### Random Forest (v0.2)

```
bash

python train.py --version v0.2 --model-type random_forest \
  --n-estimators 100 --max-depth 10
```

### Training Outputs:

- `models/model_vX.X.pkl` - Trained model + scaler
- `models/metrics_vX.X.json` - Performance metrics

## Testing

```
bash

# Install test dependencies
pip install pytest httpx

# Run tests
pytest test_api.py -v

# Test with Docker
docker build -t diabetes-predictor:test .
docker run -d -p 8000:8000 --name test diabetes-predictor:test
curl http://localhost:8000/health
docker stop test && docker rm test
```

## Project Structure

```
.
├── .github/
│   └── workflows/
│       └── train-and-deploy.yml # CI/CD pipeline
├── models/ # Trained models (gitignored)
│   ├── model_v0.1.pkl
│   └── metrics_v0.1.json
├── app.py # FastAPI service
├── train.py # Training pipeline
├── requirements.txt # Python dependencies
├── Dockerfile # Container definition
├── CHANGELOG.md # Version history & metrics
└── README.md # This file
```

## Security & Privacy

- **No PHI/PII:** Demo uses synthetic scikit-learn dataset
- **In Production:** Would require:
  - HIPAA-compliant infrastructure
  - Data encryption at rest and in transit

- Audit logging
- Authentication/authorization
- De-identification of training data

## Roadmap

### v0.2 (Next)

- ☐ Ridge/Random Forest models
- ☐ Feature selection/engineering
- ☐ Improved risk calibration
- ☐ A/B testing framework

### Future

- ☐ Model monitoring & drift detection
- ☐ Explainability (SHAP values)
- ☐ Real-time retraining
- ☐ Multi-model ensemble
- ☐ Dashboard UI

## License

MIT License - See LICENSE file

## Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Train and test your model
4. Commit changes (`git commit -m 'Add amazing feature'`)
5. Push to branch (`git push origin feature/amazing-feature`)
6. Open Pull Request (CI will auto-test)

## Contact

MLOps Team - [your-email@example.com](mailto:your-email@example.com)

**Project Link:** [https://github.com/YOUR\\_USERNAME/YOUR\\_REPO](https://github.com/YOUR_USERNAME/YOUR_REPO)