Report on

Medical Disease Classification and Progression

By

Maryam Ibrahim Mahmoud
Asmaa Mahmoud Saad
Maryhan Sabry Mousa
Malak Aly Ahmed
Naira Gamal

1. Introduction

1.1 Background

Chronic diseases such as diabetes, anemia, kidney failure, liver failure, and ischemic heart disease (IHD) represent a significant global health burden. Early detection and accurate progression prediction of these conditions can dramatically improve patient outcomes and reduce healthcare costs. Traditional diagnostic methods often rely on single-point measurements and clinical experience, which may not capture complex disease patterns.

1.2 Objectives

This project aims to:

- Develop machine learning models to classify five chronic diseases
- Predict disease progression percentages based on laboratory test results
- Compare different machine learning approaches
- Identify the most accurate models for clinical decision support

1.3 Dataset Overview

The dataset contains 30,000 patient records with 36 clinical features including:

- Basic demographics (age, gender)
- Comprehensive blood test results (hemoglobin, RBC, platelet count, etc.)
- Disease labels (diabetes, anemia, kidney failure, liver failure, IHD)
- Calculated progression metrics

---

2. Data Preprocessing

2.1 Data Cleaning

- Removed duplicate entries (none found)
- Handled missing values with median imputation
- Performed label encoding for categorical variables:
    - Gender: Men → 0, Women → 1, Children → 2

    - **Urine Protein/Glucose: Absent → 0, Trace → 1**

2.2 Feature Engineering

- **Calculated additional derived features (e.g., GFR)**
- **Outlier handling using IQR method**
- **Early data splitting (train/test: 80/20) to prevent leakage**

2.3 Feature Selection

Used mutual information classification to identify the most predictive features for each disease, retaining features with average scores ≥ 0.001.

2.4 Feature Scaling

Applied MinMaxScaler to normalize features to [0,1] range.

---

3. Model Building and Evaluation

3.1 Introduction to Model Selection

To effectively classify and predict the progression of five critical medical conditions, we implemented and compared three machine learning models:

1. **XGBoost (Extreme Gradient Boosting)**
2. **Neural Networks**
3. **Ridge Regression (for regression tasks only)**

These models were selected based on their strong performance in structured/tabular medical data, interpretability, generalization ability, and suitability for both classification and regression problems. Below is a breakdown of their methodologies and performance across both classification and progression tasks.

3.2 Classification Models

3.2.1 XGBoost Classifier

Why Chosen:
XGBoost is a state-of-the-art ensemble learning algorithm based on gradient boosting. It is

widely used in healthcare predictive modeling due to its high accuracy, robustness to noise, handling of missing values, and interpretability.

How It Works:
XGBoost builds an ensemble of decision trees sequentially. Each new tree corrects the errors of the previous trees by minimizing a differentiable loss function (e.g., binary cross-entropy for classification) using gradient descent. Regularization techniques (L1/L2) are also applied to prevent overfitting.

Advantages:

- High performance on structured data
- In-built feature importance evaluation
- Scalability and support for parallel computation
- Handles imbalanced and multi-label classification

Performance Results:

- **Train Accuracy:** 0.9976
- **Test Accuracy:** 0.9905
- **Classification Metrics (Macro F1-score):** 1.00
- **Sample-based F1-score:** 0.96
- **Confusion Matrices (Per Condition):** Extremely low misclassification rates (e.g., 2 false negatives in 3422 diabetes cases)

3.2.2 Neural Network Classifier

Why Chosen:
Neural Networks can model complex, non-linear relationships in data. They are particularly effective when feature interactions are non-obvious or highly intricate, making them suitable for biomedical classification tasks.

How It Works:
We implemented a feedforward neural network with:

- Dense (fully connected) layers
- ReLU activation functions
- Cross-entropy loss function

- Adam optimizer

The network learns weights and biases during training through backpropagation by minimizing the loss between predicted and true labels.

Performance Results:

- **Train Accuracy:** 0.873
- **Test Accuracy:** 0.843
- **Per-Disease Accuracy Ranges:** 93% to 100%
- **Strongest Areas:** Diabetes (F1-score = 1.00), Kidney Failure (F1-score = 0.97)
- **Weaker Areas:** Slightly lower performance in anemia and ischemic heart disease due to higher false positives/negatives

3.3 Progression Models (Regression)

In progression prediction tasks, we aimed to estimate the percentage of disease progression for each condition based on relevant disease-specific laboratory test results (e.g., glucose levels for diabetes, hemoglobin for anemia). The following models were evaluated:

3.3.1 Ridge Regression

Why Chosen:
Ridge Regression is a linear model that adds L2 regularization to reduce overfitting and improve generalization on correlated features. It was used as a strong, interpretable baseline.

How It Works:
It minimizes the least squares error with an added penalty term (α * sum of squares of coefficients). This shrinks the coefficients and reduces variance without significantly increasing bias.

Performance:

- Best for quick estimation with moderate accuracy

- **R^2 Scores:**
    - Diabetes: 0.8619
    - Anemia: 0.5181
    - Liver Failure: 0.8698
    - Kidney Failure: 0.9321
    - IHD: 0.8552

3.3.2 Neural Network Regressor

Why Chosen:
A neural network regressor is more flexible than linear models and can capture non-linear disease progression patterns.

How It Works:
Similar to the classification network, but optimized with **mean squared error loss** and **linear activation** in the output layer. Tuned using hyperparameters like learning rate and hidden layer size.

Performance:

- Significantly improved RMSE over Ridge
- **R^2 Scores:**
    - Diabetes: 0.9806
    - Anemia: 0.7669
    - Liver Failure: 0.9801
    - Kidney Failure: 0.9892
    - IHD: 0.9468

3.3.3 XGBoost Regressor

Why Chosen:
Just as in classification, XGBoost delivers state-of-the-art regression accuracy due to its ability to model non-linearity and interactions efficiently.

How It Works:
It applies gradient boosting to regression trees, minimizing squared error loss iteratively while applying shrinkage and subsampling.

Performance:

- **Best overall performance in all diseases**
- **Lowest RMSE values** across the board
- **Highest R^2 scores:**
    - Diabetes: 0.9915
    - Anemia: 0.8326
    - Kidney Failure: 0.9975
    - Liver Failure: 0.9933
    - IHD: 0.9538

3.4 Special Case: Diabetes Subtype Classification (No Diabetes, Type 1, Type 2)

Using XGBoost, we further classified patients into three diabetes categories.

Performance:

- **Train & Test Accuracy:** 0.99325
- **F1-score (Macro):** 0.99
- **Support across classes:**
    - No diabetes: 1224
    - Type 1: 1406
    - Type 2: 1370

This showcases XGBoost's robustness in multi-class settings as well.

3.5 Summary of Model Comparison

Task	**Best Model**	**RMSE (Progression)**	**R^2 (Progression)**	**Accuracy (Classification)**
Diabetes Progression	XGBoost	2.34	0.9915	0.99 (multi-class)
Anemia Progression	XGBoost	11.85	0.8326	________
Kidney Failure Progress.	XGBoost	1.48	0.9975	________
Liver Failure Progress.	XGBoost	2.13	0.9933	________
IHD Progression	XGBoost	6.40	0.9538	________

3.6 Conclusion

The model building process demonstrated that **XGBoost consistently outperformed other models** in both classification and regression tasks due to its ensemble-based architecture and fine control over learning dynamics. Neural Networks provided solid performance and handled complexity well, while Ridge Regression served as a useful baseline with faster training and interpretability. The ensemble learning capabilities of XGBoost make it the most suitable model for deployment in medical decision-support systems due to its high reliability and accuracy.

---

4. Deployment

4.1 MLflow Purpose in This Project:

MLflow Purpose in This Project MLflow is integrated into this project to track experiments, log model parameters and metrics, and save trained models. This enables version control, performance comparison, and reproducibility for machine learning workflows.

How MLflow Was Integrated

1. MLflow Imports At the beginning of the file: import mlflow import mlflow.sklearn import mlflow.tensorflow

2. Experiment Declaration In the main() function: mlflow.set_experiment("Depi_OOP_Project")

3. Tracking XGBoost Classification Inside train_xgboost() in ClassificationModelTrainer: with mlflow.start_run(run_name="XGBoost_Classifier"): mlflow.log_params(grid_search.best_params_) mlflow.log_metric("train_accuracy", accuracy_score(self.y_train, y_train_pred)) mlflow.log_metric("test_accuracy", accuracy_score(self.y_test, y_pred)) mlflow.sklearn.log_model(best_model, "xgboost_model")

4. Tracking Neural Network Classification Inside train_neural_network() in ClassificationModelTrainer: with mlflow.start_run(run_name="Neural_Network_Classifier"): mlflow.log_param("learning_rate", 0.0005) mlflow.log_metric("train_accuracy", accuracy_score(self.y_train, y_pred_train)) mlflow.log_metric("test_accuracy", accuracy_score(y_test_np, y_pred_test)) mlflow.tensorflow.log_model(model, "nn_model")

What's Not Yet Tracked with MLflow

- Regression models (Ridge, Neural Network, XGBoost Regressor) are not yet tracked.
- Visual artifacts (confusion matrices, plots) are also not logged.

How to Run the Project with MLflow mlflow run . or

python DepiProject_with_MLflow.py mlflow ui .

API Integration for Model Deployment

After training and evaluating the best-performing models, an API was created using **FastAPI** to allow real-time access to disease predictions.

Technical Summary

- The API receives health-related input data (e.g., hemoglobin, RBC, WBC, glucose, gender).
- It loads the saved model from **MLflow** using its run ID.
- When a POST request is sent with patient data, the model returns a prediction instantly.

How It Works

1. User (doctor or app) sends patient data to the /predict endpoint.
2. The API converts the data into a DataFrame format.
3. The model processes the input and returns the result.

Testing the API

- Run locally with: uvicorn app:app --reload
- Test the endpoint using Swagger UI at: http://localhost:8000/docs

Future Plan

This API can be connected to a web dashboard or healthcare system to automate and simplify diagnosis support.