

Youssef Ahmed Ibrahim
223101109
Assignment 2 : Diabetes Classification

Overview

This analysis focuses on the **Multi-Layer Perceptron (MLP)** implementation for binary diabetes classification based on the provided PyTorch code (Diabetes_MLP_Classifier.ipynb).

The task is **Binary classification** (Diagnosed Diabetes: Yes/No). The framework used is **PyTorch** with **scikit-learn preprocessing**. The target variable is Binary: **0** (No Diabetes) or **1** (Diagnosed Diabetes).

Dataset & Preprocessing

The dataset contains 100,000 samples (implied by 80,000 train + 20,000 test). Initial features are 31 columns.

Preprocessing Steps in Code:

Data Leakage Removal: Columns diabetes_stage and diabetes_risk_score were removed because they directly indicate or are calculated based on the diagnosis.

Categorical Encoding: gender, ethnicity, education_level, income_level, employment_status, and smoking_status were **One-Hot Encoded** using pd.get_dummies with drop_first=True.

Feature Scaling: All features were standardized using **StandardScaler** normalization.

The final input feature count after these steps is **40** features.

Data Split

The total samples are 100,000.

Training: 80% or 80,000 samples.

Test: 20% or 20,000 samples.

Class Distribution (Implied Imbalance)

The class distribution shows a slight imbalance:

Class 1 (Diagnosed Diabetes): 59,998 samples ($\approx 60\%$)

Class 0 (No Diabetes): 40,002 samples ($\approx 40\%$)

Model Architecture (MLPClassifier)

The implemented MLP is a deep, sequential network that differs from the original assignment's suggested architecture 8.

Input Dimension: 40 features.

Architecture Detail: The network consists of five linear layers with an increasing, then decreasing, number of neurons:

Linear (40 to 256) + **BatchNorm1d** + **ReLU** + **Dropout(0.3)**

Linear (256 to 128) + **BatchNorm1d** + **ReLU** + **Dropout(0.3)**

Linear (128 to 64) + **BatchNorm1d** + **ReLU** + **Dropout(0.2)**

Linear (64 to 32) + **BatchNorm1d** + **ReLU**

Linear (32 to 2) for the final output logits9.

Regularization: The implementation includes **Batch Normalization** and **Dropout** in the first three hidden layers, contrasting with the original report's configuration of "None".

Training Configuration

Parameter	Value
Optimizer	Adam
Learning Rate (lr)	\$0.001\$
Loss Function	CrossEntropyLoss
Batch Size	256 (Note: Original document specified 32 15)
Epochs	10 (Note: Original document specified 20 16)
Device	'cpu' (A CUDA-enabled GPU was not found/used in the execution)

Results

The model was trained for **10 epochs**. The progression is summarized below:

EPOCH	TRAIN LOSS	TRAIN ACC	TEST LOSS	TEST ACC
1	0.3382	85.20%	0.2752	88.50%
2	0.2818	88.20%	0.2569	89.53%
3	0.2681	88.87%	0.2502	90.09%
4	0.2617	89.30%	0.2466	90.08%
5	0.2589	89.48%	0.2435	90.51%
6	0.2556	89.68%	0.2422	90.47%
7	0.2541	89.79%	0.2401	90.75%
8	0.2497	89.99%	0.2433	90.69%
9	0.2505	89.87%	0.2351	90.70%
10	0.2485	90.07%	0.2379	90.88%

Final Performance (Epoch 10)

METRIC	TRAINING	TESTING
ACCURACY	90.07%	90.88%
LOSS	0.2485	0.2379
OVERFITTING GAP	-0.81% (Test Acc > Train Acc)	Minimal

Key Observations & Comparison

The implemented code introduces several modifications compared to the configuration in the original assignment document.

A significant improvement in the code was the **explicit removal of data leakage columns** (`diabetes_stage`, `diabetes_risk_score`) during preprocessing.

The final test accuracy achieved is **90.88%**, which is very close to the 90.99% reported in the original unregularized report. Crucially, the implemented model shows **no signs of overfitting** based on the final accuracies, with the test accuracy being slightly higher than the training accuracy (90.88% vs 90.07%). This suggests the addition of regularization successfully stabilized the training, addressing a major concern noted in the original analysis.