Neural Network Implementation on Medical Appointment No-Show Dataset

Jadhav Rudranil Sandiip
Roll Number: 24095043
CSoC 2025 – Intelligence Guild

June 2, 2025

1 Introduction

This report presents a comparative study of implementation of Artificial Neural Network on Medical Appointment Dataset. This implementation is performed by two approaches 1. Scratch (without deep learning frameworks) 2. Using Pytorch.

2 Dataset: About and Pre-processing

2.1 About

The dataset contains information on over 100,000 medical appointments. It includes 14 features. Primary objective being to analyze and predict whether a patient will show up for their scheduled medical appointment

2.2 Preprocessing

To prepare the dataset for modeling, several preprocessing steps were applied, including label encoding, feature engineering, data cleaning, handling class imbalance, and scaling. These steps are described in detail below.

- **Label Encoding:** Categorical variables were converted into numerical representations(0 and 1) to make them suitable for machine learning algorithms. Label encoding was used for No-Show and Gender columns.
- **Feature Engineering:**Using Scheduled Day and Appointment Day a new useful feature i.e Delay was introduced furthermore Delay was categorized in 3 one hot encoded form (No dealy , Small Delay , Large delay).
- **Data Cleaning** The dataset was consisted some errors eg. Age of person was recorded to be negative , inconsistency in encoding in Handcap column. This was fixed and data was made error free.

- **Train-Test Split:** The dataset was split into training and testing sets using `train_test_split()` with 80% of data used in training and the remaining 20% in testing.
- **Feature Scaling:** Standardization was applied to scale all features.
- **Handling Class Imbalance:** The No-Show was heavily imbalanced (almost 80 : 20 ratio). The data balancing oversampling technique was used to address this challenge. I tried to handle imbalance using SMOTE (Synthetic Minority Oversampling Technique) and using imblearn but it performing resample method gave better F1 score, which was somthing I coundn't figure out why as I read that generally SMOTE is better than random oversampling.
- **Saving Processed Data:** The final training and testing sets were exported to CSV files named `train.csv` and `test.csv`.

3 Methodology

3.1 Part 1: Pure Python Implementation

- Implemented ANN in Python with 2 hidden layers of 8 and 16 neurons by scratch.
- SGD optimizer was used and the decaying learning rate was implemented.
- The code was heavily inspired from Sentdex's book for NN from scratch as it was only material I used for understanding code of ANN.
- Relu activation was used for hidden layers where as Softmax for output layer.

3.2 Part 2: PyTorch Implementation

- Implemented ANN in Python with 2 hidden layers of 8 and 16 neurons by using PyTorch. For fair comparison the Architecture of NN wasn't altered much.
- I found out that Adam performed very slightly better than SGD optimizer and decided to use it instead of SGD in part 2.
- Relu activation was used for hidden layers where as Softmax for output layer

4 Evaluation Criteria

4.1 Metrics Used

- Confusion Matrix
- F1 score
- Precision and Recall and Area Under its curve(PR-AUC)

4.2 Results Table

Implementation	Metric	Class 0	Class 1
Pure Python (Scratch)	Precision	0.72	0.63
	Recall	0.53	0.79
	F1	0.61	0.70
	PR-AUC	0.6318	
PyTorch	Precision	0.75	0.63
	Recall	0.52	0.83
	F1	0.61	0.72
	PR-AUC	0.6815	

4.3 Confusion Matrix Of Test Dataset

	Predicted: 0	**Predicted: 1**
Actual: 0	9409	8233
Actual: 1	3682	13962

Table 1: Confusion Matrix for Scratch Implementation

	Predicted: 0	**Predicted: 1**
Actual: 0	9125	8517
Actual: 1	3006	14636

Table 2: Confusion Matrix for PyTorch Implementation