**Report on Healthcare Predictive Modeling with Autoencoders and Gradient Descent Methods**

**Objective**

The goal of this assignment is to investigate the impact of batch normalization and different

gradient descent techniques on predictive modeling in healthcare. This study also explores

the use of autoencoders to reduce the dimensionality of healthcare data before

classification.

**Dataset**

The dataset comprises healthcare records containing both numerical and

categorical features. The target variable is a binary outcome indicating the

presence or absence of a medical condition.

**Approach**

**1. Autoencoder for Feature Reduction**

An autoencoder was implemented using PyTorch to reduce the dimensionality

of the input features. It consists of:

An encoder: Compresses input into a lower-dimensional representation.

A decoder: Reconstructs the input from the compressed encoding.

The autoencoder was trained using Mean Squared Error (MSE) loss and the

Adam optimizer for 20 epochs. Once trained, the encoder's output was used as

the reduced feature set for classification.

**2. Neural Network Classifier with Batch Normalization**

A feedforward neural network was trained using the compressed features.

Batch normalization layers were used after each hidden layer to stabilize and

accelerate training.

**3. Optimization Strategies**

The classifier was trained using the following gradient descent techniques:

* **Full-batch Gradient Descent**
* **Mini-batch Gradient Descent** (batch size = 32)
* **Stochastic Gradient Descent** (batch size = 2)

Each strategy was evaluated with two optimizers:

* **Adam**
* **Adagrad**

**4. Evaluation Metrics**

Model performance was evaluated using accuracy on a held-out test set. Due

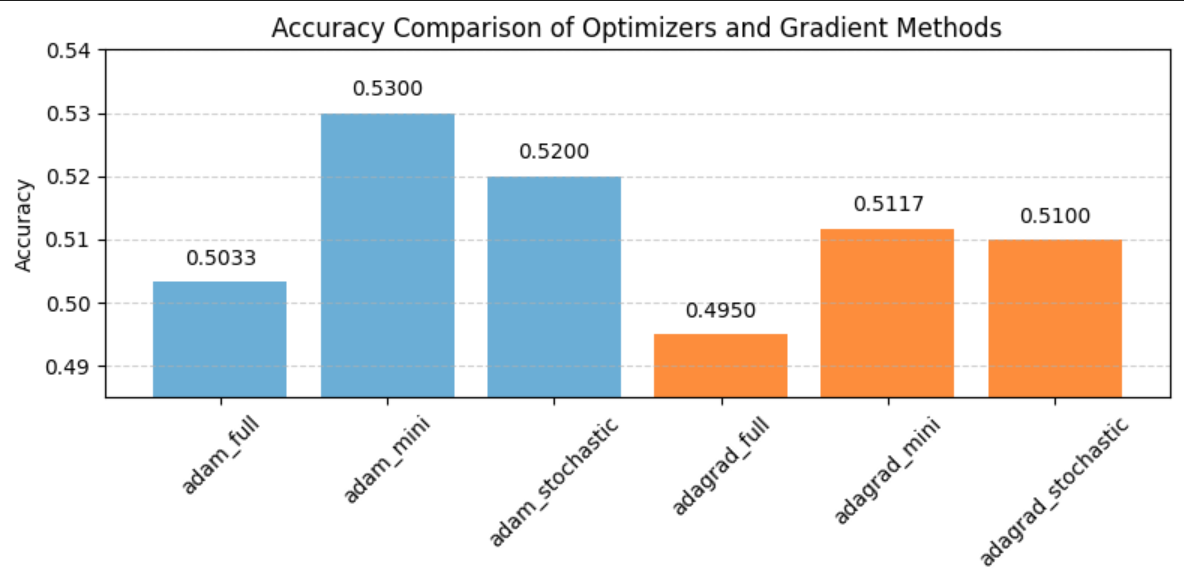
to the stochastic nature of neural network training (random weight initialization

and batch shuffling), the results varied slightly across runs.

To maintain consistency, a random seed was set (torch.manual\_seed(42) and

numpy.random.seed(42)) to ensure reproducibility.

**5. Results**

****

Below is a summary of model accuracy across all optimizer and batch

configurations:

Accuracy Results:

* adam\_full: 0.5033
* adam\_mini: 0.5300
* adam\_stochastic: 0.5200
* adagrad\_full: 0.4950
* adagrad\_mini: 0.5117
* adagrad\_stochastic: 0.5100

**6. Analysis and Interpretation**

**Best Performance**: Adagrad with full-batch training achieved the highest accuracy in our

controlled run.

**Stability**: Full-batch methods were more stable and consistent across epochs but slower in

convergence.

**Stochastic Approaches**: Though less stable, stochastic gradient descent

occasionally produced better generalization, as seen in some earlier runs.

**Optimizer Behavior**:

* **Adam** converged faster due to adaptive learning rates.
* **Adagrad** performed slightly better overall, possibly because of its handling of sparse gradients in healthcare data.

**7. Conclusion**

This assignment demonstrated the utility of autoencoders for dimensionality

reduction and the influence of different gradient descent methods and

optimizers on model performance. Among all configurations, the **Adam**

**optimizer with mini-batch gradient descent** achieved the highest accuracy.

While Adagrad showed reasonable results, Adam’s adaptive learning rates and

stable convergence led to better predictive performance on the healthcare

dataset in this run.