**Project Title**

**Using Deep Learning to Classify Human Gene Variants (SNVs) as Deleterious or Neutral**

**Project Goal**

The aim of this project was to build a deep learning pipeline that could classify single nucleotide variants (SNVs) in human genes as either *deleterious* (harmful) or *neutral*, based on their functional effects.

**Approach & Workflow**

1. **Getting the Data**

I downloaded SNV data from the **ClinVar database**, which provides rich annotations about the clinical significance of genetic variants. The data was in VCF (Variant Call Format), which contains information like chromosome number, base changes, positions, and clinical labels.

**2. Cleaning and Labeling**

Then, I parsed and cleaned the data to extract only the useful fields. Based on the CLNSIG values, I labeled the variants as:

* **1 (deleterious)**: if they were marked as pathogenic or likely pathogenic
* **0 (neutral)**: if they were benign or likely benign

The datset was extremely large, to keep things manageable, I used a **subset of 50,000 variants**, and removed any entries with missing or invalid data.

**3. Feature Extraction**

I encoded the nucleotide bases (A, C, G, T) into numbers, using a simple mapping. Also encoded the **variant type** (SNV vs. Indel) as a one-hot feature.  
This gave a **feature vector of 4 values per variant**, which we used to train our model.

**4. Splitting the Dataset**

The dataset was split into training and test sets using **stratified sampling**, to ensure both classes (deleterious and neutral) were represented evenly in both sets.

**5. Model Design**

I used a simple **Artificial Neural Network (ANN)** with:

* An input layer matching the 4 input features
* Two hidden layers (128 and 64 neurons) using ReLU activation and Dropout (0.3)
* A final output layer with a single neuron for binary classification

**6. Training the Model**

I used:

* **Binary Cross-Entropy Loss with Logits**, with weighted classes to address imbalance
* **Adam optimizer** with a learning rate of 0.001
* **10 epochs** of training
* **Batch size of 32**

**7. Evaluation**

Once trained, the model was evaluated on the test set using:

* **Accuracy**
* **Precision**
* **Recall**
* **F1-score**

**8. Visualization**

* A **confusion matrix**
* An **ROC curve**
* A **training loss curve**

**Results**

| **Metric** | **Score** |
| --- | --- |
| Accuracy | 78.96% |
| Precision | 20.76% |
| Recall | 10.33% |
| F1 Score | 13.80% |