# Iris Classification Neural Network Project

## Overview

This project implements a neural network model using PyTorch to predict classification scores for iris flowers. The model takes four input features (sepal length, sepal width, petal length, and petal width) and predicts three scores corresponding to the probability of the flower being Setosa, Versicolor, or Virginica.

## Dataset

The project uses a modified version of the classic Iris dataset (`iris\_data.csv`). The dataset contains:

- Input features:

- SepalLengthCm: Length of the sepal in centimeters

- SepalWidthCm: Width of the sepal in centimeters

- PetalLengthCm: Length of the petal in centimeters

- PetalWidthCm: Width of the petal in centimeters

- Target variables:

- SetosaScore: Score indicating Setosa classification

- VersicolorScore: Score indicating Versicolor classification

- VirginicaScore: Score indicating Virginica classification

## Project Structure

The project is organized in a Jupyter notebook (`Homework.ipynb`) with the following main components:

1. Data Preprocessing:

- Loading and splitting the dataset (80% training, 20% testing)

- Standardizing features using StandardScaler

- Converting data to PyTorch tensors

- Creating DataLoader objects for batch processing

2. Model Architecture:

- Input layer: 4 features

- Hidden layer: 512 units

- Output layer: 3 units (one for each iris type score)

3. Training Configuration:

- Loss function: Mean Absolute Error (L1Loss)

- Optimizer: Adam with learning rate 0.001

- Batch size: 8

- Number of epochs: 10

## Dependencies

- PyTorch

- pandas

- scikit-learn

- numpy

## Model Performance

The model shows good convergence during training:

- Starting loss: ~0.41

- Final training loss: ~0.29

- Test loss: ~0.21

## Usage

To use the trained model for predictions:

# Prepare input data (example)

sample = torch.tensor([[sepal\_length, sepal\_width, petal\_length, petal\_width]], dtype=torch.float32)

# Make prediction

with torch.no\_grad():

prediction = model(sample)

The output will be three scores corresponding to Setosa, Versicolor, and Virginica classifications.

## Example Prediction

The notebook includes an example prediction:

```python

sample1 = torch.tensor([[7, 4, 7, 3]], dtype=torch.float32)

# Results in scores: [-2.44, 0.28, 2.21]

```

## Future Improvements

Potential enhancements could include:

- Adding additional hidden layers

- Implementing dropout for regularization

- Experimenting with different architectures

- Adding model evaluation metrics (accuracy, precision, recall)

- Implementing cross-validation

- Adding early stopping