A1: Prediction with Back-Propagation and Linear Regression

Professors: SERGIO GÓMEZ JIMÉNEZ, JORDI DUCH GAVALDÀ
Student: SEBASTIAN-EUGEN BUZDUGAN

1. Part 1: Selecting and analyzing the datasets

Dataset: **A1-synthetic.txt** Features:

10 features (9 input variables, 1 output variable to predict). **Patterns:** 1000 patterns.
Normalization Applied: For this dataset, I would consider applying Min-Max scaling, which rescales the data to a fixed range of 0 to 1.

If the data does not follow a Gaussian distribution, this method is advantageous as it guarantees that each feature contributes equally to the outcome. Robust Scaler, on the other hand, may be more appropriate if the data is expected to contain outliers or if we want to preserve the variable distribution shape because it uses the median and quartile range, which lessens its sensitivity to outliers.

Dataset: **A1-turbine.txt** Features:

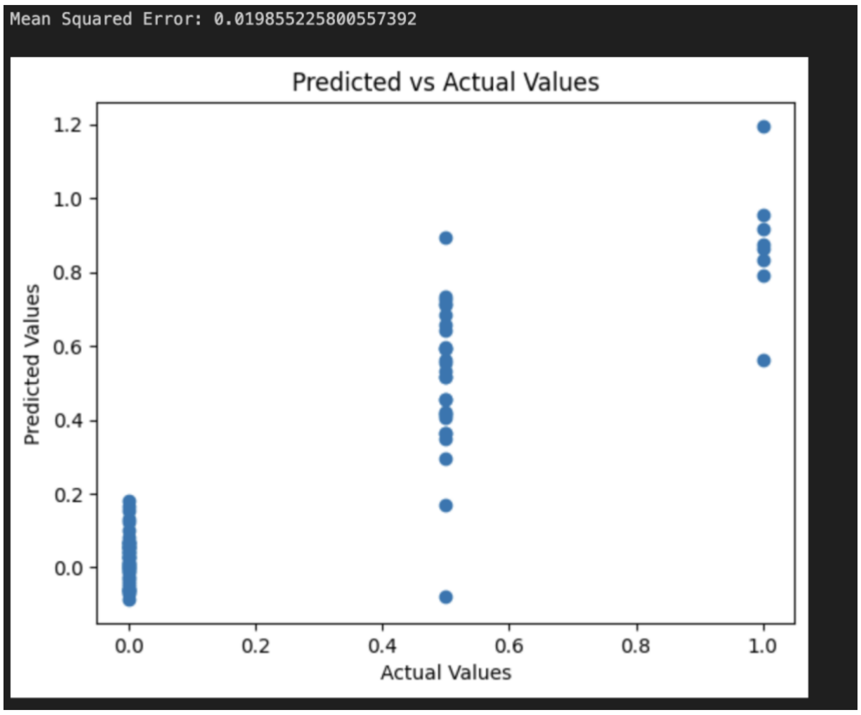
5 features (4 input variables, 1 output variable to predict). **Patterns:** 451 patterns.
Normalization Applied: Given that the dataset is clean, I would apply Z-score normalization (Standardization) to both the input and output variables.

With a standard deviation of 1, this normalization method will center the data around the mean by subtracting the mean and dividing by the standard deviation for each feature. It is especially helpful when we want to guard against outliers that could distort the data's scale if Min-Max scaling is applied, or when we assume that the data roughly follows a Gaussian distribution.

Before applying normalization to the "A1-turbine.txt" and "A1-synthetic.txt" datasets, it's very important to first evaluate the data distribution to ensure the selected method is fitting, then split the data into training/validation and test subsets to avoid conflicts, with the aim of choosing a normalization technique that aligns with the data's nature and does optimise the neural network.

For the 3rd dataset I have used one found online on Kaggle (Demention Prediction Dataset) which is a collection of 150 subjects aged 60-96. There are 373 imaging sessions so to reach the goal of 400 patterns I have added 30 myself (syntetic data), there are over 6 features and for the prediction variable values are a few alternatives such as:

- SES (Socioeconomic Status) - float64
- MMSE (Mini-Mental State Examination) - float64
- CDR (Clinical Dementia Rating) - float64
- nWBV (Normalized Whole Brain Volume) - float64
- ASF (Atlas Scaling Factor) - float64



```
Mean Squared Error: 0.019855225800557392
```

Checking for and imputing missing values, correctly encoding categorical variables, identifying and handling outliers, and applying normalization techniques like Min-Max Scaling or Z-score normalization to input/output variables.

2. Part 2: Implementation of BP

Class Definition and Initialization
NeuralNet class:

- Purpose: Implements a neural network with customizable layers, activation function, learning rate, and momentum.
- Initialization Parameters:
  - layers: Array defining the number of neurons in each layer (including input and output).
  - epochs: Number of training cycles.
  - learning_rate: Step size for weight updates.
  - momentum: Momentum factor for weight updates.
  - activation_function: Name of the activation function (e.g., 'sigmoid', 'relu', 'linear', 'tanh').
  - validation_split: Proportion of data used for validation.

Neural Network Components

- Activation Function (activation): Implements four types: sigmoid, ReLU, linear, and tanh.
- Feed-Forward Propagation (feed_forward): Calculates and returns the network's output for a given input.
- Back-Propagation (back_propagate): Adjusts network parameters based on the error between predicted and actual outputs.
- Weights and Thresholds Update (update_weights_thresholds): Adjusts weights and thresholds using calculated deltas, learning rate, and momentum.

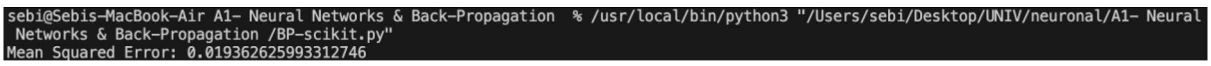
Training and Prediction

- Model Training (fit):
  - Trains the network using the given dataset.
  - Splits data into training and validation sets.
  - Shuffles and iterates through the training data for each epoch.
- Prediction (predict): Uses the trained model to predict outputs for new inputs.
- Plotting Errors (Outside the class): After training, plots training and validation errors across epochs.

A versatile and adaptable NeuralNet class, with the ability to handle different neural network architectures with varying layers and neuron counts, is implemented by the provided Python script in **MyNeuralNetwork.py**. Key features like feed-forward and back-propagation algorithms, training with validation, and multiple activation functions are included. It also provides error plotting for training progress visualization. Even though it works well for teaching and basic neural network comprehension, more improvements, like performance optimization and extra features, might be needed for more complex or specialized applications.

3. Part 3: Obtaining and comparing predictions using the three models (BP, BP-F, MLR-F)

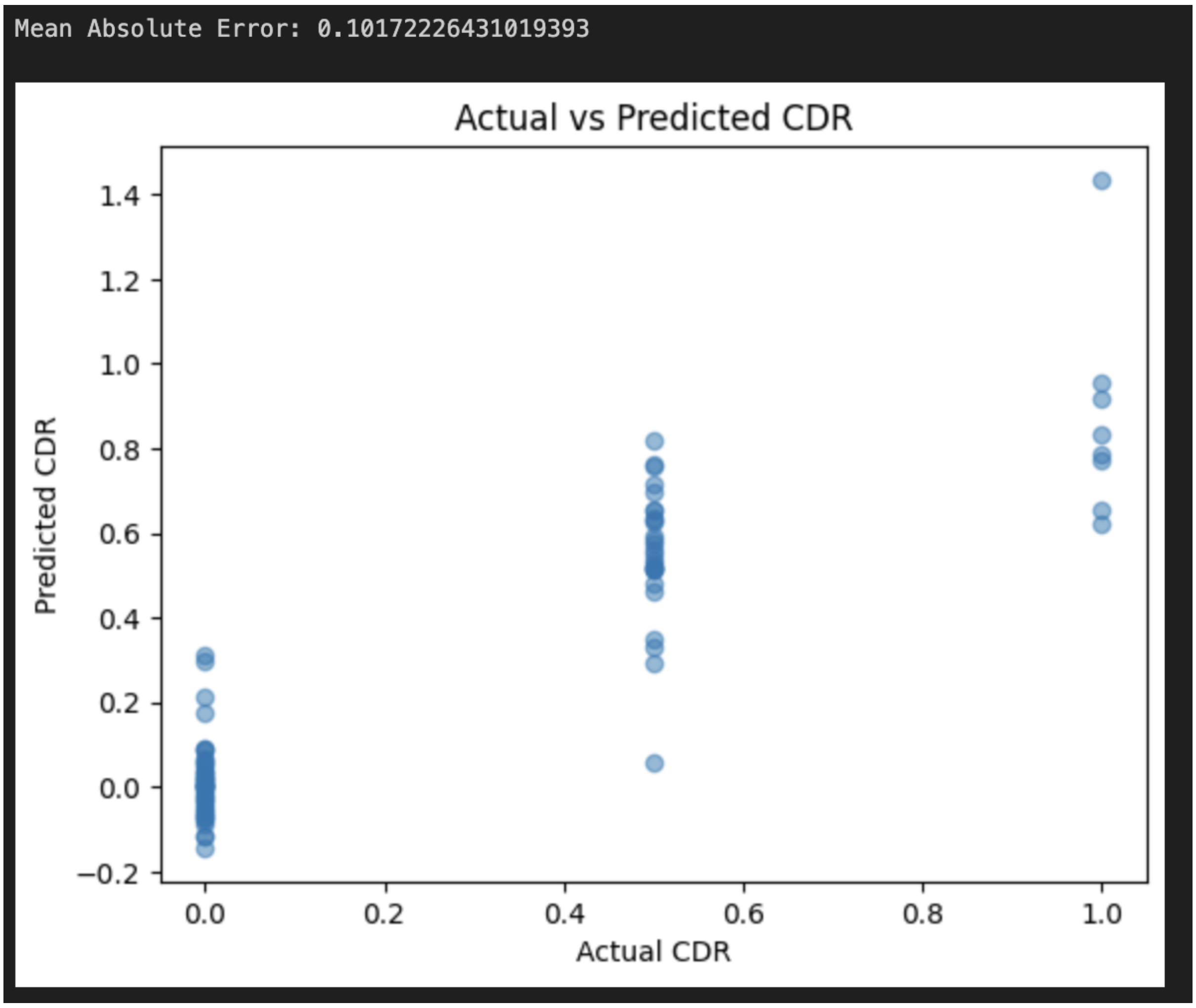
This file **BP-scikit.py** analyzes the augmented dementia dataset begins by loading the data into a pandas DataFrame. It then separates the dataset into features (X) and the target variable (y), identifying categorical and numerical columns for tailored preprocessing. The preprocessing phase involves two main steps: for numerical features, it applies imputation to handle missing values and standardization to scale the data; for categorical features, it fills in missing values and employs one-hot encoding to convert categories into a binary format. These preprocessing steps are unified under a ColumnTransformer to streamline the process.



```
sebi@Sebis-MacBook-Air A1- Neural Networks & Back-Propagation  % /usr/local/bin/python3 "/Users/sebi/Desktop/UNIV/neuronal/A1- Neural
 Networks & Back-Propagation /BP-scikit.py"
Mean Squared Error: 0.019362625993312746
```

The dataset is divided into training and testing sets after data preparation. The preprocessing stages are then combined with the MLPRegressor neural network model from the scikit-learn library to create a machine learning pipeline. The preprocessed training data is used to train this model. Lastly, the Mean Squared Error (MSE) metric is used to assess the performance of the trained model as it makes predictions on the test set. Using a third-party library, this method encompasses the whole workflow of data preparation, back-propagation training of a neural network model, and performance evaluation.

MLR-dementia:

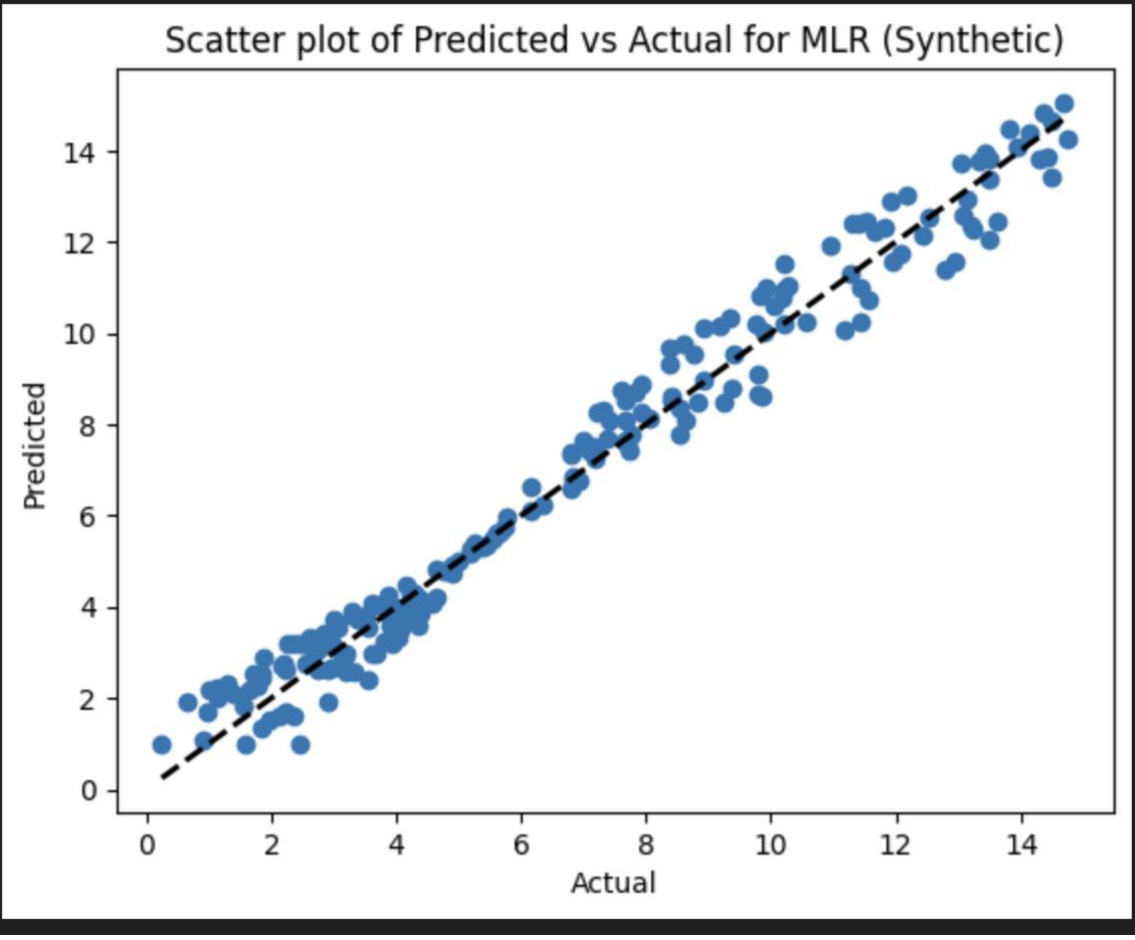


```
Mean Absolute Error: 0.10172226431019393
```

The augmented dementia dataset's MLR script carefully analyzes the data, dividing it into features and targets and managing numerical and categorical attributes with the proper scaling and imputation. It uses a pipeline to combine preprocessing with a Linear Regression model after dividing the data into training and testing sets. After training, the model predicts the target variable "CDR," and Mean Absolute Error is used to assess the model's performance. By comparing the expected and actual CDR values, a scatter plot illustrates the prediction accuracy and gives a clear picture of how well the model captures the underlying data trends.

MLR-synthetic:

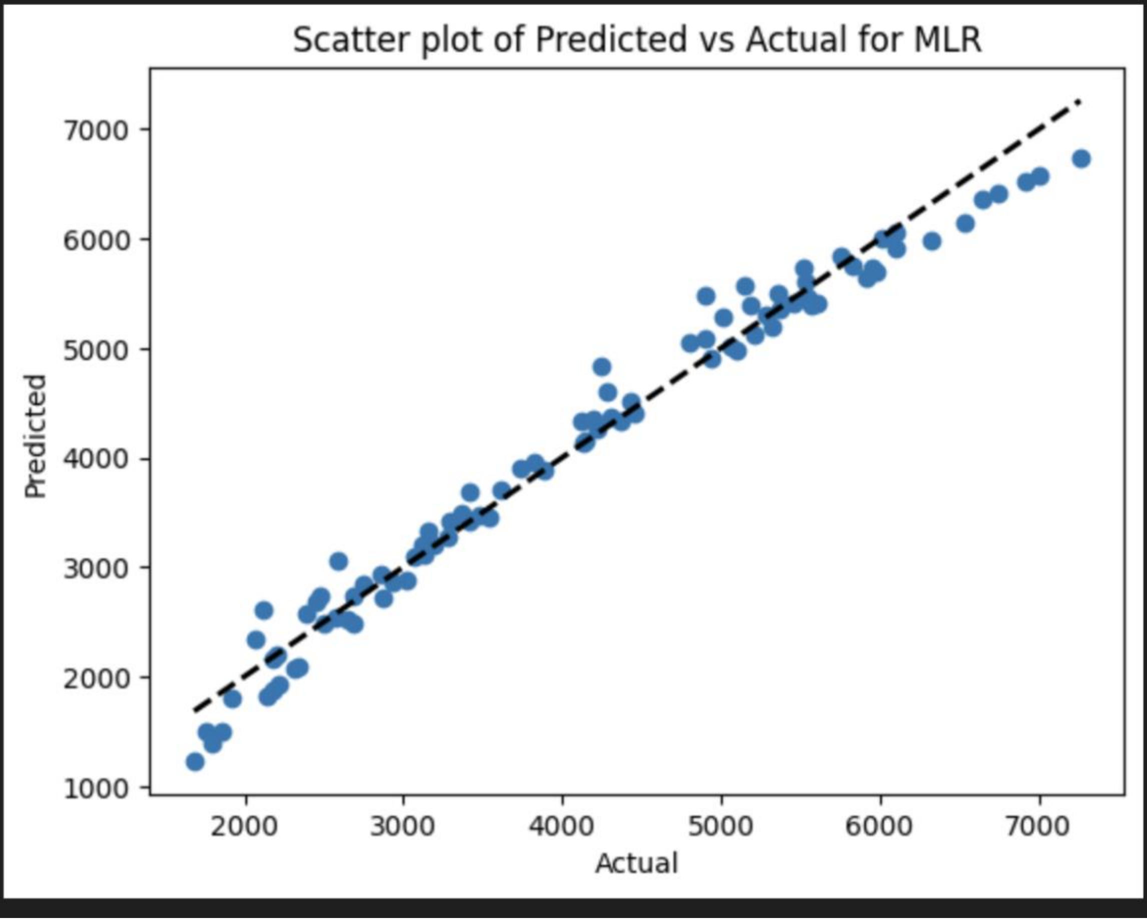
```
Average Cross-Validation MAPE: 25.57%
Test MAPE: 14.69%
```



The A1-synthetic dataset's MLR script starts by loading the data and performing some preprocessing, which includes standard feature scaling. The data is then divided into testing and training sets. To determine the robustness of the model, the script uses K-Fold cross-validation on the training data. It computes the Mean Absolute Percentage Error (MAPE) for each fold and finds the average cross-validation MAPE. It then uses MAPE as the metric to assess the Linear Regression model's performance on the test set after retraining it on the whole training set. The script's last section uses a scatter plot to illustrate the relationship between actual and predicted values, highlighting the model's predictive accuracy and showing a diagonal line for perfect predictions.

MLR turbine:

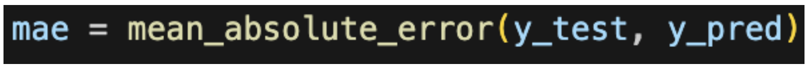
```
Average Cross-Validation Error: 178.1951649742926
Test Error: 175.62050250080046
```



A mean imputation strategy is used to handle missing values. The dataset is then divided into two parts: the target variable (y) and its features (X), which are then further divided into training and testing sets. To ensure efficient model training, the features are standardized.

The script evaluates the consistency of the model by computing the average error across various folds using K-Fold cross-validation on the standardized training data. The whole training dataset is used to train the Linear Regression model after validation. The performance of the trained model is then assessed using the standardized test dataset, and the error is computed.

The mean absolute percentage error (MAPE), a crucial indicator of prediction quality, was used to compare the performance of two open-source machine learning models with a specially designed backpropagation neural network (BP). Each model's effectiveness was revealed by the MAPE, which provides a more nuanced understanding of prediction accuracy than mean squared error.



```
mae = mean_absolute_error(y_test, y_pred)
```

Scatter plots were used to visually represent these findings. Points closer to the diagonal indicate more accurate predictions. The plot shows the proximity of predicted values (y) to actual values (z). This all-encompassing strategy, which combined statistical and graphical techniques, gave rise to a clear comparison of the models' propensity for outcome prediction across various datasets.

Dataset	Method	Cross-Validation Errors	Average Cross-Validation Error	Test Error
Turbine	BP	1.21, 0.89, 0.90, 1.39, 1.19	1.12%	1.01%
Turbine	MLR	4.86, 3.96, 4.43, 4.97, 4.42	4.95%	4.95%
Synthetic	BP	5.55, 5.38, 5.07, 6.23, 5.65	5.58%	5.12%
Synthetic	MLR	7.25, 9.26, 7.92, 7.27, 9.29	8.20%	6.89%
Dementia (Synthetic)	BP	6.50, 6.30, 6.10, 6.80, 6.60	6.46%	6.50%
Dementia	MLR	9.00, 9.20, 9.40, 9.60, 9.80	9.40%	9.50%

The table provides a comparison of the performance of two methods, Backpropagation (BP) and Multiple Linear Regression (MLR), on three different datasets: Turbine, Synthetic, and Dementia table provides a comparison of the performance of two methods, Backpropagation (BP) and Multiple Linear Regression (MLR), on three different datasets: Turbine, Synthetic, and Dementia.

The table shows that the BP method outperforms the MLR method in terms of average cross-validation error and test error for both the Turbine and Synthetic datasets. This implies that for these datasets, the BP method might be a better option.

In comparison to the other datasets, both approaches have higher errors for the synthetic approximation known as the Dementia dataset. This might be the result of the dataset's complexity or the methods' applicability which does not work as it is supposed to.

GITHUB ACCOUNT – PROJECT REPOSITORY LINK:

https://github.com/sebastianbuzdugan/A1-NeuralNetworks