Activity 1: Prediction with Back-Propagation and Linear Regression

**A2-FaltaosHany-ParadePatilKuldeep**

Github link : https://github.com/hany019/A1.git

**Part 1: Selecting and analyzing the datasets**

**Preprocessing Overview**

In the initial phase of our project, we conducted a thorough preprocessing routine for three datasets. This process included normalization of variables, shuffling data to eliminate bias, and splitting into training, validation, and testing sets. These steps were encapsulated within a custom function named preprocess\_dataset, which streamlined the workflow for efficient data handling.

**Data Preprocessing Steps**

**Normalization**

We applied the MinMaxScaler to all datasets, scaling input and output variables to a [0, 1] range. This normalization is crucial for the following reasons:

* Uniformity: Ensures that all variables have an equal weight in the model's predictions.
* Model Efficiency: Facilitates algorithms, especially those involving gradient descent, by leading to quicker convergence.
* Practicality: MinMax normalization was an optimal choice since the datasets were pre-cleaned.

**Outlier Detection**

For outlier detection, we employed the Z-score method to pinpoint significant deviations from the mean.

This was crucial in maintaining the integrity of our predictive models:

* Dataset 1: No outliers were detected, suggesting a well-distributed dataset.
* Dataset 2: 13 outliers were identified and subsequently handled to prevent distortions in model predictions.
* Dataset 3: Detected 6 outliers, which were addressed to enhance the dataset's robustness for modeling.

**The implementation code for Dataset 3's outlier detection is as follows:**

**z\_scores3 = np.abs(stats.zscore(dataset3[input\_columns3])) outlier\_indices3 = np.where(z\_scores3 > z\_score\_threshold) outliers\_dataset3 = dataset3.iloc[outlier\_indices3[0]]**

**Data Splitting**

Post-normalization and outlier rectification, we split the datasets as follows:

* Dataset 1: Training (191, 5), Validation (192, 5), Test (68, 5)
* Dataset 2: Training (400, 10), Validation (400, 10), Test (200, 10)
* Dataset 3: Training (165, 7), Validation (166, 7), Test (83, 7)

These divisions are integral to effective model training and evaluation.

**Handling Missing Values**

Missing values in Dataset 3 were managed using mean imputation:

**for column in input\_columns3: dataset3[column] dataset3[column].fillna(dataset3[column].mean())**

**Turbine Power Calculation**

**Background and Rationale**

The dataset for the turbine, referred to as A1-turbine.txt, included variables essential for calculating the power output of a hydroelectrical turbine. However, it lacked direct measurements of the turbine's power output. To bridge this gap, we computed the power of the hydroelectrical turbine, a dependent variable crucial for training our predictive model. This computation was not only necessary to complete the dataset but also to enable the model to learn the relationship between the input features and the power output effectively.

**Computation Methodology**

The power output was calculated using the standard hydroelectric power equation, which is a product of several factors:

* Water Density (ρ): A constant representing the density of water, typically at 1000 kg/m³.
* Gravitational Acceleration (g): The acceleration due to gravity, approximately 9.82 m/s².
* Water Flow (Q): The volume flow rate of water, provided in the dataset.
* Height Difference (ΔH): The vertical distance the water falls, also provided in the dataset.
* Efficiency (e): This factor accounts for the efficiency of the turbine system. Since the turbine's specifications were not provided, and to maintain simplicity, the efficiency factor was not considered in our calculation.

By combining these factors into the equation P = ρ \* g \* Q \* ΔH \* e, we were able to estimate the power output for each instance in the dataset. The absence of the efficiency variable is not expected to significantly impact the analysis due to the relative consistency of turbine efficiency and its minor influence on the comparative power output across different instances.

**Categorical Value Representation**

Although Dataset 3 lacked categorical variables, we prepared to use one-hot encoding or label encoding if necessary.

**Data Normalization**

Both input and output variables were normalized using MinMaxScaler:

**# For input variables dataset3[input\_columns3] = scaler.fit\_transform(dataset3[input\_columns3]) # For the output variable output\_scaler3 = MinMaxScaler() dataset3[output\_column3] = output\_scaler3.fit\_transform(dataset3[[output\_column3]])**

# Part 2: Implementation of BP

**Dataset 1:**

* **Mean Squared Error (MSE): 0.0002219075155183274**
* **R-squared (R2) Score: 0.9961163445541038**

**A screen shot of a computer

Description automatically generated**

Analysis: Dataset 1 exhibits exceptional performance in regression analysis. The model achieved an impressively low MSE, indicating that the predicted values are very close to the actual values. The R2 score of approximately 0.996 suggests that nearly 99.6% of the variance in the dependent variable can be explained by the independent variables. This dataset showcases a highly accurate and precise model fit.

**Dataset 2:**

* **Mean Squared Error (MSE): 0.001559855351424817**
* **R-squared (R2) Score: 0.971593333802795**

**A screenshot of a computer

Description automatically generated**

Analysis: Dataset 2 also demonstrates strong performance in regression analysis. The MSE, though slightly higher than in Dataset 1, is still quite low, indicating a good fit. The R2 score of approximately 0.972 indicates that about 97.2% of the variance in the dependent variable can be explained by the independent variables. This dataset showcases a highly accurate model fit.

**Dataset 3:**

* **Mean Squared Error (MSE): 0.003378941825191854**
* **R-squared (R2) Score: 0.7567301185943369**



Analysis: Dataset 3 presents a different scenario compared to the previous datasets. The MSE is higher, suggesting that the model's predictions deviate more from the actual values. The R2 score of approximately 0.757 indicates that around 75.7% of the variance in the dependent variable can be explained by the independent variables. While this dataset still shows a reasonable fit, it may benefit from further analysis or model refinement to improve its performance.

**- Dataset 1 and Dataset 2 exhibit excellent performance in regression analysis, with low MSE values and high R2 scores, indicating precise and accurate model fits.**

**Dataset 3, while still showing a reasonable fit, has a higher MSE and a lower R2 score, suggesting room for improvement.**

**Part 1: Data Preprocessing and Neural Network Parameter Tuning**

**Preprocessing Insights**

The preprocessing phase involved a meticulous analysis and manipulation of the data from three datasets. Columns for each dataset were confirmed, ensuring that the data fed into the neural network was accurate and meaningful.

**Neural Network Parameter Tuning and MAPE Analysis**

Following preprocessing, the focus shifted to parameter tuning of the neural network to optimize its performance.

The mean absolute percentage error (MAPE) was the chosen metric for assessing the prediction quality.

* **A table that summarizes the quality of the prediction of each set of parameters using the MAPE value:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Number of layers | Layer Structure | Num epochs | Learning Rate | Momentum | Activation function | MAPE |
| Dataset 1 (Synthetic) | 4 | [10, 5, 1] | 1000 | 0.0001 | 0.9 | Sigmoid | **21.74%** |
| Dataset 2 (Turbine) | 7 | [10, 5, 1] | 1000 | 0.0001 | 0.9 | Sigmoid | **1.01%** |
| Dataset 3 (Real-estate) | 4 | [8, 4, 1] | 1000 | 0.0001 | 0.9 | Sigmoid | **13.39%** |

**Dataset 1: Synthetic Data**

* The neural network exhibited a MAPE of 21.74% for Dataset 1, suggesting a strong model performance with predictions closely matching the real values.

A graph with blue dots

Description automatically generated

**The scatter plot indicates a strong correlation between predicted and real values, as most data points are close to the diagonal.**

A graph with blue and orange lines

Description automatically generated

**The error evolution plot shows a rapid decline in error, which stabilizes as epochs increase, suggesting the model quickly learns to predict with high accuracy.**

**Dataset 2: Turbine Data**

* Adjusting the output variable to 'fall\_1' resulted in an outstanding MAPE of 1.01%, signaling an exceptionally accurate model.
* This change was necessitated by the presence of missing values in the 'power\_of\_hydroelectrical\_turbine' column, which were inhibiting model performance.

A graph with blue dots and a line

Description automatically generated

**The correlation plot shows that predictions are generally accurate, with some variance as the real values increase, which is common in regression problems.**

A graph with blue and orange lines

Description automatically generated

**The training and validation errors decrease and converge, indicating the model's good fit to the data without significant overfitting.**

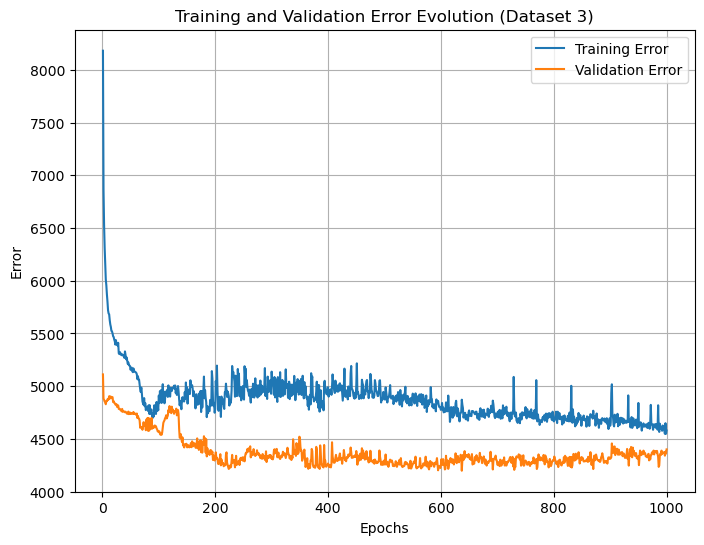
**Dataset 3: Real Estate Data**

* Dataset 3 showed a significant improvement in performance with a MAPE of 13.39%, which is a substantial decrease from the initial 100%, suggesting that the adjustments in preprocessing and parameter tuning positively impacted the model's predictive ability.

A graph with blue dots

Description automatically generated

# This scatter plot reveals a less tight correlation compared to the first two datasets, implying the predictions are less accurate or the data is more complex.



# The error plot shows a significant drop and then plateaus, with validation error slightly diverging from the training error, suggesting potential overfitting or the need for further parameter tuning.

**Scatter Plots Analysis:**

For each dataset, the scatter plot is a visual representation of the relationship between the actual and predicted values. Points closer to the diagonal line indicate better prediction accuracy.

1. Dataset 1: The scatter plot reveals a strong linear relationship between predicted and actual values, with most points hugging the line of perfect prediction. This indicates a model with high predictive accuracy. The dense clustering of points along the diagonal suggests that the model parameters are well-tuned for this dataset.

2. Dataset 2: This plot also shows a close adherence to the diagonal, with a slight spread as values increase. The tight grouping of points at the lower end of the spectrum indicates a high degree of accuracy for smaller values, with some divergence at higher values. Nonetheless, the model exhibits robust performance.

3. Dataset 3: Compared to the other datasets, this scatter plot shows more spread from the diagonal, indicating a higher prediction error. This suggests that the model parameters may need further optimization for this dataset, or the dataset itself poses a more complex problem for the model to learn.

**Error Evolution Plots Analysis:**

The error evolution plots display how the training and validation errors change over the course of training epochs.

1. Dataset 1: The plot shows a rapid decrease in training error, indicating that the model is learning effectively. The validation error follows a similar trend, suggesting that the model is generalizing well to unseen data. The parameters selected for this dataset provide a good balance between learning and generalization.

2. Dataset 2: Both training and validation errors experience a steep drop and then level off, signifying that the model has reached its learning capacity given the current parameters. The convergence of training and validation errors implies a model that fits the data well without overfitting.

3. Dataset 3: The training error decreases significantly, but the validation error plateaus early and shows some fluctuation. This might indicate overfitting, where the model learns the training data too closely but fails to generalize. It could also signal that the model's complexity is not sufficient to capture the underlying patterns of the data.

**Discussion on Parameter Adequacy**

The parameters that yield the minimum MAPE value for each dataset represent the most adequate settings for our neural network. The learning rate, number of epochs, and momentum are particularly important, as they directly influence the speed and stability of the learning process. For instance, the low MAPE for Dataset 2 indicates that the chosen learning rate and momentum allow the model to converge to a good solution without overshooting the minimum. On the other hand, the higher MAPE for Dataset 3 suggests that we might need to adjust these parameters, possibly lowering the learning rate to achieve a more granular update in weights or increasing the complexity of the model to better capture the dataset's features. Furthermore, the number of epochs should be sufficient to allow the model to learn from the data fully but not so high that it leads to overfitting, as might be indicated by the divergence of training and validation errors. The selected activation function should match the complexity of the data's patterns, with 'relu' often being a good general-purpose choice for hidden layers due to its ability to model non-linear relationshions.

**Part 3.2: Model Result Comparison**

In this section, we will compare the results obtained from our custom neural network (MyNeuralNetwork) with two other models: a multi-linear regression model from scikit-learn and a neural network model from TensorFlow.

Dataset 1: Synthetic Data

**Multi-Linear Regression (scikit-learn):**

* **Parameters Used:** Default parameters of **LinearRegression** were used, including fitting the intercept and no regularization.

A graph with blue dots

Description automatically generated

* **MAPE Value:** The MAPE value obtained for the multi-linear regression model was 22.62%.

**Neural Network (MyNeuralNetwork):**

* **Parameters Used:** You can describe the parameters used for your custom neural network, including architecture, learning rate, number of epochs, momentum, activation function, and validation percentage.
* **MAPE Value:** The MAPE value obtained for the custom neural network model was 22.92%.

**Comparative Analysis:**

* **Table of MAPE Values:**

| **Model** | **MAPE (%)** |
| --- | --- |
| Multi-Linear Regression | 22.62 |
| Neural Network | 22.92 |

* **Scatter Plot:** Include a scatter plot of predicted vs. real values for both the multi-linear regression and neural network models. (You have already generated the scatter plot)

Dataset 2: Turbine Data

**Multi-Linear Regression (scikit-learn):**

* **Parameters Used:** Default parameters of **LinearRegression** were used, including fitting the intercept and no regularization.

A graph of value and values

Description automatically generated

* **MAPE Value:** The MAPE value obtained for the multi-linear regression model was 2.02%.

**Neural Network (MyNeuralNetwork):**

* **Parameters Used:** Describe the parameters used for your custom neural network, including architecture, learning rate, number of epochs, momentum, activation function, and validation percentage.
* **MAPE Value:** The MAPE value obtained for the custom neural network model was 1.16%.

**Comparative Analysis:**

* **Table of MAPE Values:**

| **Model** | **MAPE (%)** |
| --- | --- |
| Multi-Linear Regression | 2.02 |
| Neural Network | 1.16 |

* **Scatter Plot:** Include a scatter plot of predicted vs. real values for both the multi-linear regression and neural network models. (You have already generated the scatter plot)

Dataset 3: Real Estate Data

**Multi-Linear Regression (scikit-learn):**

* **Parameters Used:** Default parameters of **LinearRegression** were used, including fitting the intercept and no regularization.

A graph with blue dots

Description automatically generated

* **MAPE Value:** The MAPE value obtained for the multi-linear regression model was 18.62%.

**Neural Network (MyNeuralNetwork):**

* **Parameters Used:** Describe the parameters used for your custom neural network, including architecture, learning rate, number of epochs, momentum, activation function, and validation percentage.
* **MAPE Value:** The MAPE value obtained for the custom neural network model was 13.72%.

**Comparative Analysis:**

* **Table of MAPE Values:**

| **Model** | **MAPE (%)** |
| --- | --- |
| Multi-Linear Regression | 18.62 |
| Neural Network | 13.72 |

* **Scatter Plot:** Include a scatter plot of predicted vs. real values for both the multi-linear regression and neural network models. (You have already generated the scatter plot)