MALIS Project 1

Written by: Qizhi Pan and Joel Brown
Date: November 4, 2024

1. Introduction

1.1 Overview

In this project, we completed a binary classification task using linear regression based on iris dataset and further, completed a ternary classification task. The accuracy for Task 1 can be 100%, and for Task 2 can be up to 96%.

1.2 ChatGPT Usage

We used ChatGPT to accelarate our coding, complete and revise our code. We originally conceived of a solution to the ternary classification problem and used ChatGPT to help us write the model in Python.

1.3 Contribution

Joel focused on the coding and project report for Task 1 while Qizhi focused on the coding and project report for Task 2.

2. Dataset Description and Analysis

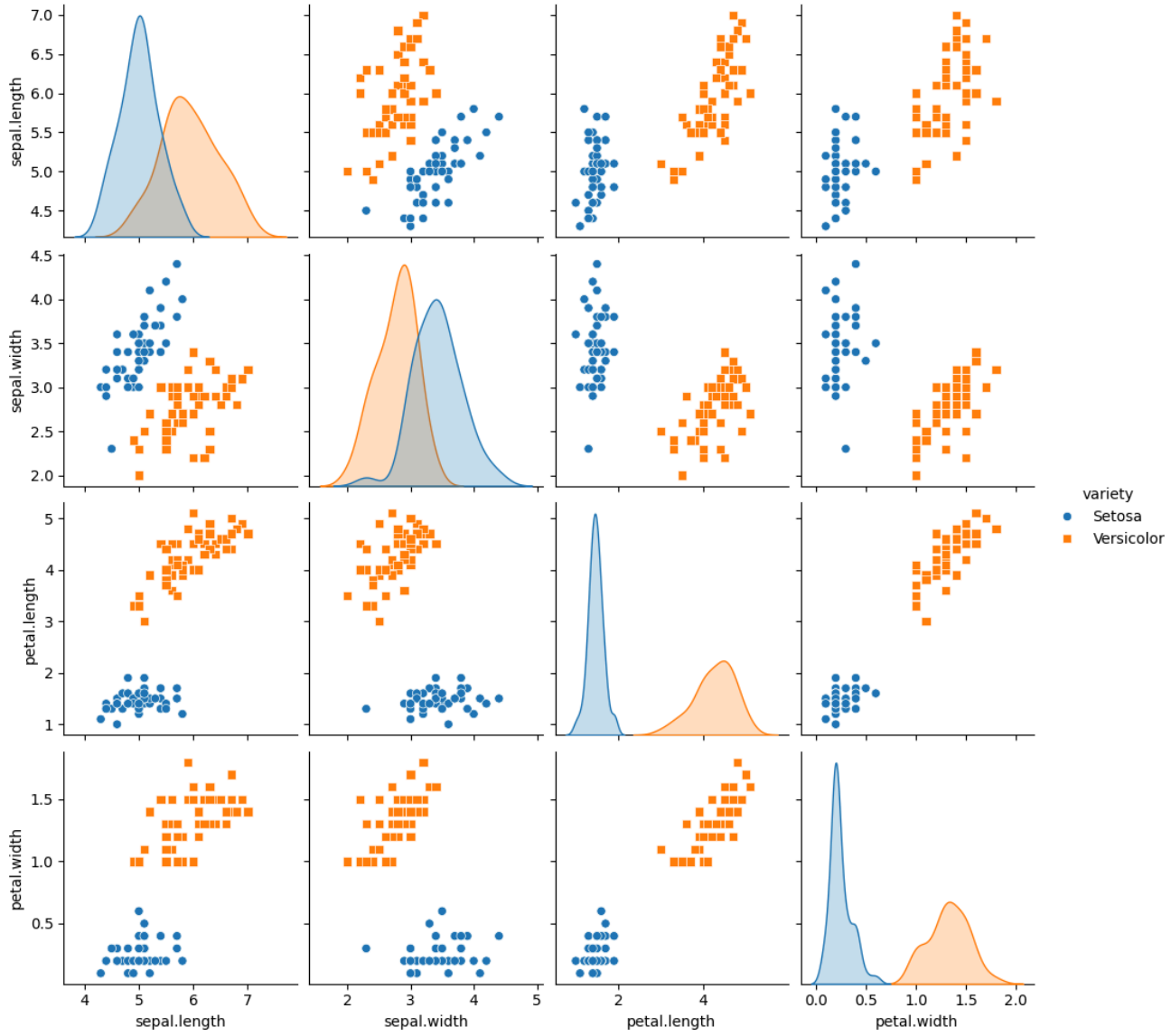
The Iris dataset consists of 150 samples, each representing a flower of one of three species: `Setosa`, `Versicolor`, and `Virginica`. The dataset contains 4 numerical features: `sepal.length`, `sepal.width`, `petal.length`, and `petal.width`, all of them are in centimeters.

The dataset is balanced, with 50 samples per class. Each sample is labeled with one of three classes corresponding to the flower species.

To better handle the classification task with linear regression, we need to understand the data.

2.1 Plot 4 Features in 2 Categories

Considering the binary classifier in **Task 1**, here we list the scatterplots and histograms for 4 features in selected 2 categories:



Here's the definition for each pairplot grid:

	`sepal.length`	`sepal.width`	`petal.length`	`petal.width`
`sepal.length`	Histogram	Scatter Plot	Scatter Plot	Scatter Plot
`sepal.width`	Scatter Plot	Histogram	Scatter Plot	Scatter Plot
`petal.length`	Scatter Plot	Scatter Plot	Histogram	Scatter Plot
`petal.width`	Scatter Plot	Scatter Plot	Scatter Plot	Histogram

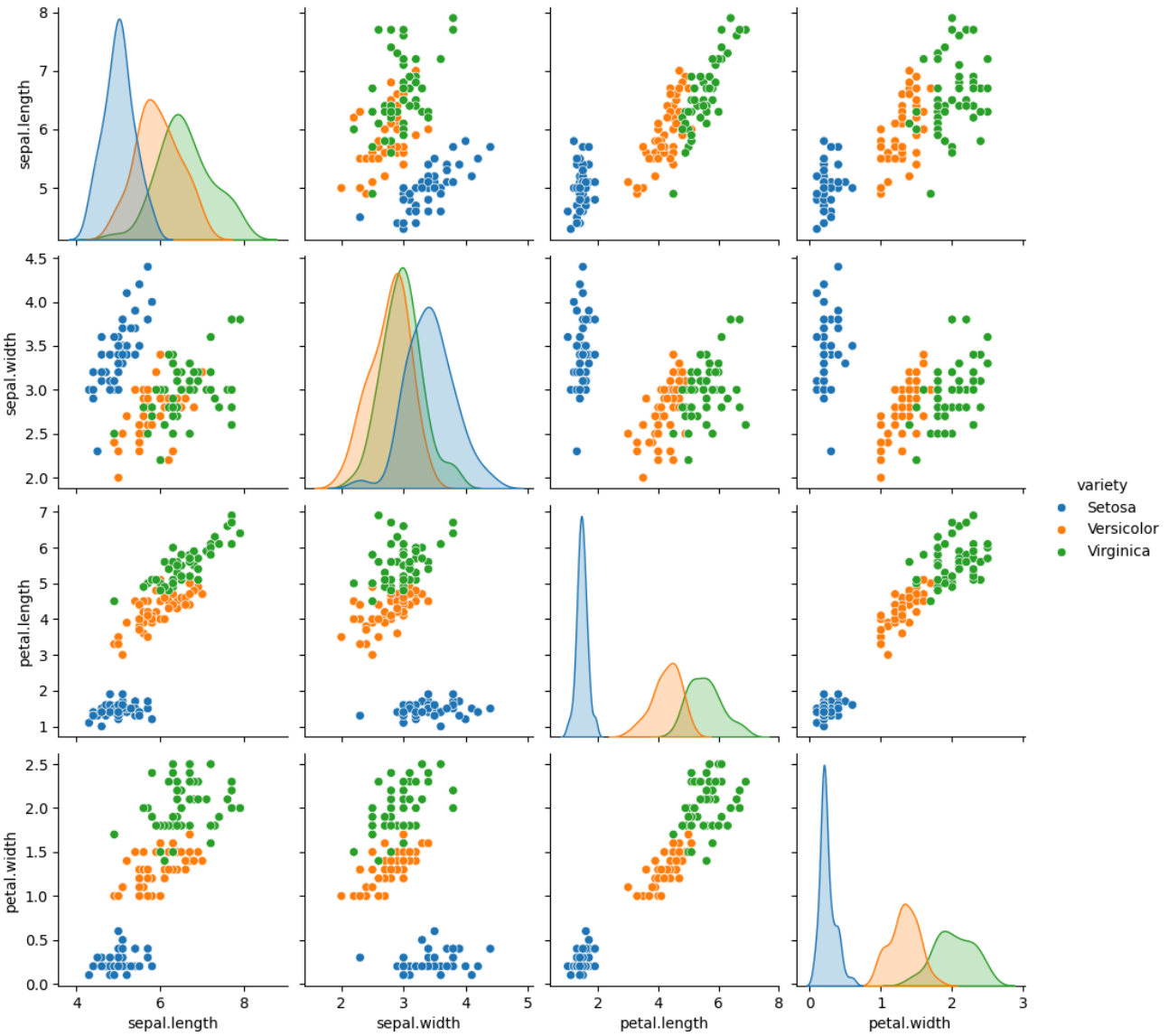
This layout provides a **comprehensive overview** of all pairwise relationships in the dataset.

For the categories we choose(`Setosa` , `Versicolor`), we can see that the distributions of `petal.length` or `petal.width` has distinct boundary between 2 categories and other features are more or less overlapped.

Theoreticlly, either the `petal.length` or `petal.width` itself should be enough to differentiate between these 2 categories. In **Task 1**, we decided to only use `petal.length` to create a regression model.

2.2 Plot 4 Features in 3 Categories

Considering the ternary classifier in **Task 2**, here we plot the 4 features in 3 categories with the same grid definition as above:



`petal.length`
`petal.width`

Thus, in Task 2, we decide to use both `petal.length` and `petal.width` features to create our regression model.

3. Experiments

3.1 Task 1 - Binary Classifier

In this project, we implemented a regression model in Python to classify flowers as either *Setosa (0)* or *Versicolor (1)* based on the feature `petal.length`. The model is based on the hypothesis stemming from our scatterplot, that a linear relationship of the form $y = wx + b$ exists, where:

- `y` - the predicted value.
- `w` and `b` - the parameters to be learned.

In the end, the classification will be:

- If `y <0.5`, classify *Setosa (0)*
- f `y >=0.5`, classify *Versicolor (1)*

We defined the Mean Squared Error (MSE) as the loss function to quantify how accurately the model predicts the target values. To optimize the model parameters (`w` and `b`), we implemented Gradient Descent, which minimizes the loss function by iteratively updating the parameters as follows:

$$w = w - \alpha \cdot \frac{\partial w}{\partial Loss}$$

$$b = b - \alpha \cdot \frac{\partial b}{\partial Loss}$$

where α is the learning rate (defaulted to 0.01 arbituarly)

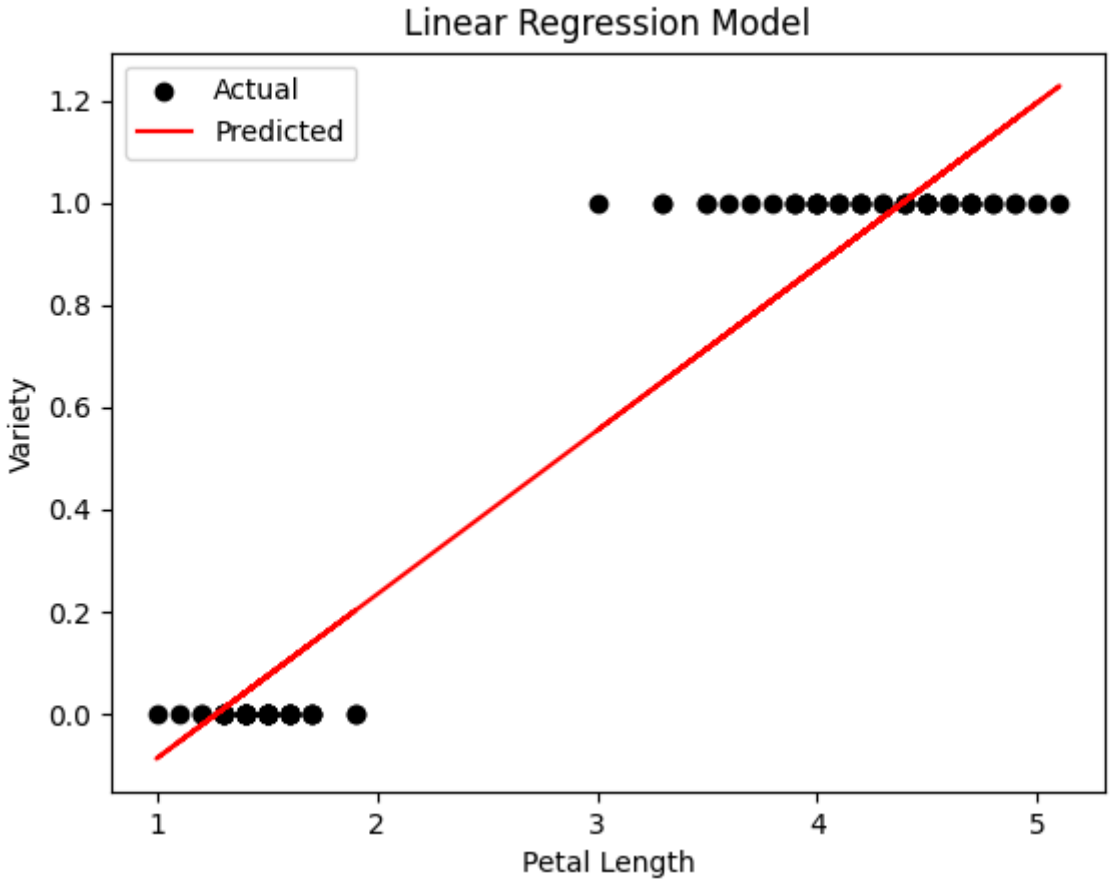
We define `w` and `b` as 0 initially, and for each epoch, the model calculates the gradient, computes the loss, then updates the parameters. We also log the loss in an array for troubleshooting analysis.

The model trains itself over 300 epochs, optimizing `w` and `b` to decrease the loss. After which, the new trained parameters are used and the 0.5 threshold is applied to correctly classify between *Setosa (0)* and *Versicolor (1)*

3.1.1 Linear Regression Model

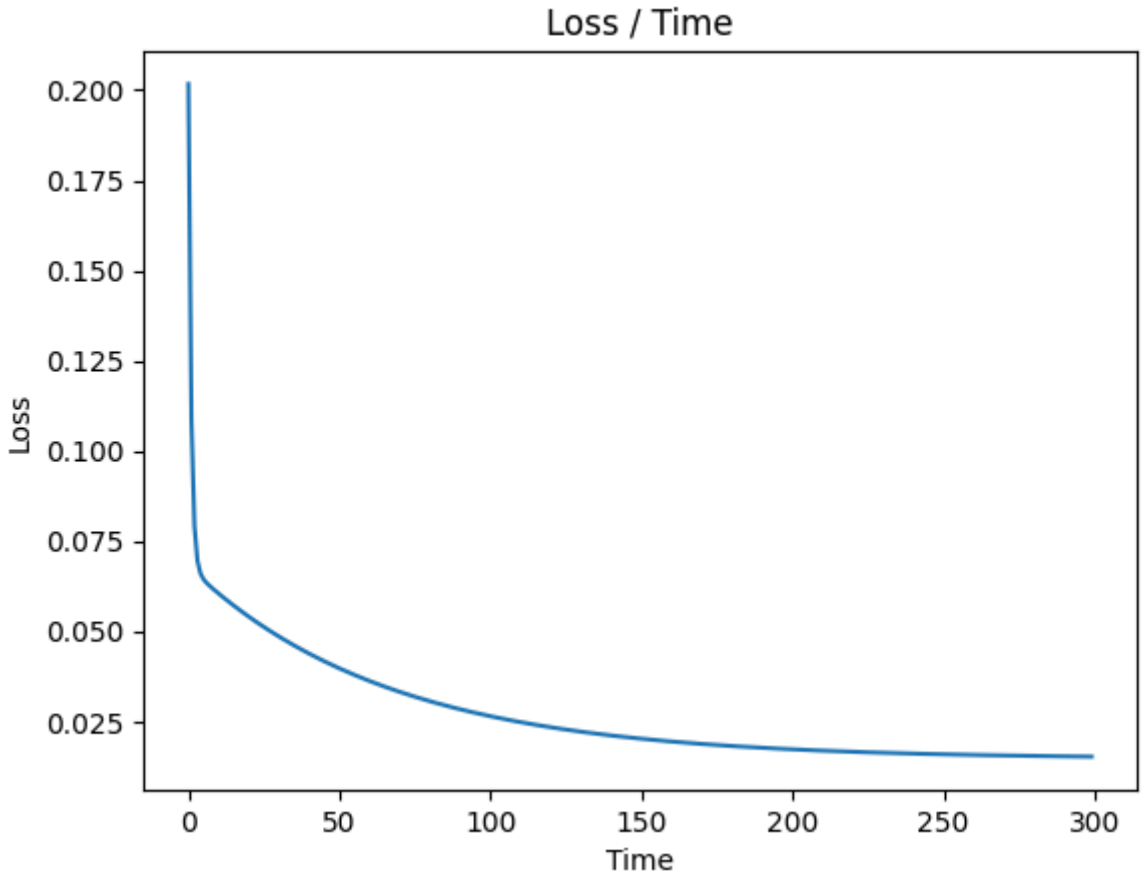
The plot shows how the predicted class labels depend on petal.length

- A red line represents the model's decision boundary (y=0.5y=0.5), dividing the two classes (Setosa and Versicolor)
- Data points below the line are classified as Setosa, and those above are classified as Versicolor



3.1.2 Loss Function

The loss function seemed to smooth out at a local minimum, indicating the model is performing well over the epochs.



3.1.3 Evaluating the Model

To evaluate the model's performance, we used the following metrics:

- Accuracy: 1.0
  - Measured the proportion of correct predictions across the test dataset.
  - The model achieved **100% accuracy**, correctly predicting the class (Setosa or Versicolor) for all test samples
- Confusion Matrix:

$$\begin{pmatrix} 50 & 0 \\ 0 & 50 \end{pmatrix}$$

  Provided a summary of 50 true positives, 0 false positives, 50 true negatives, and 0 false negatives.
  demonstrating a perfect balance between precision and recall

3.1.4 Conclusion

The model performed perfectly on the test dataset, achieving 100% accuracy across all evaluation metrics. While this is an excellent result, it is worth noting that such perfect performance might be **specific to this dataset** and **may not generalize to new or unseen data**.

In this task, we only used one feature `petal.length` out of the 4 available features to classify the flowers.

In the next task, we will explore multi-variable classification, where the dimensions of the dataset will have to increase as we incorporate additional features.

3.2 Task 2 - Ternary Classifier

3.2.1 Solution

We can still use the previous binary classification to solve the ternary classification task. In this case, two binary classifiers are needed. As discussed in 2.2, we need to train new models using `petal.length` and `petal.width`.

Here's the solution:

1. We train two models. Let's say A and B. A tells `Setosa` from the `Virgincia` and `Versicolor` and B tells `Virginica` from the rest.
2. When we classify one sample, we put it into both A and B and get their predict values $y_{A_{pre}}, y_{B_{pre}}$.
3. If $y_{A_{pre}} > y_{B_{pre}}$ and $y_{A_{pre}} >$ a threshold, then we classifed this sample to `Setosa`. If $y_{B_{pre}} > y_{A_{pre}}$ and $y_{B_{pre}} >$ a threshold, then we classified this sample to `Virginaca`. If $y_{A_{pre}}$ and $y_{B_{pre}}$ are both less than the threshold, we classified this sample to `Versicolor`. In our practice, we set the threshold to be 0.75.

3.2.2 Models

Model A:

$$y_{A_{pre}} = W_A^T X + b_A$$
$$W_A = [w_{A1}, w_{A2}], X = [x_1, x_2]$$

Model B:

$$y_{B_{pre}} = W_B^T X + b_B$$
$$W_B = [w_{B1}, w_{B2}], X = [x_1, x_2]$$

3.2.3 Training, Testing and Results

The training procedure in Task 2 was quite similar to Task 1. The slight changes were:

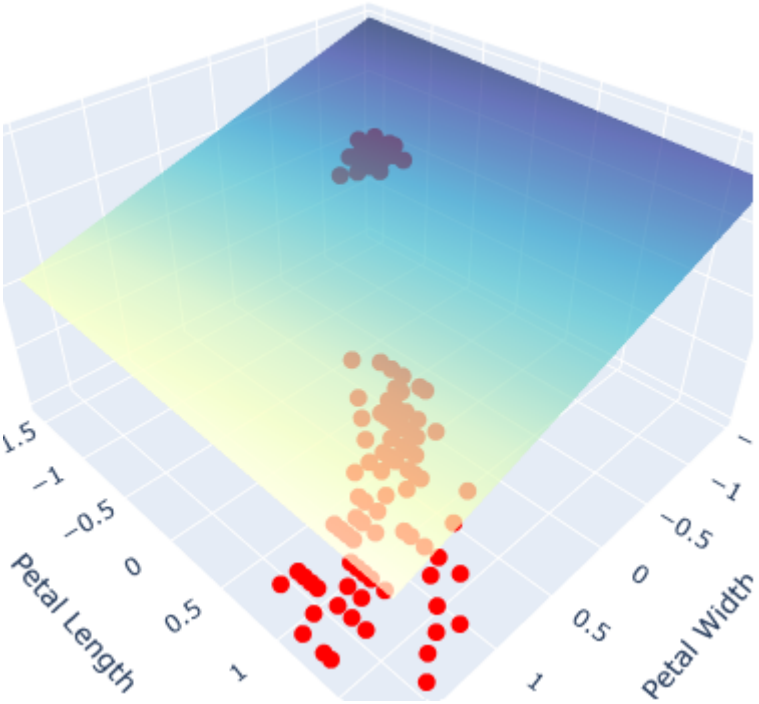
1. In the data preprocessing, to train model A, we set all target `Setosa` to value `1`, the rest to `0`. While trainig model B, we set all target `Versicolor` to value `1`, the rest to `0`.
2. The expression of gradient decents were different.
3. The regression model we got here is a plane, not a line as seen in Task 1.

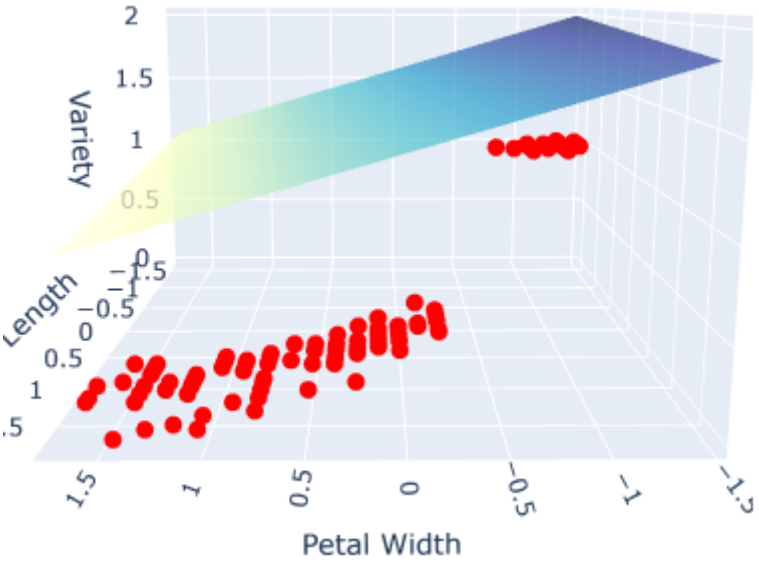
One important change that we did here is we started to use the valid set to reject the trained model. In Task 1, the model was too simple, we're able to get a good model without using the valid set. But in the practice, while doing Task 2, we have to use the valid set to make sure that we get a good model.

One result in a successful training can be seen below:

Model A:

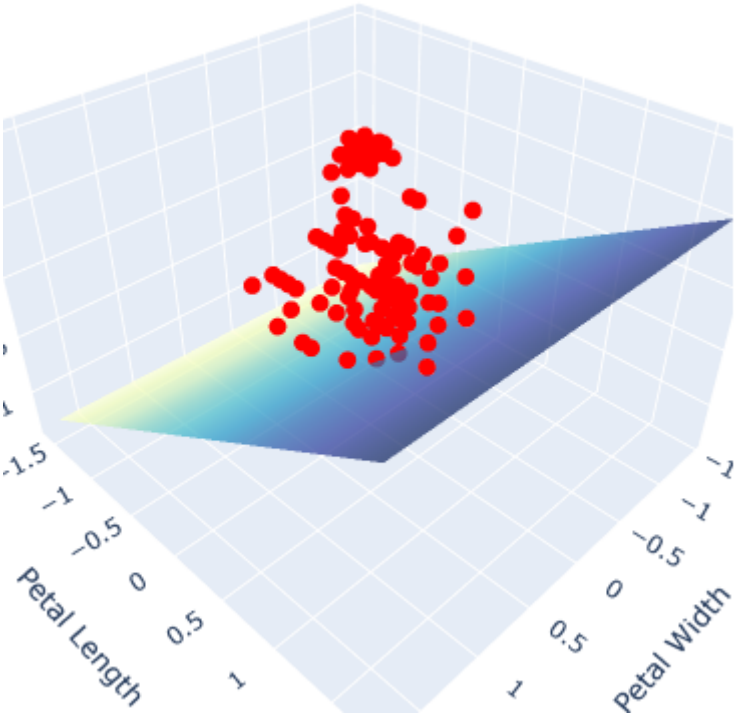
$$[w_{A1}, w_{A2}] = [-0.2971, -0.0644], b_A = 1.5450$$

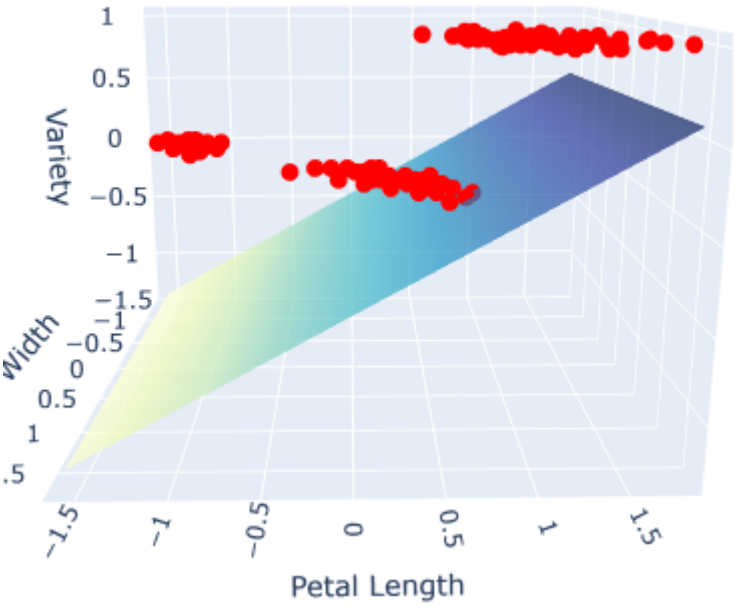




Model B:

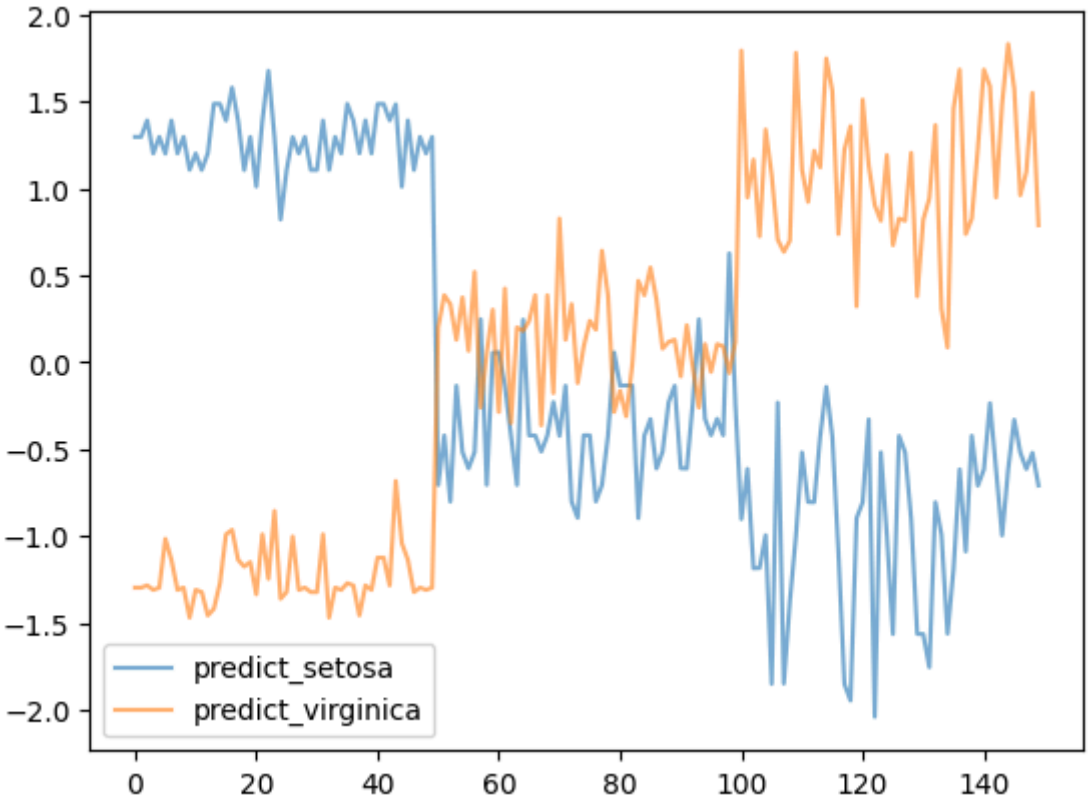
$$[w_{B1}, w_{B2}] = [0.0265, 0.5822], b_B = -0.4908$$



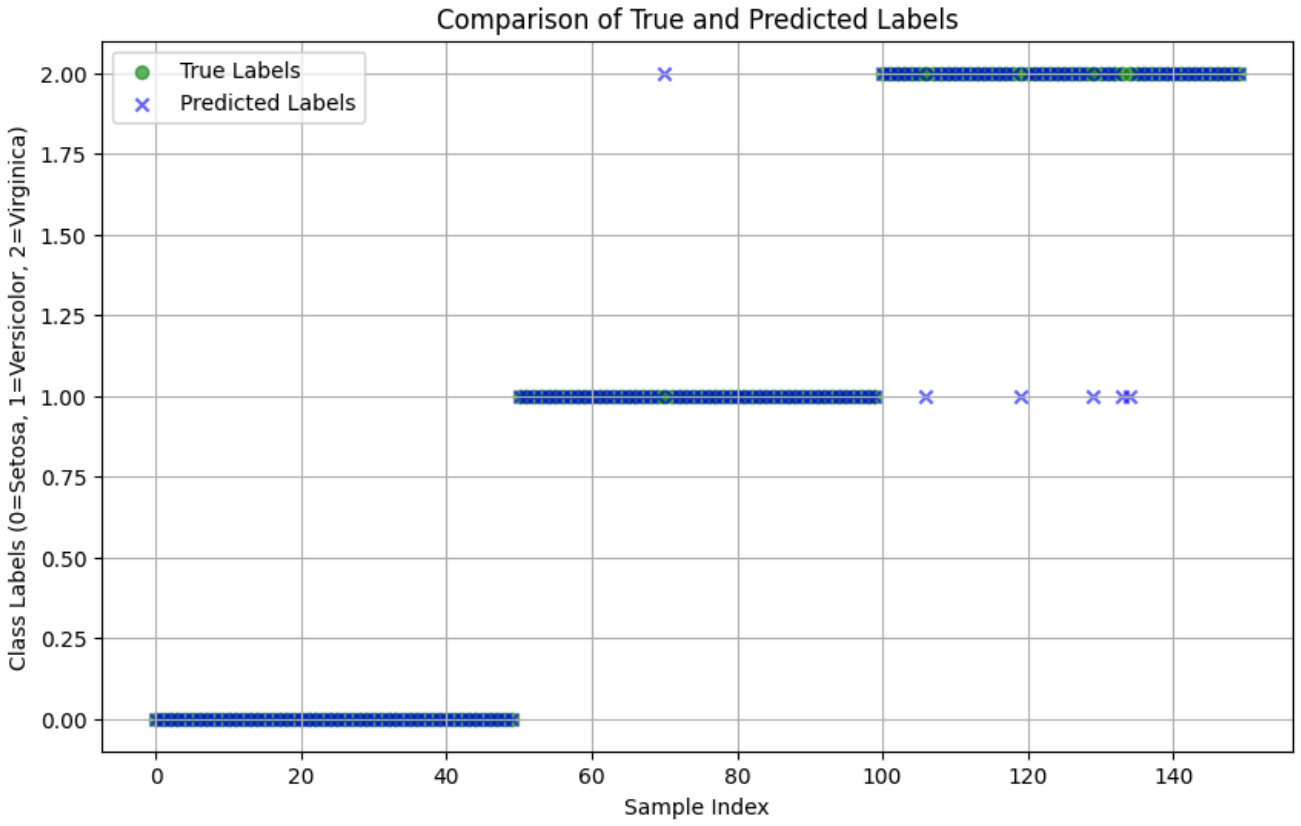


Then we used the whole iris dataset to do the classification based on 3.2.1.

Predicted label:



Classification result:



Total samples: 150
Number of misclassifications: 6
Accuracy: 96.00%

In general, the accuracy for the method proposed in Task 2 can be around 90%.

4. Discussion

4.1 Analysis of Model Performance

4.1.1 More features?

In Task 1, we only used 1 feature `petal.length` and in Task 2, we used 2 features `petal.length` and `petal.width`. Theoretically, we could use all 4 features together to train the model and still get ideal results but as discussed in 3.2.3, the trained model could be more unstable and would require a well-designed use of a valid set. Simplicity proved effective, achieving perfect accuracy.

4.1.2 Alternative Model Forms

In this report, we used the $y = wX + b$ model. In Task 2, we initially thought and attempted to use $z = a \cdot (x + y)^2 + b \cdot (x + y) + c$ to do the linear regression, especially since `Versicolor` occupies a middle ground between the other two classes. It turns out that this kind of model can be very accurate when it's well-trained, but it's also very hard to train it right.

This highlighted the importance of model simplicity—not only does it make the training process more manageable, but it also contributes to better generalization in some cases.

4.2 Evaluation of Advantages and Disadvantages

4.2.1 Advantages

1. **Simplicity**: Linear Regression is straight forward to implement and understand
2. **Low Computational Cost**: Training Linear Regression models is inexpensive

4.2.2 Disadvantages

1. Linear regression is not inherently designed for classification tasks, leading to suboptimal decision boundaries(like what we observed in 2.2)
2. When the data distributions show non-linear properties, it will be hard to implement
3. For more categories to be classified, it needs more tricks and may results in limited performance.

4.3 General Notes for Improvement

Our findings suggest that while linear regression can be adapted for simple binary classification tasks with noticeably linear separable classes, it is not the most effective method for more multiclass classification problems.

Therefore, improvements could include:

1. Alternative Models: More classification centered algorithms such as logistic regression or decision trees could be used to acertain a better accuracy in classification
2. Feature Engineering: Investigating the impact of different feature combinations or engineered features to improve model performance

5. Conclusion

For the binary classification in Task 1, the model performed exceptionally well, achieving 100% accuracy by using the clear linear separation between Setosa and Versicolor based on `petal.length`.

However, in Task 2, when extending the approach to aall features `Setosa`, `Versicolor`, and `Virginica`, the limitations of linear regression became evident. Despite achieving an accuracy of up to 96%, the model required additional complexity, such as training multiple binary classifiers to handle the multiclass scenario.

The challenges in capturing non-linear relationships in this task, highlighted that linear regression is not ideally suited for such tasks.

Overall, this project provided valuable insights into the capabilities and limitations of linear regression. It reinforced the importance of aligning the choice of algorithm with the specific requirements of the task and the characteristics of the dataset. By evaluating our approach and results, we gained a deeper understanding of machine learning principles. It was fun.