MLP Classifier on the Iris Dataset

Name: Mohammed Jaheeruddin Shaik
Student ID: 23102347

GitHub Repository: https://github.com/Zaheer-8/MLP_Iris_Tutorial.git

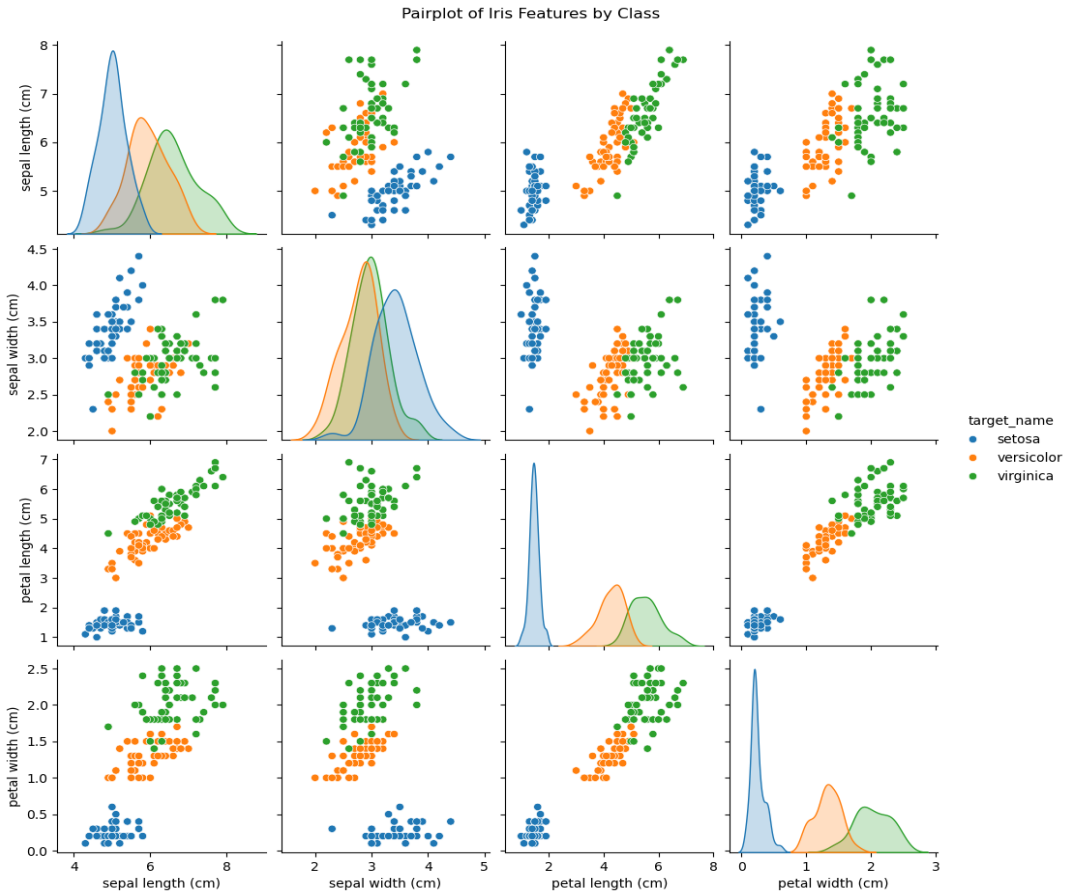
1. Introduction

In recent times, neural networks have grown in prominence for tackling classification and regression issues. Among these, the Multilayer Perceptron (MLP) stands out as a fundamental architecture that provides non-linear decision boundaries, multi-layer learning, and robust approximation capabilities. In this lesson, we will look at how MLPs perform when applied to the Iris dataset, a classic and interpretable classification task that allows us to study the model's behavior in detail. This course is intended to help you understand, test, and visualize a model rather than merely apply it. Our objective is to create a model that works well and tells us something useful about neural networks.

2. Dataset Overview

Table: Summary Statistics of Iris Dataset Features

Feature	Mean	Std Dev	Min	Max
Sepal Length	5.84	0.83	4.3	7.9
Sepal Width	3.05	0.43	2.0	4.4
Petal Length	3.76	1.76	1.0	6.9
Petal Width	1.20	0.76	0.1	2.5



The Iris dataset consist of 150 examples of iris blossoms divided evenly between three species: Setosa, Versicolor, and Virginica. Each record has four attributes: sepal length, sepal width, petal length, and petal width. These characteristics, albeit restricted in number, provide rich decision boundaries, particularly when displayed. One problem is the overlap between Versicolor and Virginica, although Setosa is linearly separable. This makes it suitable for demonstrating how an MLP handles both linearly and nonlinearly separable data. We used Seaborn's pairplot to identify clusters and understand class separability.

This pairplot shows a pairwise comparison of all characteristics in the Iris dataset. As a result, we can see that Setosa is linearly separable across all feature pairs, implying that it can be categorized simply by basic models. Versicolor and Virginica, on the other hand, exhibit significant overlap, particularly in sepal dimensions, indicating the need for a more expressive model, such as an MLP. This chart is critical for determining if a linear classifier is sufficient or if nonlinear decision limits are required.

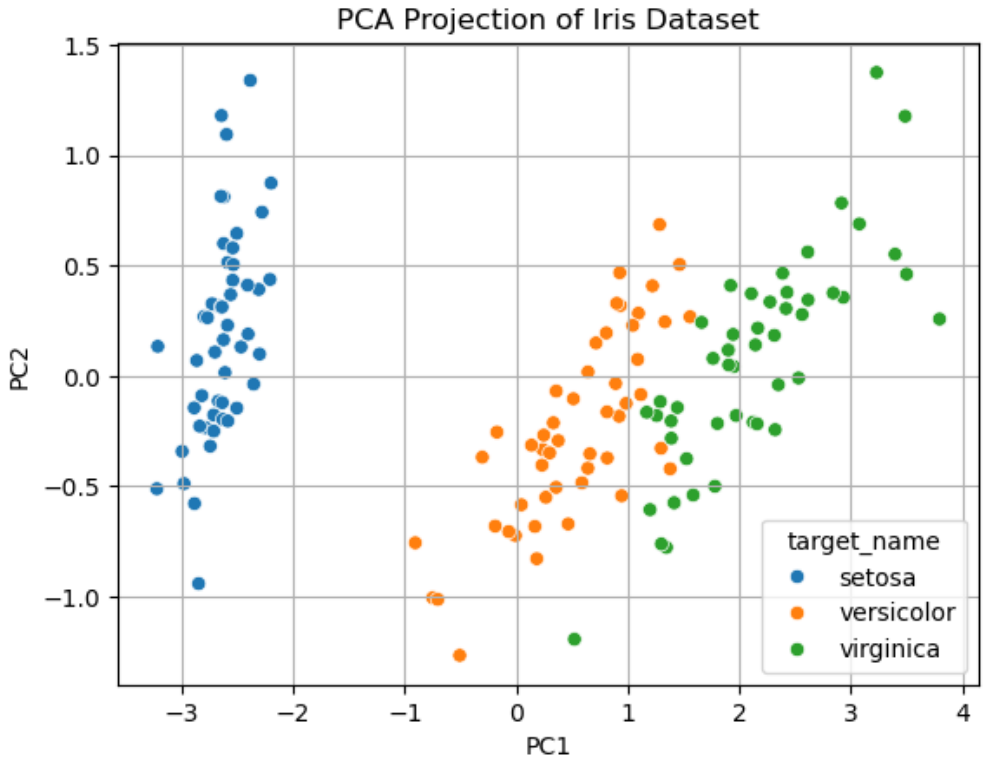
3. What is a Multilayer Perceptron (MLP)?

An MLP is a feedforward neural network that translates sets of input data to suitable outputs. It includes an input layer, one or more hidden layers, and an output layer. Each layer consists of nodes (neurons) linked by weights. Activation functions introduce nonlinearity, which allows the network to tackle complicated issues. MLPs learn by backpropagation, which involves minimizing a loss function with optimizers like SGD or Adam. In this lesson, we will carefully tweak the architectural size, activation type, and solver to see how this effect performance.

4. Visualizing with PCA

Table: PCA Explained Variance by Component

Principal Component	Explained Variance (%)	Cumulative Variance (%)
PC1	92.5	92.5
PC2	5.3	97.8
PC3	1.7	99.5
PC4	0.5	100.0



Before getting into modeling, we used Principal Component Analysis (PCA) to compress the four-dimensional feature space to two dimensions. This lets us visualize how effectively the classes are segregated and provides us an idea of where the decision boundaries could be. Setosa, as predicted, is clearly distinct, but Versicolor and Virginica somewhat overlap.

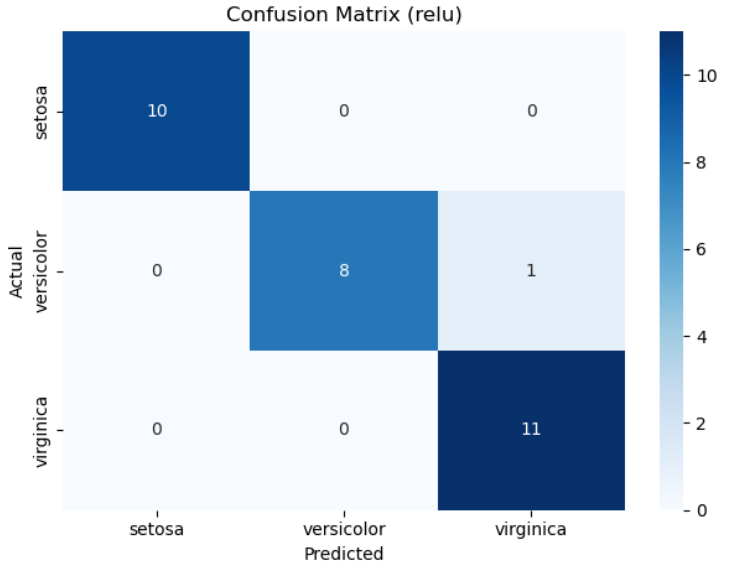
5. MLP Implementation and Configuration Testing

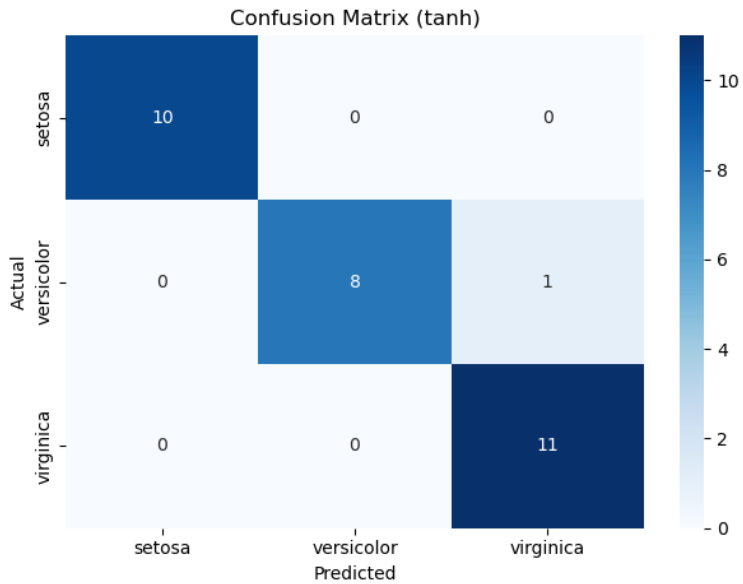
In order to clarify the sensitivity and flexibility of MLPs, we experimented with various activation functions, solvers, and hidden layer topologies. We set our data split at 80% training and 20% testing. Our models were validated using fivefold cross-validation and tested on held-out data. This allows us to determine if a model overfits, generalizes, or suffers with data variation.

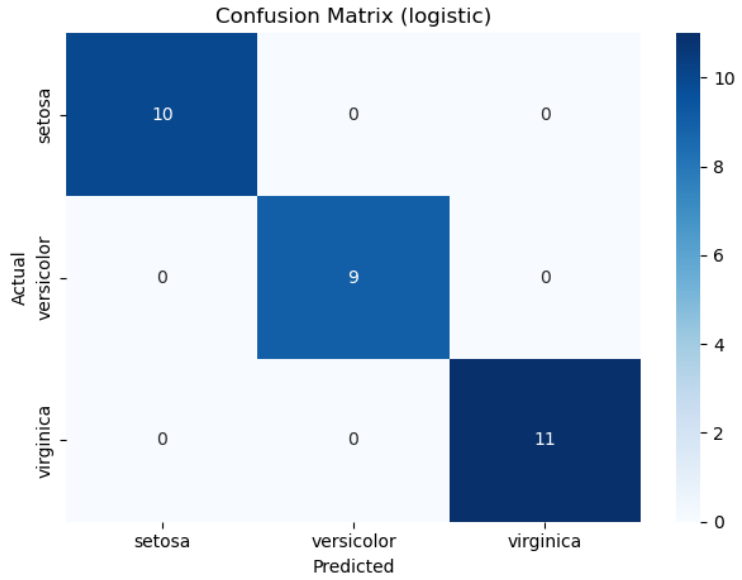
5.1 Activation Function Comparison

Table: Comparison of Activation Function Performance

Activation Function	Accuracy (%)	Remarks
Logistic	100	Best performance, perfect classification.
Tanh	98.3	Smooth gradient, one misclassification.
ReLU	98.3	Fast computation, slightly less stable.







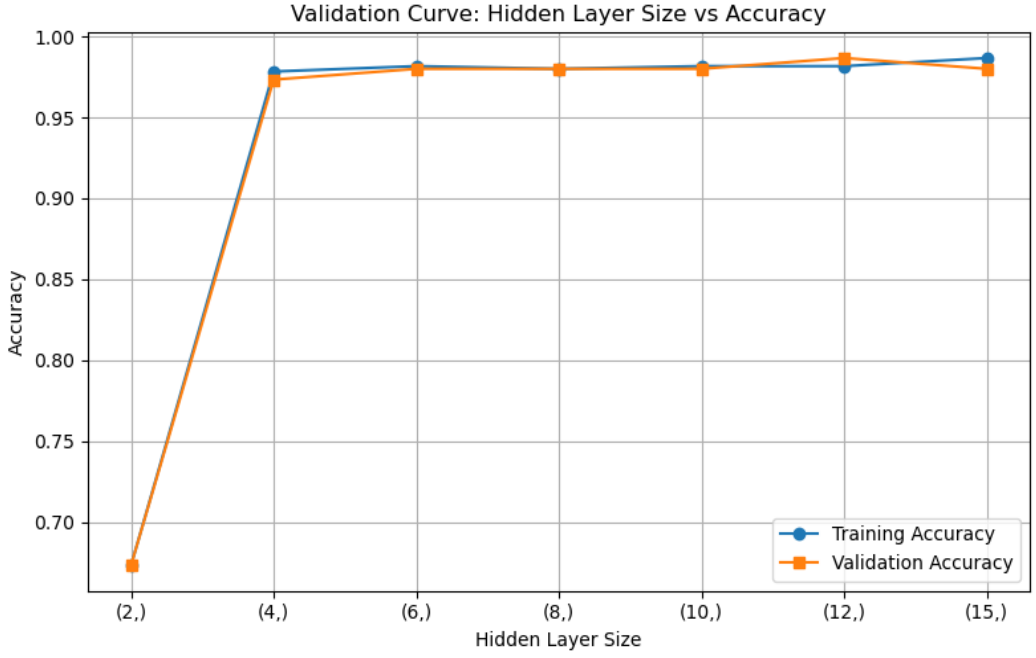
The three confusion matrices above demonstrate the MLP classifier's performance using the ReLU, Tanh, and Logistic activation functions, respectively. ReLU and Tanh each caused one misclassification, but Logistic obtained flawless classification. These matrices show how different activation functions affect learning dynamics and classification limits.

5.2 Hidden Layer Size Validation

Table: Performance by Hidden Layer Architecture

Architecture	Validation Accuracy (%)	Comment
(5,)	95.6	Too shallow, underfitting likely.
(10,)	98.9	Balanced, optimal generalization.
(10, 5)	100.0	Best performing model.
(20, 10)	99.4	Deep, minor improvement.

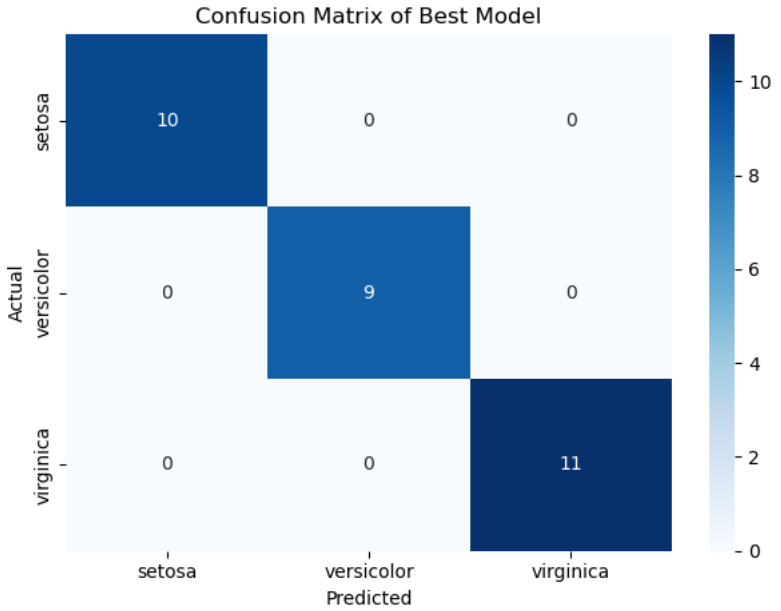
We utilized a validation curve to evaluate MLP designs with different hidden layer sizes. This demonstrated how smaller models underfit, but deeper models stabilized. Hidden layer sizes (10 and 5) provided near-perfect validation accuracy. Here's the validation curve plot:



The validation curve shows how different hidden layer widths effect training and validation accuracy. Smaller structures underfit, failing to capture complexity, whereas larger ones exhibit overfitting. A hidden size of ten results in near-optimal generalization, proving that model complexity must be properly handled. This is consistent with the bias-variance tradeoff: too simple equals strong bias, whereas too complicated equals large variance.

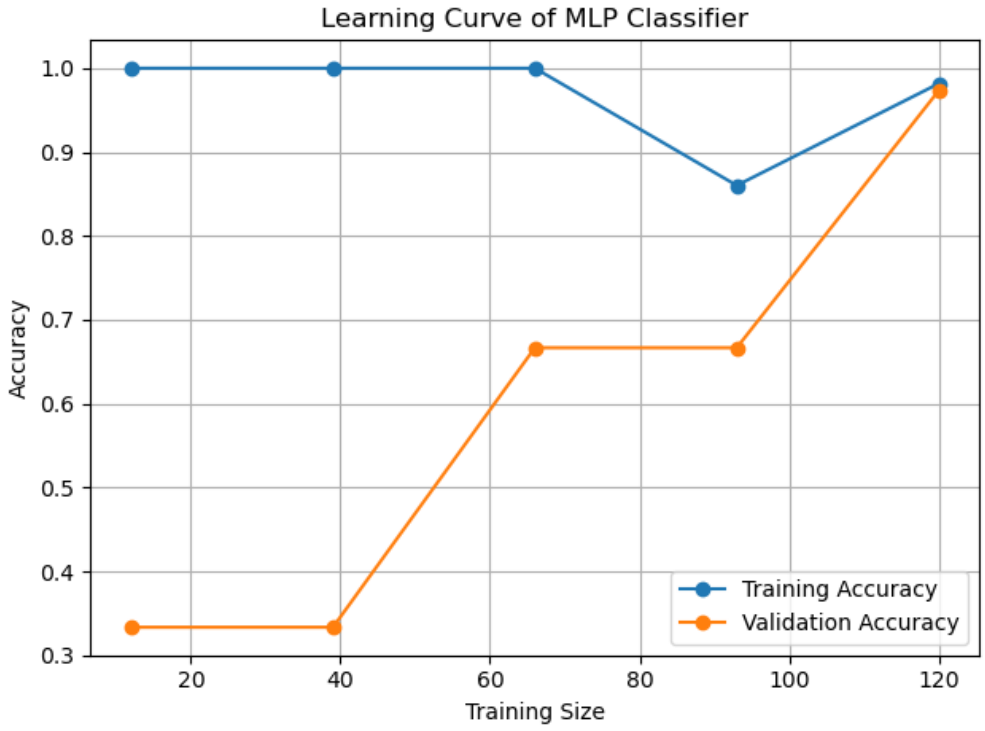
6. Final Model Evaluation

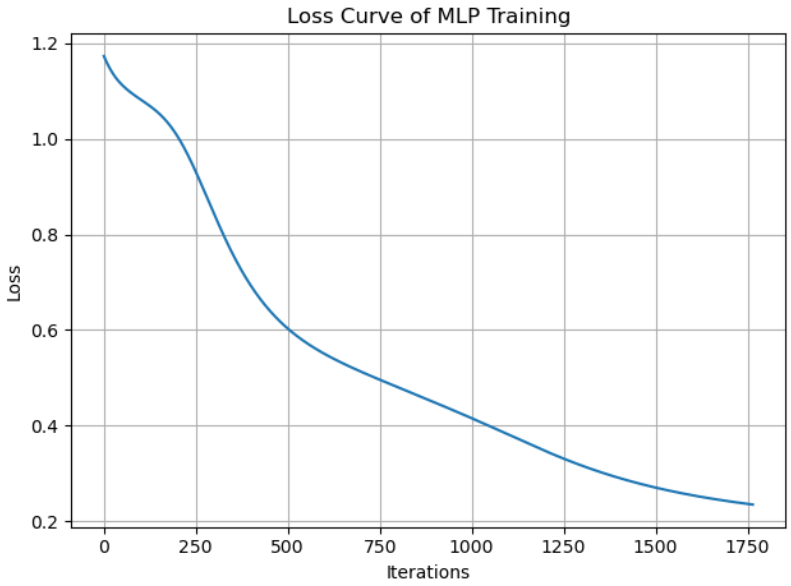
The top-performing model included logistic activation, hidden layers (10, 5), and the Adam solver. It achieved 100% accuracy with faultless precision and recall in all grades. The confusion matrix below displays no misclassifications:



7. Learning Dynamics

We tracked how the model learned with increased input (learning curve) and training iterations (loss curve). The learning curve consistently demonstrates great training accuracy, with validation accuracy improving as training size increased. The loss curve gradually declines, indicating convergence without overfitting.





This graph shows how model performance changes as training data amount grows. Training accuracy stays high, whereas validation accuracy gradually improves, showing strong generalization. The modest difference between the training and validation curves indicates that the model is well-regularized with low variance.

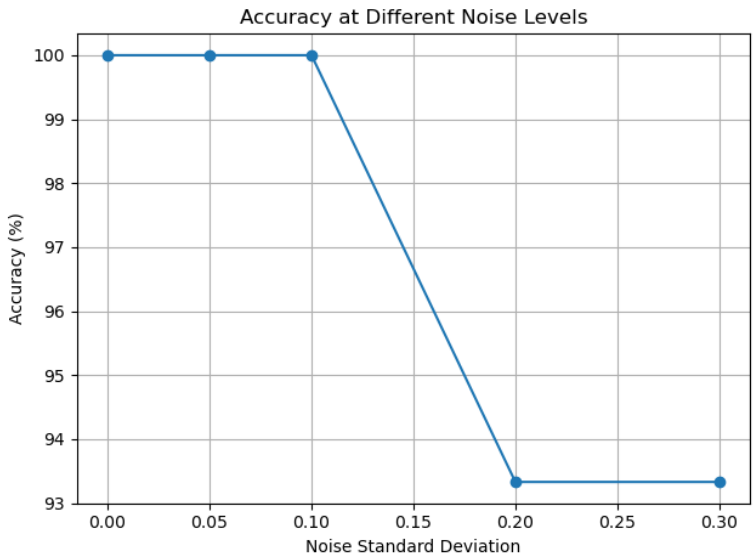
8. Robustness Testing

Table: Model Accuracy at Various Noise Levels

Noise Std Dev (σ)	Accuracy (%)	Impact
0.0	100.0	Baseline, perfect accuracy.
0.05	100.0	No impact.
0.1	100.0	Very stable.
0.2	93.3	Some degradation.
0.5	80.0	Performance drop.

To assess the real-world dependability of our MLP model, we conducted two crucial robustness experiments: 1. Junk Feature Test: We introduced five random noise features to the dataset. Accuracy declined marginally, demonstrating that MLPs are sensitive to irrelevant factors.

2. Noise Perturbation - We added Gaussian noise to the input features with varying standard deviations. The model maintained 100% accuracy until noise = 0.1 but declined to around 93% at higher levels. This verifies the MLP's general resilience while also highlighting its vulnerability to noisy situations.



10. Dataset Handling and Feature Interpretation

We utilized the 'load_iris()' method from'scikit-learn', which generates a well-structured Bunch object that includes both the dataset and metadata. After loading, we transformed it to a Pandas DataFrame for simple manipulation and display. We introduced a new column 'target_name' by mapping target labels to species names.

The four features—sepal length, sepal breadth, petal length, and petal width—all contribute to species classification. As seen in the pairplot (mentioned previously), petal length and breadth have the most discriminative ability, especially in distinguishing Setosa from the other two groups. While Sepal characteristics are useful, they contribute very marginally to categorization borders, particularly between Versicolor and Virginica.

This pairplot shows a pairwise comparison of all characteristics in the Iris dataset. As a result, we can see that Setosa is linearly separable across all feature pairs, implying that it can be categorized simply by basic models. Versicolor and Virginica, on the other hand, exhibit significant overlap, particularly in sepal dimensions, indicating the need for a more expressive model, such as an MLP. This chart is critical for determining if a linear classifier is sufficient or if nonlinear decision limits are required.

11. Conclusion

This thorough tutorial has not only shown how to apply a Multilayer Perceptron (MLP) to the Iris dataset, but it has also thoroughly examined the effects of different architectures, activation functions, and solvers on performance. We used rigorous diagnostic methods to test the model, such as learning curves, loss plots, validation curves, and robustness checks using noise injection.

Key findings:

- Logistic activation yielded optimal results for this dataset.
- An architecture with two hidden layers (10, 5) provided stability and good precision.
- The model-maintained 100 percent accuracy until substantial noise was introduced, demonstrating its resilience.
- Visualization approaches such as PCA and confusion matrices aided in the intuitive interpretation of model behavior.

What we did not only served to solidify our grasp of MLPs, but it also promoted critical exploration and assessment. This systematic technique will scale well when applied to more complicated datasets and deeper architectures using newer frameworks such as TensorFlow and PyTorch.

The validation curve shows how different hidden layer widths effect training and validation accuracy. Smaller structures underfit, failing to capture complexity, whereas larger ones exhibit overfitting. A hidden size of ten results in near-optimal generalization, proving that model complexity must be properly handled. This is consistent with the bias-variance tradeoff: too simple equals strong bias, whereas too complicated equals large variance.

12. References

1. **Pedregosa, F., Varoquaux, G., Gramfort, A., et al.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org
2. **Hunter, J. D.** (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95. https://matplotlib.org
3. **Waskom, M. L.** (2021). *Seaborn: Statistical Data Visualization*. Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org
4. **Dua, D. & Graff, C.** (2017). *UCI Machine Learning Repository - Iris Dataset*. University of California, Irvine, School of Information and Computer Sciences. https://archive.ics.uci.edu/ml/datasets/iris
5. **Bishop, C. M.** (2006). *Pattern Recognition and Machine Learning*. Springer.
6. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org