Task 1. Implementing a Perceptron with Different Activation Functions

1. Implement a Perceptron class in Python with support for choosing activation functions:
   - Sigmoid,
   - ReLU,
   - Tanh.
2. Train the perceptron on a real dataset, e.g., Iris (two classes only).
3. Visualize the decision boundary for each activation function.

Goals:

- Explore how activation functions affect performance.
- Explain why one activation function works better than others.

Outcomes:

- Write a report discussing differences in results between activation functions and their applicability to real-world tasks.

Task 2. Influence of Hyperparameters on Perceptron Performance

1. Using the make_classification dataset from sklearn, analyze the effect of the following hyperparameters:
   - Learning rate.
   - Number of epochs.
2. Plot accuracy against hyperparameter changes.
3. Optimize hyperparameters to achieve the best accuracy on the test set.

Goals:

- Learn to understand and tune model hyperparameters.
- Determine optimal settings for the data.

Task 3. Model Extension: Multiclass Classification

1. Modify the perceptron to handle multiclass classification using the "one-vs-rest" approach.
2. Use the Iris dataset (all three classes).
3. Evaluate the model's accuracy and visualize results in 2D space (e.g., PCA for dimensionality reduction).

Goals:

- Master techniques to scale models for multiclass classification.
- Apply visualization methods to present results.

Task 4. Experiments with Different Datasets

1. Test the perceptron on the following datasets:
   - sklearn.datasets.load_breast_cancer (classification task),
   - sklearn.datasets.make_moons (non-linearly separable data).
2. Analyze how the model handles linearly and non-linearly separable data.
3. Add a hidden layer for make_moons and explain how it improves performance.

Goals:

- Understand the limitations of linear perceptrons.
- Explore the impact of non-linear transformations on model quality.

Task 5. Study of Activation Functions

1. Plot activation functions (Sigmoid, ReLU, Tanh) and their derivatives.
2. Analyze their properties:
   - Value range,

- Smoothness,
- Gradient vanishing problem.

3. Explain which function is better suited for specific tasks and why.

Goals:

- Understand the mathematics behind activation functions.
- Evaluate their applicability to different tasks.

Task 6. Model Error Analysis

1. Compute and visualize a confusion matrix for a perceptron trained on a real dataset (e.g., load_breast_cancer).
2. Calculate metrics:

- Accuracy,
- Precision,
- Recall,
- F1-score.

3. Compare results using different activation functions.

Goals:

- Learn how to interpret model evaluation metrics.
- Identify model weaknesses and suggest improvements.

Task 7. Model Robustness to Noise

1. Add random noise to the training data:

- Add Gaussian noise to features,
- Change up to 10% of labels to the opposite class.

2. Analyze how the model performs on noisy data.
3. Visualize changes in the decision boundary before and after adding noise.

Goals:

- Investigate the perceptron's robustness to noisy data.
- Understand how data quality affects model performance.

Task 8. Data Normalization Experiment

1. Implement data preprocessing:

- Scaling (MinMaxScaler),
- Standardization (StandardScaler).

2. Train the perceptron with each normalization strategy.
3. Compare results of the model with and without normalization.

Goals:

- Understand the importance of data preprocessing.
- Study the effect of normalization and standardization on model training.

Task 9. Impact of Feature Dimensionality

1. Create synthetic data using make_classification:

- Number of features: 2, 10, 50.
- Number of informative features: 2.

2. Train the perceptron on data with varying dimensions.
3. Compare model accuracy and explain how data dimensionality affects results.

Goals:

- Recognize the "curse of dimensionality."
- Understand how feature selection impacts model training.

Task 10. Comparison of Optimizers for Training Perceptrons

1. Implement the perceptron model and train it using the following optimizers:
- Stochastic Gradient Descent (SGD),
- Adam,
- RMSprop.
2. Use a real dataset, such as load_breast_cancer, to compare the performance of each optimizer.
3. Evaluate and plot the following:
- Training loss over epochs,
- Model accuracy on the validation set.

Goals:

- Understand the differences between optimizers and their impact on model training.
- Analyze which optimizer is better suited for different data distributions or tasks.