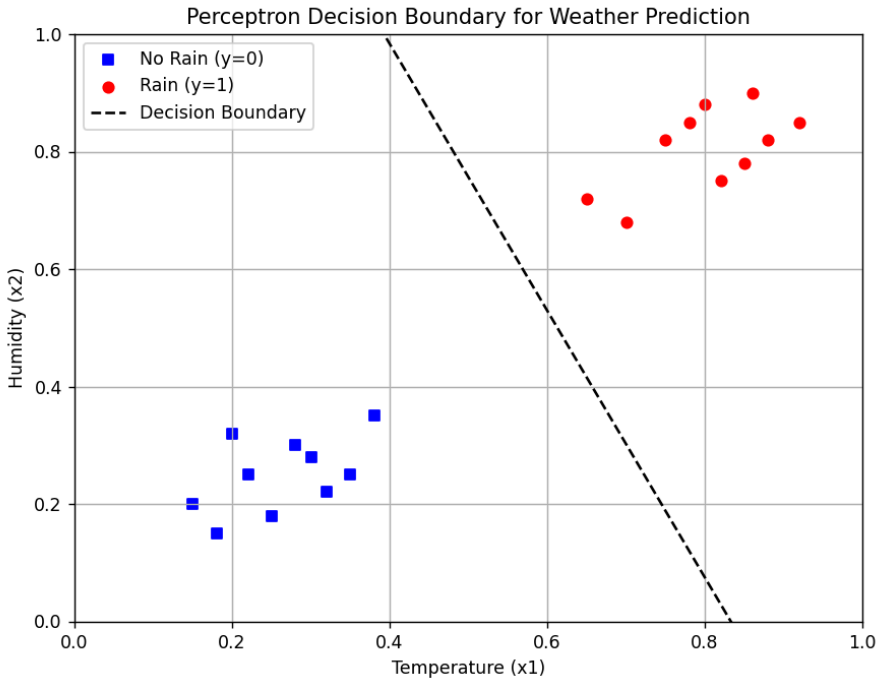
Does the perceptron separate the two classes (rain vs. no rain)?
>Yes the perceptron model successfully separates the two calls (rain vs no rain) in the given dataset. It converges just after a few iterations and achieves 100% accuracy on both training and testing data.

Based on the dataset's pattern, explain why or why not the perceptron works well (or fails).
> The perceptron does work well as the dataset has a clear separation between the points where it rains, and the points where it does not rain. The dataset achieved 100% accuracy on the perceptron, meaning that the dataset is linearly separable and a decision boundary can separate the two classes “Rain” and “No Rain”.

Suggest possible improvements (e.g., using a different model or feature engineering).
>in cases where the perceptron deals with more complex data, a non-linear classifier, such as a multi-layer perceptron (MLP), could capture more complex patterns. Interaction terms and hierarchical order could improve performance as well.

Plot the decision boundary on the same graph as your data points
>Code found in q3_c_decision_boundary.py



Experiment with at least two different train/test splits and compare the results
Original split
train = 15
Test = 5

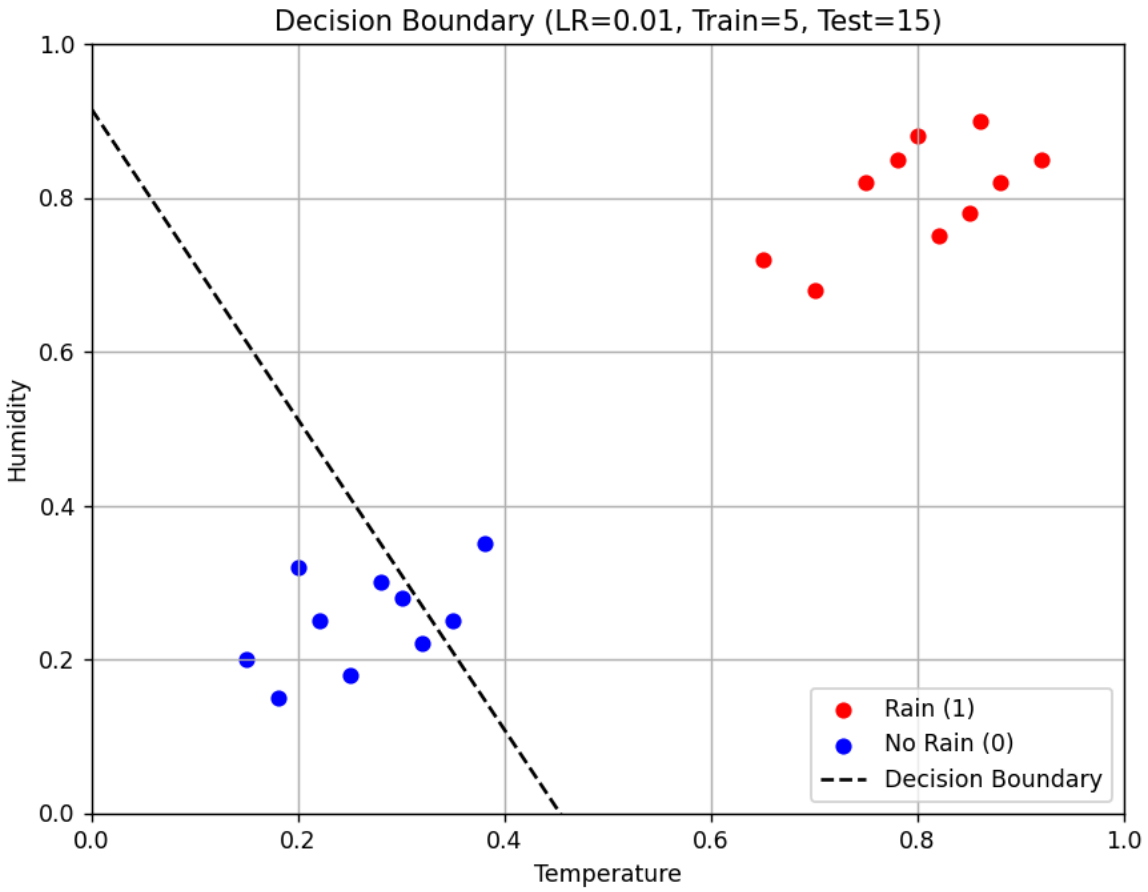
Experiment 1 (50% train, 50% test)

--- Training with Learning Rate = 0.01, Train size = 5, Test size = 15 ---

Converged after 6 iterations.

Train Accuracy: 100.00%

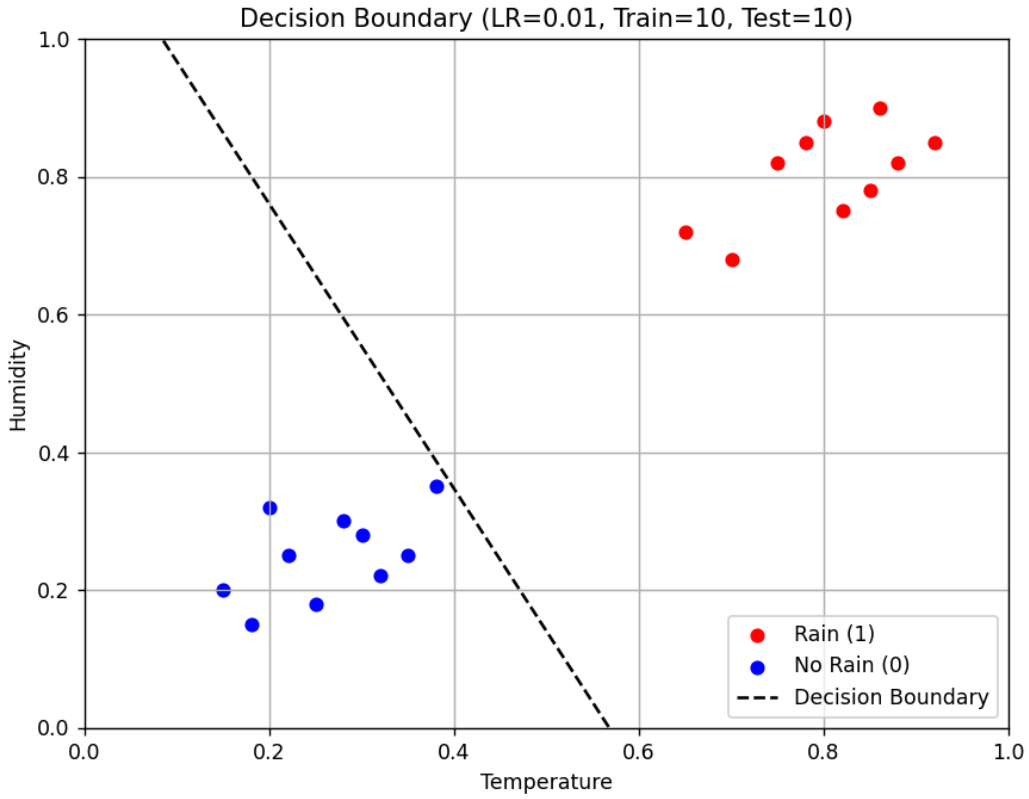
Test Accuracy: 86.67%



:

Experiment 2

--- Training with Learning Rate = 0.01, Train size = 10, Test size = 10 ---
Converged after 5 iterations.
Train Accuracy: 100.00%
Test Accuracy: 100.00%



Using 1000 iterations to train the model

In the model training it appears the the test accuracy goes as the amount of instances increases whereas the training set seems to be consistent with 100% accuracy regardless of the value assigned to it.

Discuss how different learning rates might affect the model's performance
>Learning rates that are too small can affect the perceptron because it leads to a very gradual solution which is much slower when aiming to find the optimal solution. Learning rates which are higher can aid in speeding up the learning process by making larger updates to the weights, however, a larger learning rate can cause the model to overshoot the optimal weights, missing the best solution. Each model performance is different but it's often best to take both ranges into account and shoot for a learning rate somewhere in the middle if the solution is not yet known.