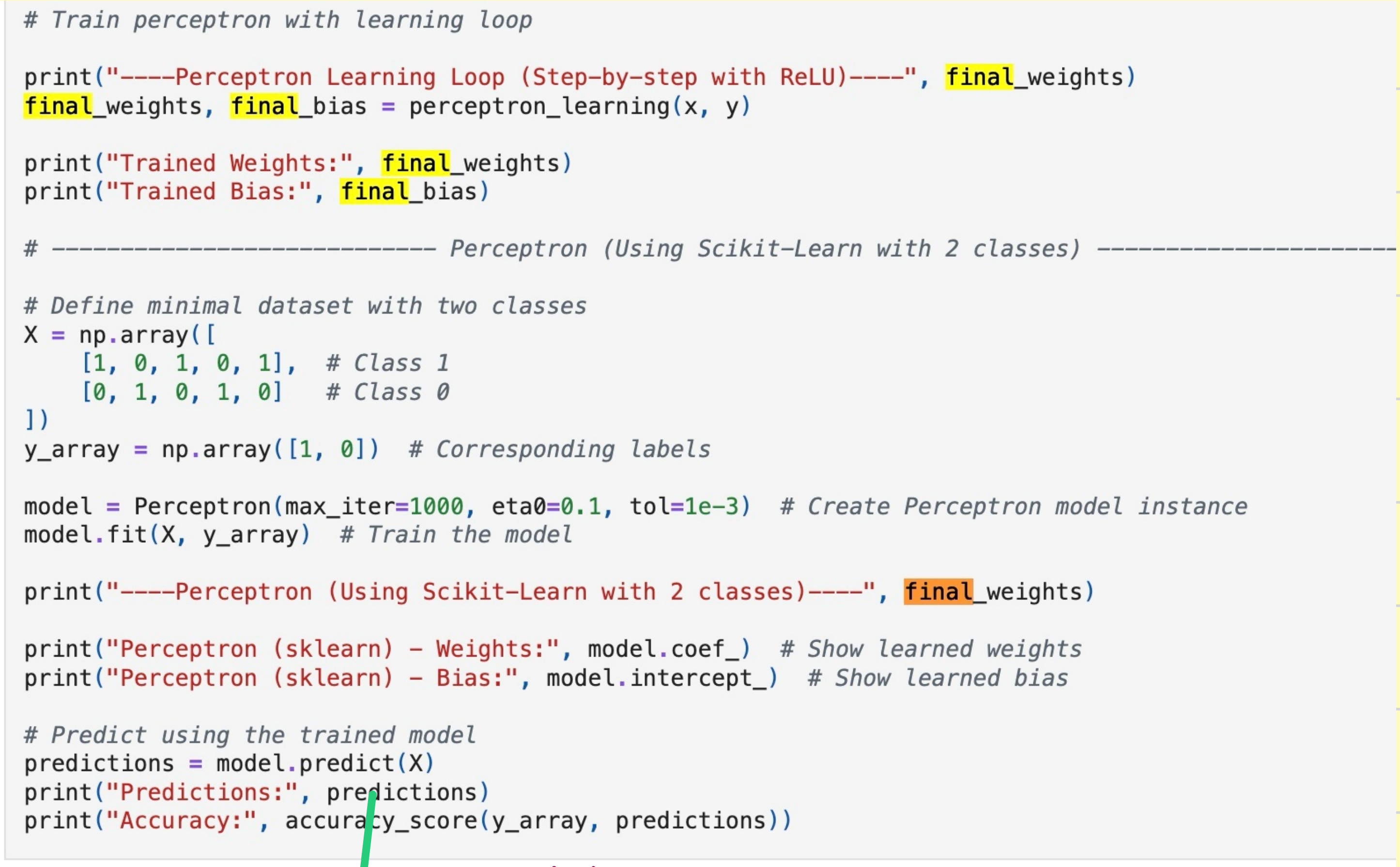
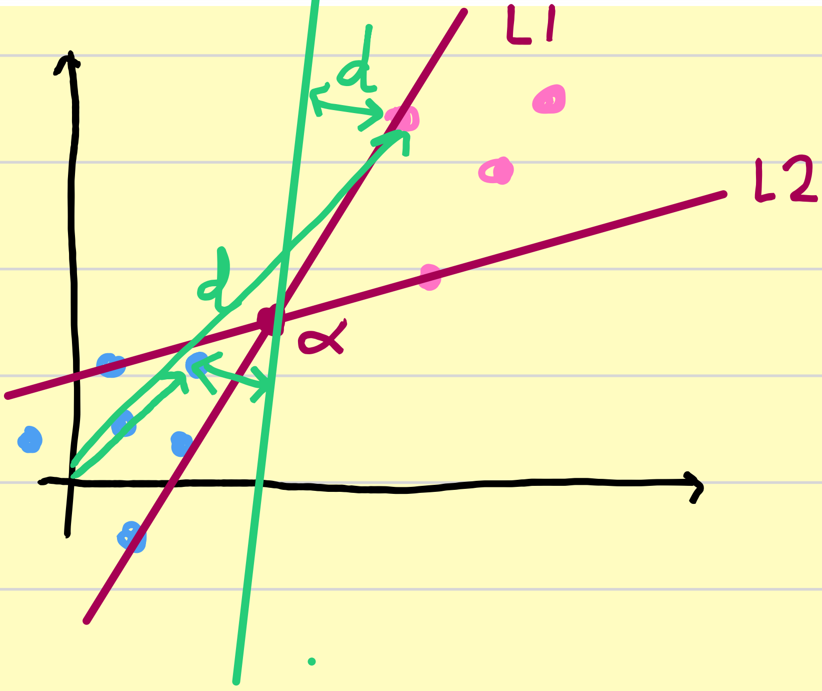
20250604_WIN2_Statistik



```
# Train perceptron with learning loop

print("----Perceptron Learning Loop (Step-by-step with ReLU)----", final_weights)
final_weights, final_bias = perceptron_learning(x, y)

print("Trained Weights:", final_weights)
print("Trained Bias:", final_bias)

# ---------------------------- Perceptron (Using Scikit-Learn with 2 classes) ---------------------

# Define minimal dataset with two classes
X = np.array([
    [1, 0, 1, 0, 1],  # Class 1
    [0, 1, 0, 1, 0]   # Class 0
])
y_array = np.array([1, 0])  # Corresponding labels

model = Perceptron(max_iter=1000, eta0=0.1, tol=1e-3)  # Create Perceptron model instance
model.fit(X, y_array)  # Train the model

print("----Perceptron (Using Scikit-Learn with 2 classes)----", final_weights)

print("Perceptron (sklearn) - Weights:", model.coef_)  # Show learned weights
print("Perceptron (sklearn) - Bias:", model.intercept_)  # Show learned bias

# Predict using the trained model
predictions = model.predict(X)
print("Predictions:", predictions)
print("Accuracy:", accuracy_score(y_array, predictions))
```



20250416

$$\text{KOV MAT}\left[X_1, X_2, X_3, \dots, X_n\right] = A = \begin{bmatrix} \mathrm{VAR}(X_1) & \mathrm{KOV}[X_1, X_2] & \dots & \mathrm{KOV}[X_1, X_n] \\ \mathrm{KOV}[X_1, X_2] & \mathrm{VAR}[X_2] & & \mathrm{KOV}[X_2, X_n] \\ \vdots & & & \\ \mathrm{KOV}[X_1, X_n] & \mathrm{KOV}[X_2, X_n] & & \mathrm{VAR}[X_n] \end{bmatrix}$$