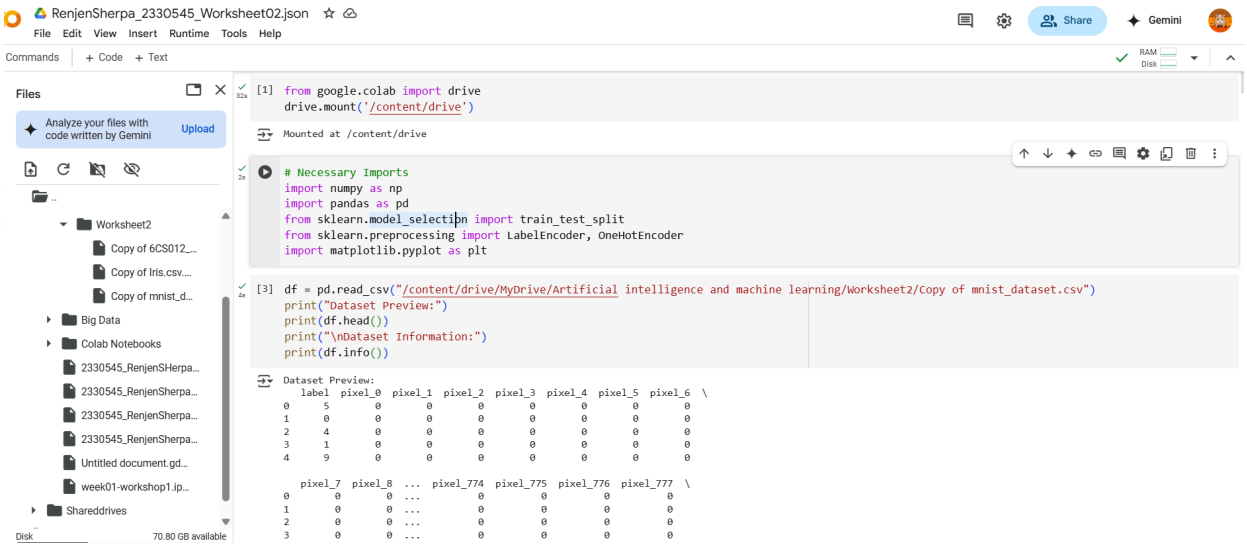
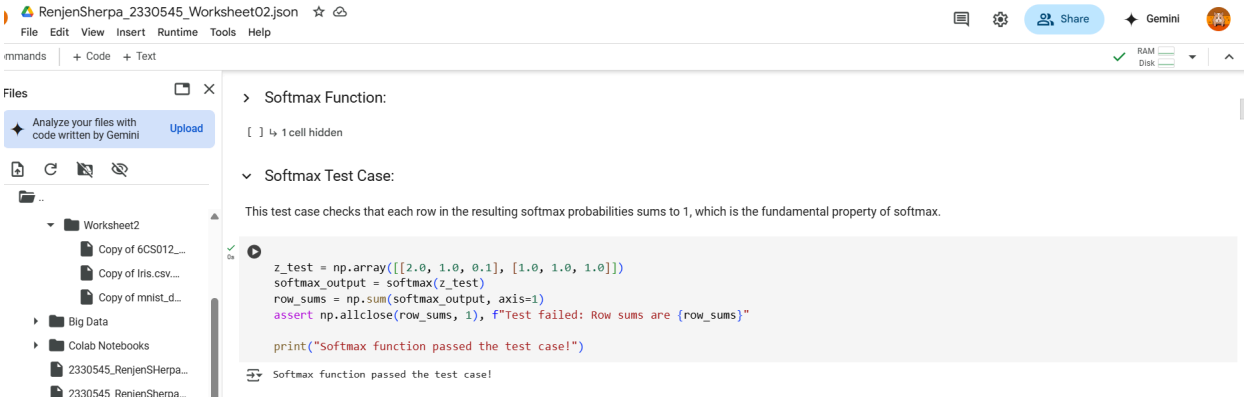
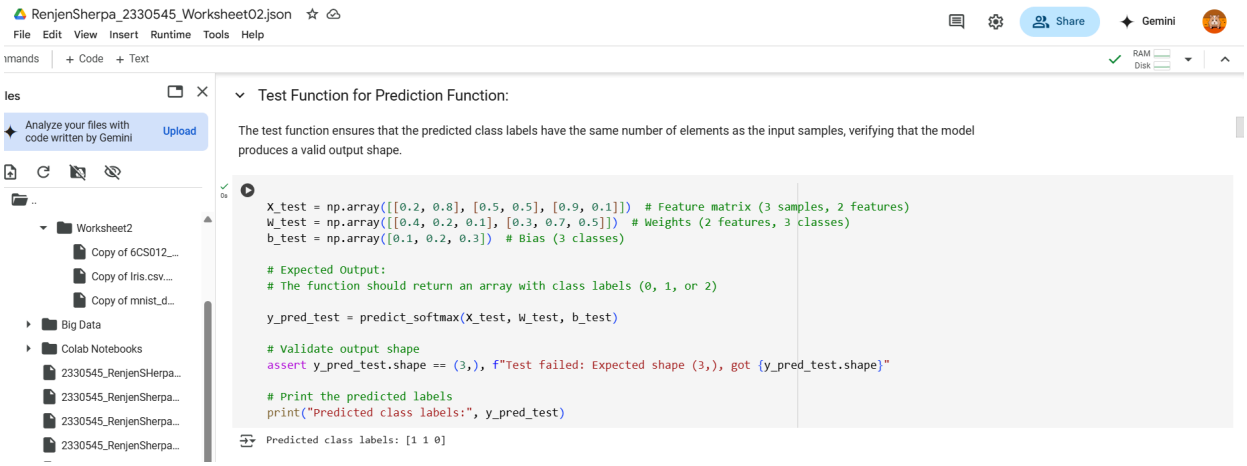
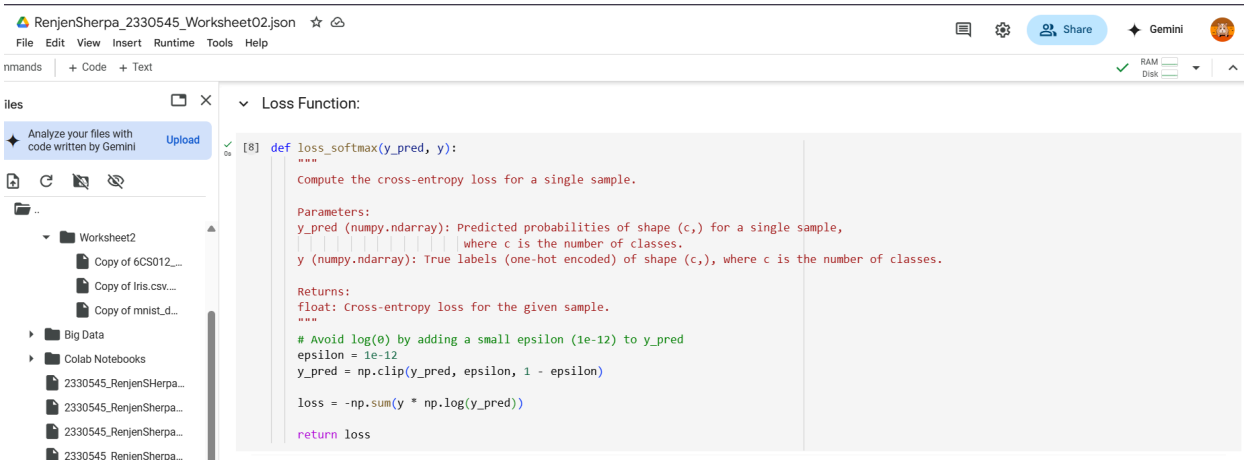
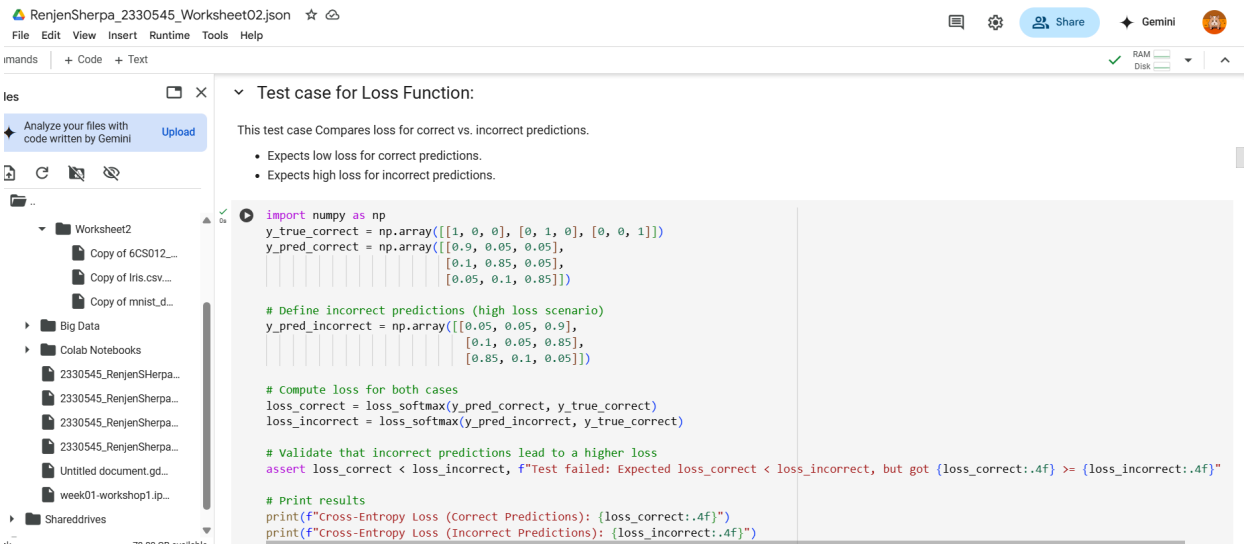
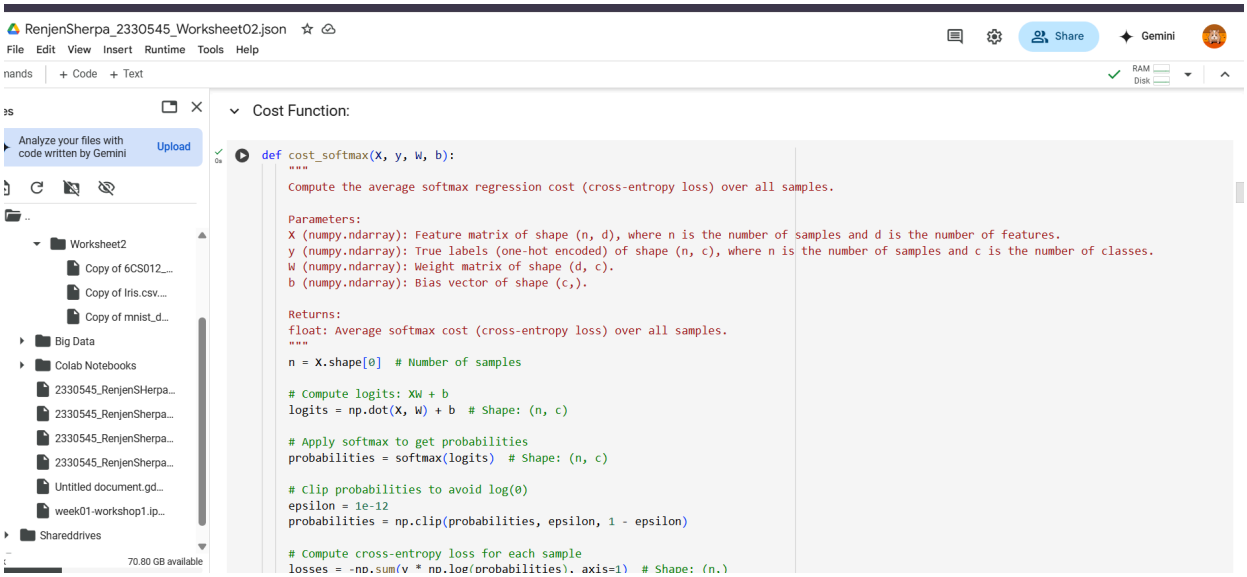

RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Files
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives
Disk
70.80 GB available
[1] from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
# Necessary Imports
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import matplotlib.pyplot as plt
[3] df = pd.read_csv("/content/drive/MyDrive/Artificial intelligence and machine learning/Worksheet2/Copy of mnist_dataset.csv")
print("Dataset Preview:")
print(df.head())
print("\nDataset Information:")
print(df.info())
Dataset Preview:
label pixel_0 pixel_1 pixel_2 pixel_3 pixel_4 pixel_5 pixel_6 \
0 5 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0
2 4 0 0 0 0 0 0 0
3 1 0 0 0 0 0 0 0
4 9 0 0 0 0 0 0 0
pixel_7 pixel_8 ... pixel_774 pixel_775 pixel_776 pixel_777 \
0 0 0 ... 0 0 0 0
1 0 0 ... 0 0 0 0
2 0 0 ... 0 0 0 0
3 0 0 ... 0 0 0 0


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text
Files
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
Softmax Function:
[ ] ↳ 1 cell hidden
Softmax Test Case:
This test case checks that each row in the resulting softmax probabilities sums to 1, which is the fundamental property of softmax.
z_test = np.array([[2.0, 1.0, 0.1], [1.0, 1.0, 1.0]])
softmax_output = softmax(z_test)
row_sums = np.sum(softmax_output, axis=1)
assert np.allclose(row_sums, 1), f"Test failed: Row sums are {row_sums}"
print("Softmax function passed the test case!")
Softmax function passed the test case!


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Test Function for Prediction Function:
The test function ensures that the predicted class labels have the same number of elements as the input samples, verifying that the model produces a valid output shape.
X_test = np.array([[0.2, 0.8], [0.5, 0.5], [0.9, 0.1]]) # Feature matrix (3 samples, 2 features)
W_test = np.array([[0.4, 0.2, 0.1], [0.3, 0.7, 0.5]]) # Weights (2 features, 3 classes)
b_test = np.array([0.1, 0.2, 0.3]) # Bias (3 classes)
# Expected Output:
# The function should return an array with class labels (0, 1, or 2)
y_pred_test = predict_softmax(X_test, W_test, b_test)
# Validate output shape
assert y_pred_test.shape == (3,), f"Test failed: Expected shape (3,), got {y_pred_test.shape}"
# Print the predicted labels
print("Predicted class labels:", y_pred_test)
Predicted class labels: [1 1 0]


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
Loss Function:
[8] def loss_softmax(y_pred, y):
"""
Compute the cross-entropy loss for a single sample.
Parameters:
y_pred (numpy.ndarray): Predicted probabilities of shape (c,) for a single sample,
where c is the number of classes.
y (numpy.ndarray): True labels (one-hot encoded) of shape (c,), where c is the number of classes.
Returns:
float: Cross-entropy loss for the given sample.
"""
# Avoid log(0) by adding a small epsilon (1e-12) to y_pred
epsilon = 1e-12
y_pred = np.clip(y_pred, epsilon, 1 - epsilon)
loss = -np.sum(y * np.log(y_pred))
return loss
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
Test case for Loss Function:
This test case Compares loss for correct vs. incorrect predictions.
Expects low loss for correct predictions.
Expects high loss for incorrect predictions.
import numpy as np
y_true_correct = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
y_pred_correct = np.array([[0.9, 0.05, 0.05],
[0.1, 0.85, 0.05],
[0.05, 0.1, 0.85]])
# Define incorrect predictions (high loss scenario)
y_pred_incorrect = np.array([[0.05, 0.05, 0.9],
[0.1, 0.05, 0.85],
[0.85, 0.1, 0.05]])
# Compute loss for both cases
loss_correct = loss_softmax(y_pred_correct, y_true_correct)
loss_incorrect = loss_softmax(y_pred_incorrect, y_true_correct)
# Validate that incorrect predictions lead to a higher loss
assert loss_correct < loss_incorrect, f"Test failed: Expected loss_correct < loss_incorrect, but got {loss_correct:.4f} >= {loss_incorrect:.4f}"
# Print results
print(f"Cross-Entropy Loss (Correct Predictions): {loss_correct:.4f}")
print(f"Cross-Entropy Loss (Incorrect Predictions): {loss_incorrect:.4f}")
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
Cost Function:
def cost_softmax(X, y, W, b):
"""
Compute the average softmax regression cost (cross-entropy loss) over all samples.
Parameters:
X (numpy.ndarray): Feature matrix of shape (n, d), where n is the number of samples and d is the number of features.
y (numpy.ndarray): True labels (one-hot encoded) of shape (n, c), where n is the number of samples and c is the number of classes.
W (numpy.ndarray): Weight matrix of shape (d, c).
b (numpy.ndarray): Bias vector of shape (c,).
Returns:
float: Average softmax cost (cross-entropy loss) over all samples.
"""
n = X.shape[0] # Number of samples
# Compute logits: XW + b
logits = np.dot(X, W) + b # Shape: (n, c)
# Apply softmax to get probabilities
probabilities = softmax(logits) # Shape: (n, c)
# Clip probabilities to avoid log(0)
epsilon = 1e-12
probabilities = np.clip(probabilities, epsilon, 1 - epsilon)
# Compute cross-entropy loss for each sample
losses = -np.sum(y * np.log(probabilities), axis=1) # Shape: (n,)
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives
70.80 GB available

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

```
[10]  # Compute cross-entropy loss for each sample
      losses = -np.sum(y * np.log(probabilities), axis=1)  # Shape: (n,)

      # Compute total loss
      total_loss = np.sum(losses)

      # Return average loss
      return total_loss / n
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2
- Copy of 6CS012_...
- Copy of Iris.csv....
- Copy of mnist_d...

Big Data

Colab Notebooks
- 2330545_RenjenSHerpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- Untitled document.gd...
- week01-workshop1.ip...

Shareddrives

Test Case for Cost Function:

The test case assures that the cost for the incorrect prediction should be higher than for the correct prediction, confirming that the cost function behaves as expected.

```
import numpy as np

# Example 1: Correct Prediction (Closer predictions)
X_correct = np.array([[1.0, 0.0], [0.0, 1.0]])  # Feature matrix for correct predictions
y_correct = np.array([[1, 0], [0, 1]])  # True labels (one-hot encoded, matching predictions)
W_correct = np.array([[5.0, -2.0], [-3.0, 5.0]])  # Weights for correct prediction
b_correct = np.array([0.1, 0.1])  # Bias for correct prediction

# Example 2: Incorrect Prediction (Far off predictions)
X_incorrect = np.array([[0.1, 0.9], [0.8, 0.2]])  # Feature matrix for incorrect predictions
y_incorrect = np.array([[1, 0], [0, 1]])  # True labels (one-hot encoded, incorrect predictions)
W_incorrect = np.array([[0.1, 2.0], [1.5, 0.3]])  # Weights for incorrect prediction
b_incorrect = np.array([0.5, 0.6])  # Bias for incorrect prediction

# Compute cost for correct predictions
cost_correct = cost_softmax(X_correct, y_correct, W_correct, b_correct)

# Compute cost for incorrect predictions
cost_incorrect = cost_softmax(X_incorrect, y_incorrect, W_incorrect, b_incorrect)

# Check if the cost for incorrect predictions is greater than for correct predictions
assert cost_incorrect > cost_correct, f"Test failed: Incorrect cost {cost_incorrect} is not greater than correct cost {cost_correct}"
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2

```
[12]  # Print the costs for verification
      print("Cost for correct prediction:", cost_correct)
      print("Cost for incorrect prediction:", cost_incorrect)

      print("Test passed!")
```

```
Cost for correct prediction: 0.00062343643133349324
Cost for incorrect prediction: 0.29930861359446115
Test passed!
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2
- Copy of 6CS012_...
- Copy of Iris.csv....
- Copy of mnist_d...

Big Data

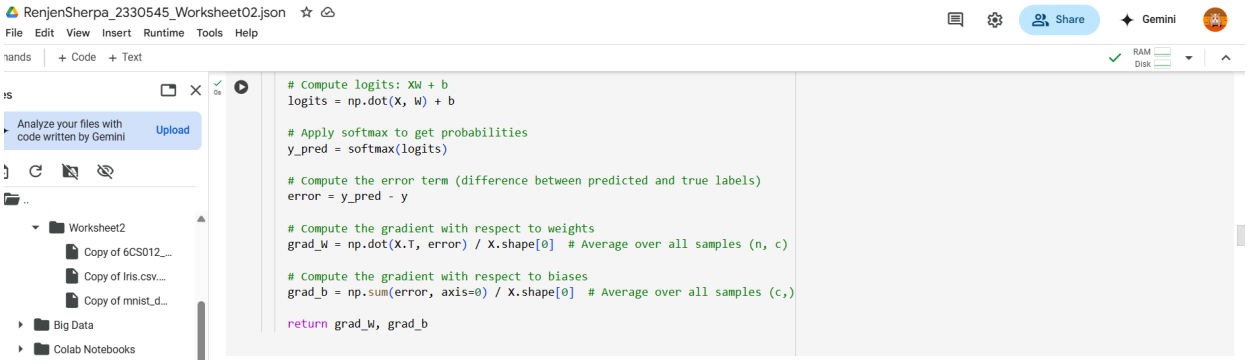
Colab Notebooks
- 2330545_RenjenSHerpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- Untitled document.gd...
- week01-workshop1.ip...

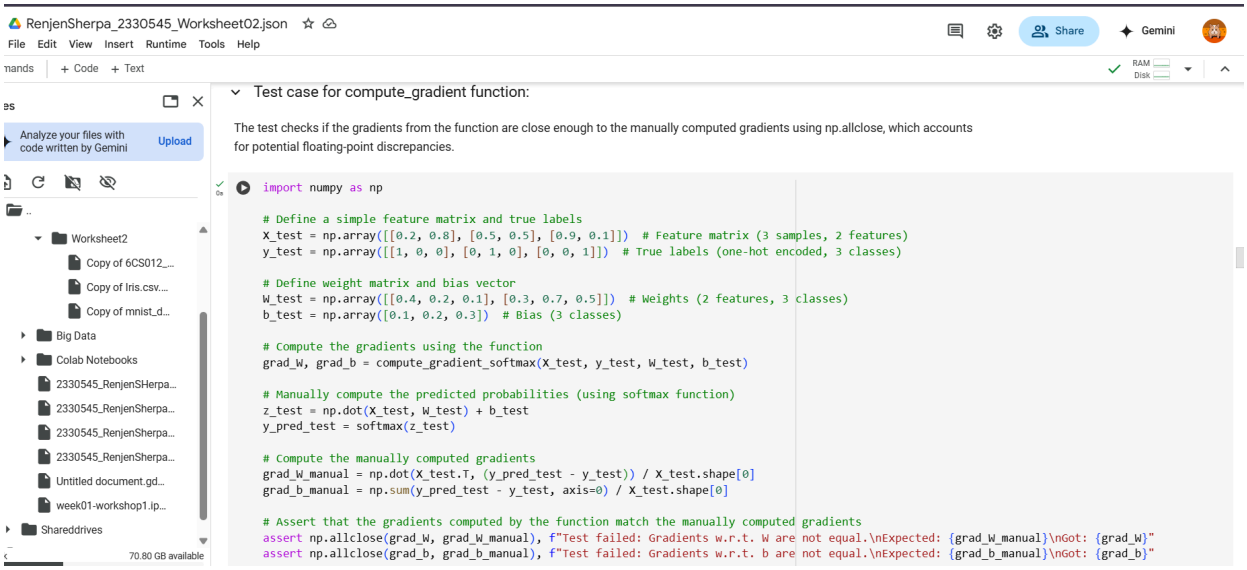
Shareddrives

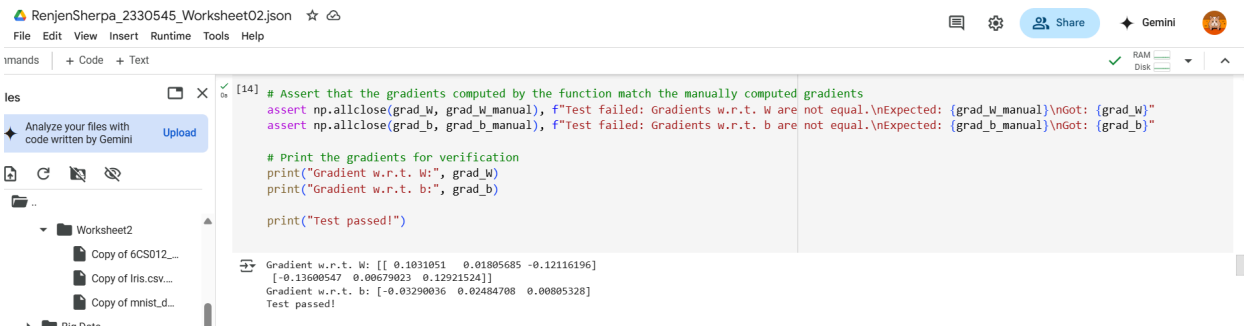
70.80 GB available

Computing Gradients:

```
import numpy as np

def compute_gradient_softmax(X, y, W, b):
    """
    Compute the gradients of the cost function with respect to weights and biases.

    Parameters:
    X (numpy.ndarray): Feature matrix of shape (n, d).
    y (numpy.ndarray): True labels (one-hot encoded) of shape (n, c).
    W (numpy.ndarray): Weight matrix of shape (d, c).
    b (numpy.ndarray): Bias vector of shape (c,).

    Returns:
    tuple: Gradients with respect to weights (d, c) and biases (c,).
    """

    # Compute logits: XW + b
    logits = np.dot(X, W) + b

    # Apply softmax to get probabilities
    y_pred = softmax(logits)

    # Compute the error term (difference between predicted and true labels)
    error = y_pred - y

    # Compute the gradient with respect to weights
    grad_W = np.dot(X.T, error) / X.shape[0]  # Average over all samples (n, c)
```


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text
RAM
Disk
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
# Compute logits: XW + b
logits = np.dot(X, W) + b
# Apply softmax to get probabilities
y_pred = softmax(logits)
# Compute the error term (difference between predicted and true labels)
error = y_pred - y
# Compute the gradient with respect to weights
grad_W = np.dot(X.T, error) / X.shape[0] # Average over all samples (n, c)
# Compute the gradient with respect to biases
grad_b = np.sum(error, axis=0) / X.shape[0] # Average over all samples (c,)
return grad_W, grad_b


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text
RAM
Disk
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives
70.80 GB available
Test case for compute_gradient function:
The test checks if the gradients from the function are close enough to the manually computed gradients using np.allclose, which accounts for potential floating-point discrepancies.
import numpy as np
# Define a simple feature matrix and true labels
X_test = np.array([[0.2, 0.8], [0.5, 0.5], [0.9, 0.1]]) # Feature matrix (3 samples, 2 features)
y_test = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]]) # True labels (one-hot encoded, 3 classes)
# Define weight matrix and bias vector
W_test = np.array([[0.4, 0.2, 0.1], [0.3, 0.7, 0.5]]) # Weights (2 features, 3 classes)
b_test = np.array([0.1, 0.2, 0.3]) # Bias (3 classes)
# Compute the gradients using the function
grad_W, grad_b = compute_gradient_softmax(X_test, y_test, W_test, b_test)
# Manually compute the predicted probabilities (using softmax function)
z_test = np.dot(X_test, W_test) + b_test
y_pred_test = softmax(z_test)
# Compute the manually computed gradients
grad_W_manual = np.dot(X_test.T, (y_pred_test - y_test)) / X_test.shape[0]
grad_b_manual = np.sum(y_pred_test - y_test, axis=0) / X_test.shape[0]
# Assert that the gradients computed by the function match the manually computed gradients
assert np.allclose(grad_W, grad_W_manual), f"Test failed: Gradients w.r.t. W are not equal.\nExpected: {grad_W_manual}\nGot: {grad_W}"
assert np.allclose(grad_b, grad_b_manual), f"Test failed: Gradients w.r.t. b are not equal.\nExpected: {grad_b_manual}\nGot: {grad_b}"


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text
RAM
Disk
Analyze your files with code written by Gemini
Upload
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
[14]
Assert that the gradients computed by the function match the manually computed gradients
assert np.allclose(grad_W, grad_W_manual), f"Test failed: Gradients w.r.t. W are not equal.\nExpected: {grad_W_manual}\nGot: {grad_W}"
assert np.allclose(grad_b, grad_b_manual), f"Test failed: Gradients w.r.t. b are not equal.\nExpected: {grad_b_manual}\nGot: {grad_b}"
# Print the gradients for verification
print("Gradient w.r.t. W:", grad_W)
print("Gradient w.r.t. b:", grad_b)
print("Test passed!")
Gradient w.r.t. W: [[0.1031051   0.01805685 -0.12116196]
 [-0.13600547 0.00679023 0.12921524]]
Gradient w.r.t. b: [-0.03290036 0.02484708 0.00805328]
Test passed!

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Implementing Gradient Descent:

```
[15] def gradient_descent_softmax(X, y, W, b, alpha, n_iter, show_cost=False):
    """
    Perform gradient descent to optimize the weights and biases.

    Parameters:
    X (numpy.ndarray): Feature matrix of shape (n, d).
    y (numpy.ndarray): True labels (one-hot encoded) of shape (n, c).
    W (numpy.ndarray): Weight matrix of shape (d, c).
    b (numpy.ndarray): Bias vector of shape (c,).
    alpha (float): Learning rate.
    n_iter (int): Number of iterations.
    show_cost (bool): Whether to display the cost at intervals.

    Returns:
    tuple: Optimized weights, biases, and cost history.
    """
    cost_history = []

    for i in range(n_iter):
        # Compute gradients
        grad_W, grad_b = compute_gradient_softmax(X, y, W, b)

        # Update weights and biases using gradient descent
        W -= alpha * grad_W
        b -= alpha * grad_b
```

Analyze your files with code written by Gemini

Upload

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv....

Copy of mnist_d...

Big Data

Colab Notebooks

2330545_RenjenSHerpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

Untitled document.gd...

week01-workshop1.ip...

Shareddrives

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

```
    # Update weights and biases using gradient descent
    W -= alpha * grad_W
    b -= alpha * grad_b

    # Compute the cost and add to cost history
    cost = cost_softmax(X, y, W, b)
    cost_history.append(cost)

    # Optionally display cost at intervals
    if show_cost and i % 10 == 0:
        print(f"Iteration {i}, Cost: {cost:.4f}")

return W, b, cost_history
```

Analyze your files with code written by Gemini

Upload

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv....

Copy of mnist_d...

Big Data

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Preparing Dataset:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

def load_and_prepare_mnist(csv_file, test_size=0.2, random_state=42):
    """
    Reads the MNIST CSV file, splits data into train/test sets, and plots one image per class.

    Arguments:
    csv_file (str)       : Path to the CSV file containing MNIST data.
    test_size (float)    : Proportion of the data to use as the test set (default: 0.2).
    random_state (int)   : Random seed for reproducibility (default: 42).

    Returns:
    X_train, X_test, y_train, y_test : Split dataset.
    """

    # Load dataset
    df = pd.read_csv(csv_file)

    # Separate labels and features
    y = df.iloc[:, 0].values  # First column is the label
    X = df.iloc[:, 1:].values  # Remaining columns are pixel values
```

Analyze your files with code written by Gemini

Upload

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv....

Copy of mnist_d...

Big Data

Colab Notebooks

2330545_RenjenSHerpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

Untitled document.gd...

week01-workshop1.ip...

Shareddrives

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

Analyze your files with code written by Gemini Upload

Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives

70.80 GB available

```
    # Normalize pixel values (optional but recommended)
    X = X / 255.0  # Scale values between 0 and 1

    # Split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=random_state)

    # Plot one sample image per class
    plot_sample_images(X, y)

    return X_train, X_test, y_train, y_test

def plot_sample_images(X, y):
    """
    Plots one sample image for each digit class (0-9).

    Arguments:
    X (np.ndarray): Feature matrix containing pixel values.
    y (np.ndarray): Labels corresponding to images.
    """

    plt.figure(figsize=(10, 4))
    unique_classes = np.unique(y)  # Get unique class labels

    for i, digit in enumerate(unique_classes):
        index = np.where(y == digit)[0][0]  # Find first occurrence of the class
        image = X[index].reshape(28, 28)  # Reshape 1D array to 28x28

        plt.subplot(2, 5, i + 1)
        plt.imshow(image, cmap='gray')
        plt.title(f"Digit: {digit}")
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

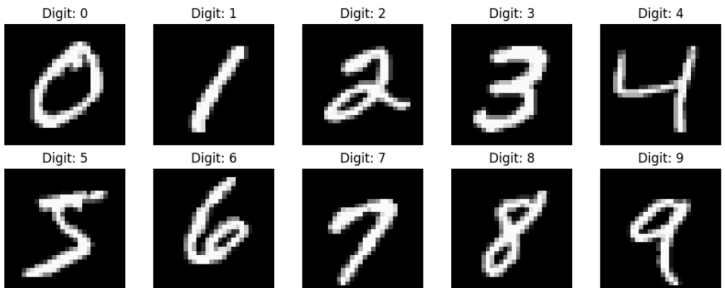
+ Code + Text

Analyze your files with code written by Gemini Upload

Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives

```
        plt.imshow(image, cmap='gray')
        plt.title(f"Digit: {digit}")
        plt.axis('off')

    plt.tight_layout()
    plt.show()
```

```
csv_file_path = "/content/drive/MyDrive/Artificial intelligence and machine learning/Worksheet2/Copy of mnist_dataset.csv"  # Path to saved datas
X_train, X_test, y_train, y_test = load_and_prepare_mnist(csv_file_path)
```



RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

Analyze your files with code written by Gemini Upload

Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives

70.80 GB available

A Quick debugging Step:

```
[19] # Assert that X and y have matching lengths
     assert len(X_train) == len(y_train), f"Error: X and y have different lengths! X={len(X_train)}, y={len(y_train)}"
     print("Move forward: Dimension of Feture Matrix X and label vector y matched.")
```

```
Move forward: Dimension of Feture Matrix X and label vector y matched.
```

Train the Model:

```
[20] print(f"Training data shape: {X_train.shape}")
     print(f"Test data shape: {X_test.shape}")
```

```
Training data shape: (48000, 784)
Test data shape: (12000, 784)
```

```
[21] from sklearn.preprocessing import OneHotEncoder

     # Check if y_train is one-hot encoded
     if len(y_train.shape) == 1:
         encoder = OneHotEncoder(sparse_output=False)  # Use sparse_output=False for newer versions of sklearn
         y_train = encoder.fit_transform(y_train.reshape(-1, 1))  # One-hot encode labels
         y_test = encoder.transform(y_test.reshape(-1, 1))  # One-hot encode test labels

     # Now y_train is one-hot encoded, and we can proceed to use it
     d = X_train.shape[1]  # Number of features (columns in X_train)
     c = y_train.shape[1]  # Number of classes (columns in y_train after one-hot encoding)
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2
- Copy of 6CS012_...
- Copy of Iris.csv....
- Copy of mnist_d...

Big Data

Colab Notebooks
- 2330545_RenjenSHerpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- Untitled document.gd...
- week01-workshop1.ip...

Shareddrives

70.80 GB available

```
[21] # Now y_train is one-hot encoded, and we can proceed to use it
     d = X_train.shape[1]  # Number of features (columns in X_train)
     c = y_train.shape[1]  # Number of classes (columns in y_train after one-hot encoding)

     # Initialize weights with small random values and biases with zeros
     W = np.random.randn(d, c) * 0.01  # Small random weights initialized
     b = np.zeros(c)  # Bias initialized to 0

     # Set hyperparameters for gradient descent
     alpha = 0.1  # Learning rate
     n_iter = 1000  # Number of iterations to run gradient descent

     # Train the model using gradient descent
     W_opt, b_opt, cost_history = gradient_descent_softmax(X_train, y_train, W, b, alpha, n_iter, show_cost=True)

     # Plot the cost history to visualize the convergence
     plt.plot(cost_history)
     plt.title('Cost Function vs. Iterations')
     plt.xlabel('Iterations')
     plt.ylabel('Cost')
     plt.grid(True)
     plt.show()
```

```
Iteration 0, Cost: 2.2128
Iteration 10, Cost: 1.5064
Iteration 20, Cost: 1.1661
Iteration 30, Cost: 0.9817
Iteration 40, Cost: 0.8679
Iteration 50, Cost: 0.7906
Iteration 60, Cost: 0.7345
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2
- Copy of 6CS012_...
- Copy of Iris.csv....
- Copy of mnist_d...

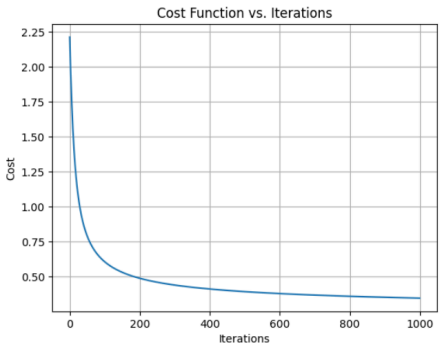
Big Data

Colab Notebooks
- 2330545_RenjenSHerpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- Untitled document.gd...
- week01-workshop1.ip...

Shareddrives

70.80 GB available

```
Iteration 910, Cost: 0.3551
Iteration 920, Cost: 0.3525
Iteration 930, Cost: 0.3519
Iteration 940, Cost: 0.3512
Iteration 950, Cost: 0.3506
Iteration 960, Cost: 0.3500
Iteration 970, Cost: 0.3494
Iteration 980, Cost: 0.3488
Iteration 990, Cost: 0.3483
```



RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

Worksheet2
- Copy of 6CS012_...
- Copy of Iris.csv....
- Copy of mnist_d...

Big Data

Colab Notebooks
- 2330545_RenjenSHerpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- 2330545_RenjenSherpa...
- Untitled document.gd...
- week01-workshop1.ip...

Shareddrives

Evaluating the Model:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Evaluation Function
def evaluate_classification(y_true, y_pred):
    """
    Evaluate classification performance using confusion matrix, precision, recall, and F1-score.

    Parameters:
    y_true (numpy.ndarray): True labels
    y_pred (numpy.ndarray): Predicted labels

    Returns:
    tuple: Confusion matrix, precision, recall, F1 score
    """
    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)

    # Compute precision, recall, and F1-score
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    return cm, precision, recall, f1
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

mands + Code + Text

RAM Disk

les

Analyze your files with code written by Gemini Upload

..

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv...

Copy of mnist_d...

Big Data

Colab Notebooks

2330545_RenjenSHerpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

Untitled document.gd...

week01-workshop1.ip...

Shareddrives

70.80 GB available

```
# Predict on the test set
y_pred_test = predict_softmax(X_test, W_opt, b_opt)

# Evaluate accuracy
y_test_labels = np.argmax(y_test, axis=1)  # True labels in numeric form

# Evaluate the model
cm, precision, recall, f1 = evaluate_classification(y_test_labels, y_pred_test)

# Print the evaluation metrics
print("\nConfusion Matrix:")
print(cm)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")

# Visualizing the Confusion Matrix
fig, ax = plt.subplots(figsize=(12, 12))
cax = ax.imshow(cm, cmap='Blues')  # Use a color map for better visualization

# Dynamic number of classes
num_classes = cm.shape[0]
ax.set_xticks(range(num_classes))
ax.set_yticks(range(num_classes))
ax.set_xticklabels([f'Predicted {i}' for i in range(num_classes)])
ax.set_yticklabels([f'Actual {i}' for i in range(num_classes)])

# Add labels to each cell in the confusion matrix
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
```

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

mands + Code + Text

RAM Disk

es

Analyze your files with code written by Gemini Upload

..

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv...

Copy of mnist_d...

Big Data

Colab Notebooks

2330545_RenjenSHerpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

2330545_RenjenSherpa...

Untitled document.gd...

week01-workshop1.ip...

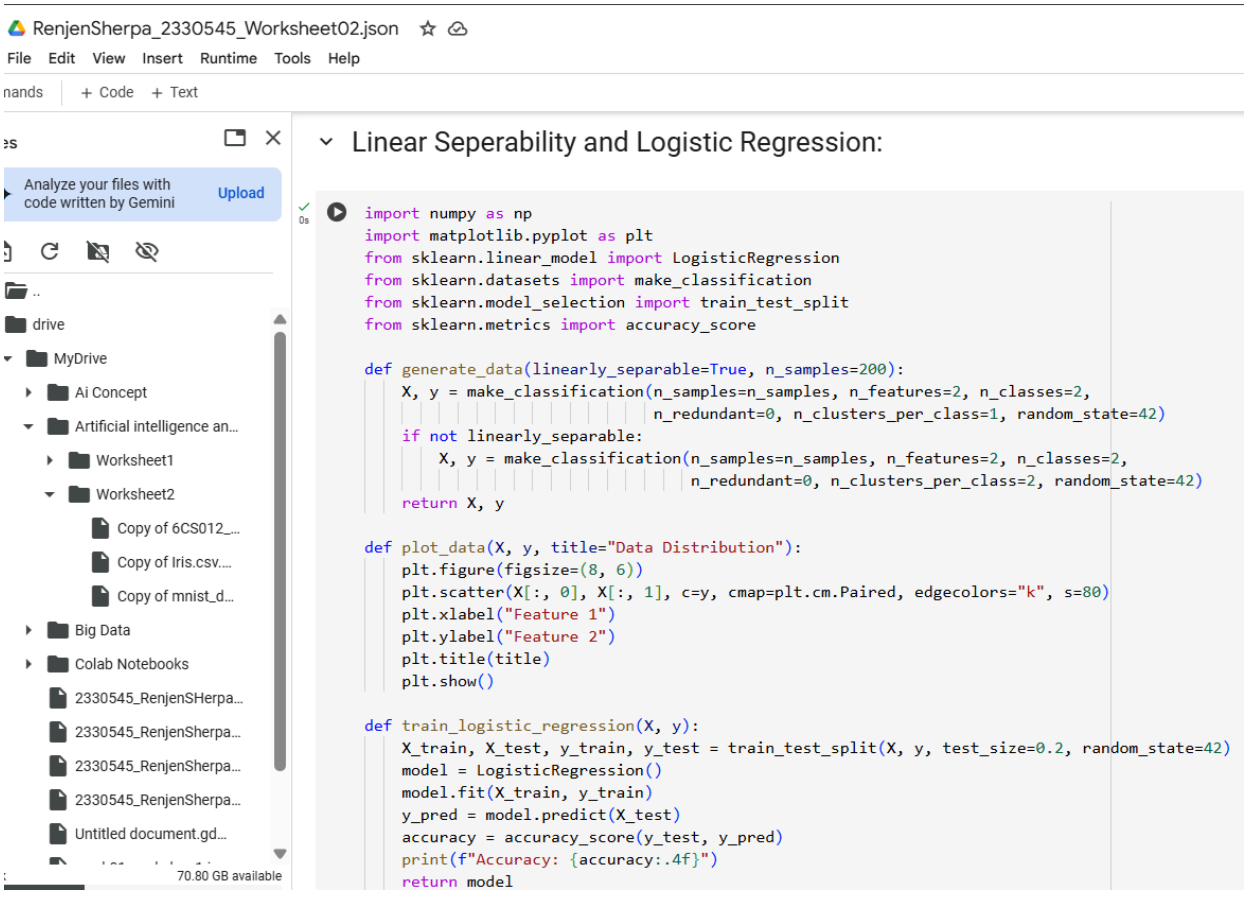
Shareddrives

70.80 GB available

```
# Add labels to each cell in the confusion matrix
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='white' if cm[i, j] > np.max(cm) / 2 else 'black')

# Add grid lines and axis labels
ax.grid(False)
plt.title('Confusion Matrix', fontsize=14)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('Actual Label', fontsize=12)

# Adjust layout
plt.tight_layout()
plt.colorbar(cax)
plt.show()
```

```
Confusion Matrix:
[[1127    0    5    2    3   11    9    2   13    3]
 [   0 1274    7   11    1    5    1    4   18    1]
 [   1   15 1029   15   20    4   26   24   33    7]
 [   8    5   33 1048    1   56    8    8   31   21]
 [   1    5    7    1 1092    0   11    4    4   51]
 [  22   14   14   43   12  924   14    7   41   13]
 [   7    2    9    1   10   16 1120    2   10    0]
 [   7   27   24    5   15    3    0 1182    7   29]
 [   9   27   14   34    9   31   13    6 1002   15]
 [   8    6   10   18   43    9    0   38   10 1052]]
Precision: 0.90
Recall: 0.90
F1-Score: 0.90
```


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
nands + Code + Text
es
Analyze your files with code written by Gemini
Upload
Artificial intelligence an...
Worksheet1
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
week01-workshop1.ip...
Shareddrives
sample_data
70.80 GB available
Confusion Matrix
Actual Label
Actual 0 1127 0 5 2 3 11 9 2 13 3
Actual 1 0 1274 7 11 1 5 1 4 18 1
Actual 2 1 15 1029 15 20 4 26 24 33 7
Actual 3 8 5 33 1048 1 56 8 8 31 21
Actual 4 1 5 7 1 1092 0 11 4 4 51
Actual 5 22 14 14 43 12 924 14 7 41 13
Actual 6 7 2 9 1 10 16 1120 2 10 0
Actual 7 7 27 24 5 15 3 0 1182 7 29
1000
800
600
400


RenjenSherpa_2330545_Worksheet02.json
File Edit View Insert Runtime Tools Help
nands + Code + Text
es
Analyze your files with code written by Gemini
Upload
drive
MyDrive
Ai Concept
Artificial intelligence an...
Worksheet1
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks
2330545_RenjenSHerpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
2330545_RenjenSherpa...
Untitled document.gd...
70.80 GB available
Linear Seperability and Logistic Regression:
0s
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
def generate_data(linearly_separable=True, n_samples=200):
X, y = make_classification(n_samples=n_samples, n_features=2, n_classes=2,
n_redundant=0, n_clusters_per_class=1, random_state=42)
if not linearly_separable:
X, y = make_classification(n_samples=n_samples, n_features=2, n_classes=2,
n_redundant=0, n_clusters_per_class=2, random_state=42)
return X, y
def plot_data(X, y, title="Data Distribution"):
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired, edgecolors="k", s=80)
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title(title)
plt.show()
def train_logistic_regression(X, y):
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
return model

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

Share Gemini

+ Code + Text

RAM Disk

Analyze your files with code written by Gemini Upload

```
    return model

def plot_decision_boundary(X, y, model):
    h = 0.02
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.Paired)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors="k", cmap=plt.cm.Paired, s=80)
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.title("Logistic Regression Decision Boundary")
    plt.show()

# Execute Steps
X, y = generate_data(linearly_separable=True)
plot_data(X, y, "Generated Data")

model = train_logistic_regression(X, y)
plot_decision_boundary(X, y, model)
```

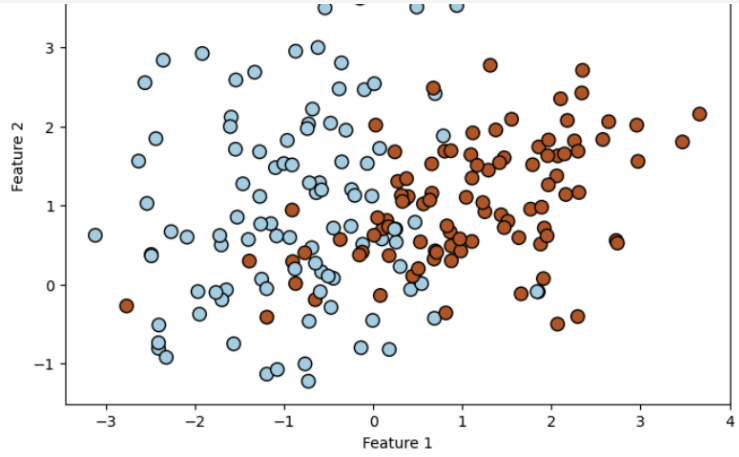
RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

+ Code + Text

Analyze your files with code written by Gemini Upload

```
model = train_logistic_regression(X, y)
plot_decision_boundary(X, y, model)
```




```
Accuracy: 0.8750
```

drive

MyDrive

Ai Concept

Artificial intelligence an...

Worksheet1

Worksheet2

Copy of 6CS012_...

Copy of Iris.csv....

Copy of mnist_d...

Big Data

Colab Notebooks

RenjenSherpa_2330545_Worksheet02.json

File Edit View Insert Runtime Tools Help

mands + Code + Text

les

Analyze your files with code written by Gemini Upload

..
drive
MyDrive
Ai Concept
Artificial intelligence an...
Worksheet1
Worksheet2
Copy of 6CS012_...
Copy of Iris.csv....
Copy of mnist_d...
Big Data
Colab Notebooks