**Task 1: Load MNIST and Show Montage**

# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

# Load MNIST dataset

(x\_train, \_), (\_, \_) = mnist.load\_data()

# Display a montage of images

plt.figure(figsize=(10, 10))

montage = np.zeros((28 \* 5, 28 \* 5))

for i in range(5):

for j in range(5):

montage[i \* 28: (i + 1) \* 28, j \* 28: (j + 1) \* 28] = x\_train[np.random.randint(0, x\_train.shape[0])]

plt.imshow(montage, cmap='gray')

plt.axis('off')

plt.title('Montage of MNIST Images')

plt.show()

## Task 2: Run Random y=mx Model on MNIST

# Assuming you want a random slope (m) for y=mx model

m = np.random.rand()

# Flatten images for linear regression

x\_train\_flat = x\_train.reshape(x\_train.shape[0], -1)

# Generate random predictions using y=mx model

predictions = m \* x\_train\_flat

# Display results (you might want to compare predictions with actual labels)

print(f"Random Slope (m): {m}")

print(f"Sample Predictions: {predictions[:5]}")

## Task 3: Train Random Walk Model to at Least 75%

# Assuming you want a simple random walk model

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Flatten images for random walk model

x\_train\_flat = x\_train.reshape(x\_train.shape[0], -1)

# Generate random labels for random walk

y\_random\_walk = np.random.randint(0, 2, size=x\_train\_flat.shape[0])

# Split data into training and testing sets

x\_train\_rw, x\_test\_rw, y\_train\_rw, y\_test\_rw = train\_test\_split(x\_train\_flat, y\_random\_walk, test\_size=0.2, random\_state=42)

# Train Random Forest classifier

rf\_classifier = RandomForestClassifier(random\_state=42)

rf\_classifier.fit(x\_train\_rw, y\_train\_rw)

# Make predictions on the test set

y\_pred\_rw = rf\_classifier.predict(x\_test\_rw)

# Calculate accuracy

accuracy = accuracy\_score(y\_test\_rw, y\_pred\_rw)

print(f"Accuracy of Random Walk Model: {accuracy \* 100:.2f}%")