**MNIST data analysis**

***General view***

**Here's a breakdown on how to effectively demonstrate Orange's MNIST analysis capabilities**

1. **Directions of Using MNIST in Orange** (c)**:**
   * MNIST dataset can be used in Orange for various tasks such as classification, clustering, visualization, and dimensionality reduction.
   * It serves as a popular benchmark dataset for evaluating machine learning algorithms and models.
2. **File Organization for MNIST in Orange:**

2.1 For classification purposes:

2.1.1 You can use preprocessed MNIST files where each row represents a digit image, *and the label (digit class) is included in a separate column*. One such popular format is CSV (Comma-Separated Values) format. You can find preprocessed MNIST datasets in this format on various websites such as Kaggle or GitHub.

2.1.2 If you have the original MNIST dataset in its original format (e.g., binary files), you can write a Python code to preprocess it into the desired format.

1. **Widgets in Orange for MNIST:**

3.1 To demonstrate MNIST: - You can use widgets like "Data Table" to visualize the dataset, "*Image Viewer*" to view the digit images, and "Scatter Plot" or t-SNE to visualize the data in a lower-dimensional space.

3.2 For classification: - Widgets like "Test and Score," "Neural Network," "Random Forest," "Support Vector Machine (SVM)," "Rank," etc., can be used for building and evaluating classification models.

3.3 For training and testing: It's recommended to split dataset into training and testing sets. A common split is 80% for training and 20% for testing, but you can adjust this ratio based on your specific requirements. You can store the dataset on your laptop or use it directly from an internet resource, depending on the dataset size and your preference.

1. **Other Tasks in Orange:**

Apart from classification, you can perform various other tasks such as clustering, regression, association rule mining, feature selection, and visualization using Orange.

The Python code for ***preprocessing the original MNIST dataset into a CSV*** format *with digits indicated in rows*.

Here's the code:

* **USE the following ref with preprocessed MNIST in CSV  
  https://www.kaggle.com/datasets/oddrationale/mnist-in-csv?resource=download**

**(v2)**

**Using MNIST with Orange for Classification** ( г)

Here's a breakdown of how we can demonstrate Orange's capabilities with MNIST data for classification.

**1. Directions for Using MNIST in Orange:**

Orange offers various ways to utilize MNIST data:

* **Classification:** Train a model to identify handwritten digits (0-9) in images.
* **Data Exploration:** Analyze the distribution of pixel intensities and visualize the image data.
* **Feature Engineering:** Use techniques like Principal Component Analysis (PCA) and others to extract relevant features from the images.
* **Comparison of Learners:** Train different classification algorithms (e.g., Neural Networks, Support Vector Machines, others) and compare their performance.

**2. Widgets for MNIST Classification:**

**2.1 Demonstrating MNIST:**

* **Import Data:** Use the "Import Data" widget to load the preprocessed MNIST data (CSV or similar format).
* **Data Table:** Explore the data in the "Data Table" widget, visualizing sample images and their corresponding digit labels.
* **Image Viewer:** Connect the "Data Table" to the "Image Viewer" widget to display individual images on selection.

**2.2 Classification Widgets:**

* **Test & Score:** Evaluate the performance of different learners (classification algorithms) on the MNIST data.
* **Learners:** You can experiment with various learners including:
  + **NNets:** Train a Neural Network model for digit classification.
  + **PCA:** Use PCA for dimensionality reduction before classification with other learners (optional).
  + **Random Forest:** Train a Random Forest classifier for digit recognition.
  + **Support Vector Machine (SVM):** Train an SVM classifier for MNIST data.
  + **KNN:** Implement K-Nearest Neighbors for classification (may require additional coding).
  + **Rank approach (widgets)**

**2.3 Training and Testing Samples:**

* A common split is 60,000 images for training and 10,000 images for testing. You can adjust this based on your computational resources and desired training time.
* Using all samples for training is possible but might lead to overfitting.
* Downloading the MNIST dataset is recommended for this demonstration. However, for larger projects, consider online resources for real-time data access.

Here are some online MNIST resources:

* [**http://yann.lecun.com/exdb/mnist/**](http://yann.lecun.com/exdb/mnist/)
* [**https://www.tensorflow.org/datasets/keras\_example**](https://www.tensorflow.org/datasets/keras_example)

**3. Other Tasks in Orange:**

Orange allows various analysis tasks beyond classification:

* **Clustering:** Group similar digits (e.g., clustering digits with similar shapes).
* **Regression:** Predict continuous values from MNIST images (e.g., predicting the "thickness" of a handwritten digit).
* **Visualization:** Explore the data using techniques like scatter plots and parallel coordinates.

**Further recommendations**

**Importing and Splitting Data (using Preprocessed CSV):**

1. Import both files (training and testing) into Orange using the "Import Data" widget.
2. Split the training data into training and validation sets (optional but recommended) using the "***Data Sampler***" widget within Orange. This helps prevent overfitting.
3. *Connect the training sets* (which is used for training and validation) (from training data) *to your learners* (NNet, PCA, SVM, etc.) for training.
4. *Connect the testing data* to the "Test & Score" widget to evaluate the trained model's performance.

**Test & Score Configuration:**

* In the "Test & Score" widget, set the "*Learner*" *input to the trained model* (e.g., the NNet you trained).
* *Set the "Test Data" input to the testing data*.
* Choose the evaluation metrics (e.g., accuracy) to assess the model's performance.

**Using Widgets like Scatter Plots and t-SNE:**

* Use the "Scatter Plot" widget to visualize relationships between features in the data (e.g., plotting pixel intensities of different digits).
* Use the "t-SNE" widget for dimensionality reduction, which can be helpful before training models with many features (28 x 28 pixels per image in MNIST).