TECH-GB.3332 Fall 2024 Prof. Alex Tuzhilin

Assignment 2

**Due date:** October 11 (11am)

In this homework, you will use AI Studio to build predictive models that use Deep Learning based Embeddings, Feedforward and Convolutional Neural Networks methods.

To start this assignment, download the [trip\_advisor\_2000.xlsx](https://docs.google.com/spreadsheets/d/1sw2LqJ_AFfzyFlKwMJFsMF9shhV3enfm/edit?usp=sharing&ouid=114141838929637485833&rtpof=true&sd=true) dataset and open AI Studio. Then, import the dataset into AI Studio following the same process that we used in the Lab session.

## Part 1. Embeddings

In this part, your task is to predict the ratings of restaurants on Trip Advisor using review embeddings and other attributes from the downloaded “trip\_advisor\_2000” dataset (somewhat similar to what we have done in the Lab). As a starting point, you need to transform the texts of the reviews into embeddings.

To do this, drag the Trip Advisor dataset from the Repository panel to the Process panel (and thus create the “Retrieve” operator icon there). In the “Operators” panel, search for “Embeddings” and drag “Embeddings (FastEmbed)” (its path: Extensions\Generative Models\Embeddings) to the Process panel. Next, in the “Operators” panel, search for the “Split” operator (its path: Blending\Values) and drag it to the Process panel. Finally, in the “Operators” panel, search for the “Write Excel” operator (its path: Data Access\Files\Write) and drag it to the Process panel.

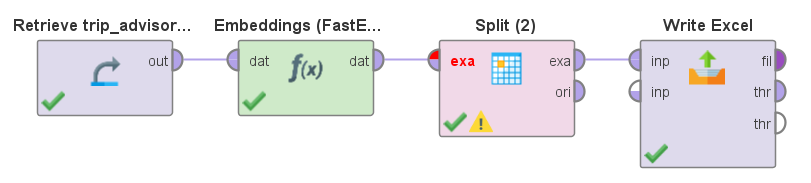
Link the “Retrieve”, “Embedding”, “Split” and “Write Excel” operators as is shown in Figure 1. After that, select the parameters of these operators (in the Parameters panel) as follows:

*Embeddings (FastEmbed):* set the model as “BAAI/bge-small-en-v1.5[384dim]” that will generate a 384-dimensional embedding. Next, select “review\_content” for the input and enter “review\_embed,” as the output column name.

*Split*: Select the attribute filter type as “single” and identify attribute as “review\_embed” (Note that you need to *manually* type “review\_embed” in the attribute field because the review\_embed column has not been created before executing the program).

*Write Excel*: Specify the excel file as “trip\_advisor\_embedding.xlsx” and its file format as “xlsx” (also select the path where you want to save the embedding file).

After selecting the paraments, save the embedding generation model in the Local Repository in its “Processes” subfolder (see the panel on your left) and run the model by pressing the “Run” button in the top menu. When model execution terminates, you can go to the directory where the “trip\_advisor\_embedding.xlsx” spreadsheet has been saved and examine it (note that the initial “trip\_advisor\_2000” spreadsheet has been extended with 384 embeddings (numbers) in its rightmost columns).

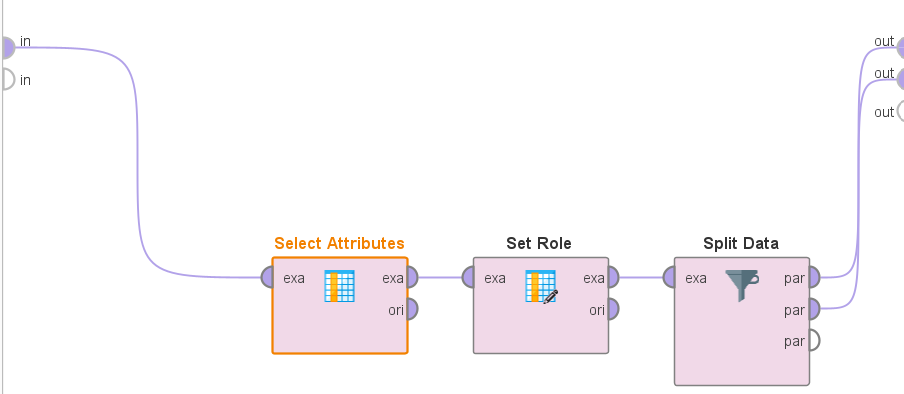


**Figure 1.** The flowchart of the Embedding Generation Process

Next, close the saved embedding generation process and open a new “blank” process by going to the File menu and selecting the “New Process” option. Next, load the “trip\_advisor\_embedding.xlsx” dataset back to the AI Studio by clicking on the “Import Data” button in the Repository panel and repeating the spreadsheet loading process. Please, ensure that all the “review\_embed” columns (384 of them) have the type *real* and that the “review\_rating” column is of type *polynominal* (or *nominal)* type*.* If this is not the case, right-click on the “trip\_advisor\_embedding” dataset in the Local Repository data folder and select the “Edit” option; then edit the data type of the “review\_rating” column to the *polynominal* type. Also make sure that the rest of the columns in the “trip\_advisor\_embedding” dataset have the same data types as in the “trip\_adviser\_2000” dataset and also as described in the Lab assignment. Your embedding dataset is ready to be used for building deep learning models now.

## Part 2. Preprocessing the Data

Similarly to how it was done in the Lab, drag the “trip\_advisor\_embedding” dataset into the Process panel, thus creating the Retrieve icon there, and also create the “Subprocess” operator in the Process panel that includes the “Select Attributes”, “Set Role” and “Split Data” submodules, as is shown in Figure 2. As in the Lab, chose “review\_rating” as the label in the “Set\_Role” module. To evaluate performance of the model that we will build as described below, you need to split the dataset into the train and the test sets in the 0.8/0.2 ratio (80% training and 20% testing data), which should be done in the “Split Data” module. Also, connect all these modules, as shown in Figure 2.



**Figure 2.** The flowchart of the Subprocess Operator.

After that, return to the Process level and connect the Retrieve and the Select\_Subprocess modules similarly to how it was done in the Lab. Next, we build the DL module, as is explained in the next section.

## Part 3. Building the FeedForward Neural Network

(a) *Simple Feedforward Network Based on Ratings*. In this part, we will use only the numeric ratings information from the “Trip\_advisor\_2000” dataset to predict the overall rating (similar to what we have done in the Lab). First, select attributes *rest\_nb\_review, rest\_rating, rest\_rating\_neutral, rest\_rating\_poor, rest\_rating\_terrible, rest\_rating\_very\_good, review\_rating, review\_rating\_atmosphere, review\_rating\_food, review\_rating\_service, review\_rating\_value* by going to the Subprocess/Select\_Attributes operator and selecting those attributes, as we have done it in the Lab. After that, build the Deep Learning model, as shown in Figure 3 of the Lab. In particular, in the “Deep Learning” operator [see the Subprocess diagram in Figure 5 of the Lab], use one fully-connected layer as the first layer (with 32 neurons and the ReLU activation function) and an output layer (fully connected with 5 neurons, the loss function set to “Multiclass Cross Entropy”, and the activation being “Softmax”; note that you need to click on the “Show advanced parameters” option at the bottom of the Parameters panel to select 5 neurons in the Output layer). Also, set parameters of the Deep Learning module to 50 epochs and the learning rate to 0.005. In the “Performance (Classification)” module (that follows the “Apply Model” operator) select “accuracy” as your (only) performance criterion. Finally, press the “Run” button in the top menu, run the model and report the performance results (by clicking on the “Performance Vector” tab in the “Results” panel and examining the confusion matrix and the accuracy rate) and the running time (measure how much time it took you to run the model). Also, comment briefly on the performance results.

(b) *Rating Prediction Based on the Review Embedding*. Go to the “Select Attributes” icon in the “Select Subprocess” module and choose the 384 “review\_embedding” attributes in this task (*all* 384 of them (!) by clicking on the “Select Attributes” parameter in the Parameters panel for the "Select Attributes" operator) and *also* the “review\_rating” attribute. The rest of the rating prediction process looks exactly the same as in Part 3(a) that you have just done: use the same first DL layer, the output layer, and the performance operator as in Part 3(a) – with the same operators and the same parameter settings. “Run” the model, as in Part 3(a) and report the performance results and the running time for this model. How would these performance results compare with those achieved by the Ratings-based model from Part 3(a)? How would you explain the performance differences?

(c) *Combining Ratings with the Embeddings*. In this part, we will combine the numeric ratings attributes from Part 3(a) and the embeddings attributes from Part 3(b) to predict the review rating. Go to the “Select Attributes” icon in the “Select Subprocess” module and add all the ratings attributes selected in Part 3(a) to the 384 “review\_embed” attributes selected in Part 3(b). Everything else remains the same in this model: the first input layer, the output layer, and the performance operator are the same as in Part 3(a) and (b). Run this model and report the performance results and the running time as in Part 3(a) and (b). How would these performance results compare with those achieved by the models from Parts 3(a) and (b)? How would you explain the performance differences?

Which of these three models would you pick for the Trip Advisor if being asked by them as a consultant and why?

## Part 4. Building Convolutional Neural Networks for Image Processing

As the first step, we need to install certain packages needed for image processing. To do this, go to the “Extensions” tab in the AI Studio and select the “Marketplace (Updates and Extensions)” option within that tab. Next, enter “Image Handling” in the search box and press the “Search” button. Then click on the “Image Handling 0.2.1” option in the search results and install the Image Handling package. After that, enter “ND4J Back End” in the search box, press the “Search” button and click on the “ND4J Back End 1.2.0” search result. Then install the ND4J Back End package.

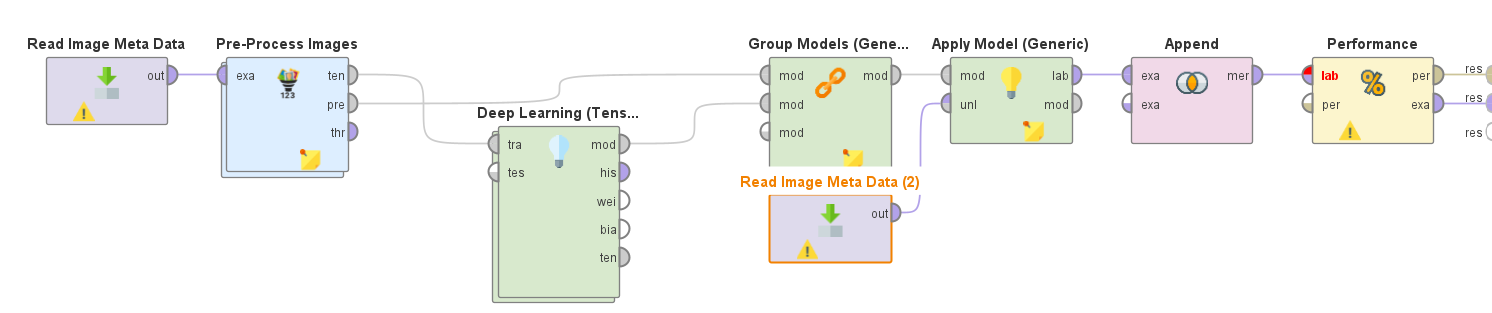
After installing both the “Image Handling” and the “ND4J Back End” image processing packages that we are going to use in this assignment,  download the [MNIST dataset](https://drive.google.com/file/d/15zQH9AeSlKFe94gfqIgyhaGdWqBu7Q48/view?usp=sharing) (by clicking on this link) and extracting it to the appropriate folder on your hard drive for later use (note that the MNIST dataset is extensively discussed in the Kneusel book in Chapter 5 and in other parts of the book).

(a) **Building a Simple CNN Classification Model Identifying Hand-written Digits**

In this exercise, we build a simplified version of the LeNet Convolutional Neural Network model (discussed in the class) that takes handwritten images of digits (as stored in the MNIST dataset) and classifies them into actual digits (from 0 to 9).

The process starts with searching for the “Read Image Meta Data” in the Operators panel and dragging the “Read Image Meta Data” operator to the Process panel. After that, search for the “Pre-Process Images”, “Deep Learning (Tensor)”, “Group Models (Generic)”, “Apply Model (Generic)”, “Append” [Path: Blending\Table\Joins] and the “Performance (Classification)” operators in the Operators panel, drag them into the Process panel and connect them, as is shown in Figure 3 (don’t forget to connect the output of the Performance module to the right wall connector of the Process panel; also note that the “Read Image Meta Data” operator should be dragged twice to the panel: the first time for the Train and the second time for the Test purposes, as shown in Figure 3 (note that the MNIST image data stored in the downloaded “mnist” folder on your hard drive is already split into the Train and the Test components).

Note that the “Group Models (Generic)” module, shown in Figure 3, allows to combine certain types of pre-processing and machine-learning models. In our case, this module combines the image pre-processing steps with the trained model to obtain one single module that can be applied to testing image files. Also, the Apply Model (Generic) module presented in Figure 3 is similar to the one we used in the Lab; but it can process high-dimensional data like images instead of only the tabular data, like spreadsheets. Finally, the Append operator is added in Figure 3 for format matching between the Apply Model (Generic)’s output and the Performance’s input.



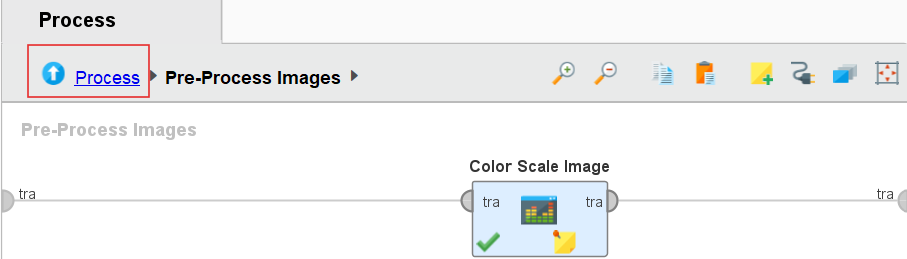
**Figure 3. Flowchart of Image Classification Model**

Furthermore, we need to set the parameters for these operators (in the Parameters panel) as follows:

*Read Image Meta Data*: Set the directory of the *left* “Read Image Meta Data” operator to the path of the mnist\_**train** dataset, and also check the “use label” parameter.

*Read Image Meta Data (2)*: Set the directory of the *right* “Read Image Meta Data” operator to the path of the mnist\_**test** dataset, and also check the “use label” parameter.

*Pre-Process Images:* Set the path parameter to “Path”. Then, double-click on the “Pre-Process Images” icon and search for the “Color Scale Image” operator in the Operators panel. Drag the “Color Scale Image” operator to the Pre-Process Images panel and connect it as shown in Figure 4. Then return to the Process panel and work on setting the parameters of the remaining operators in Figure 3 as follows.



**Figure 4. Flowchart within Pre-Process Images.**

*Deep Learning (Tensor):* click on the “Show advanced parameters” option at the bottom of the Parameters panel and set them up as follows: epochs=20, optimization method = Stochastic Gradient Descent, backpropagation = Standard, and updater = AdaDelta. After that, double-click on the *Deep Learning (Tensor)* icon to set up the Convolutional Neural Network architecture as shown in Figure 5. In particular, in the “Operator” panel search for the “Add Convolutional Layer”, “Add Pooling Layer”, “Add Dropout Layer”, “Add Fully-Connected Layer” and “Add Output Layer” operators, drag all of them to the Deep Learning Tensor panel and connect them as shown in Figure 5. Then set the parameters for these operators (in the Parameters panel) as follows:

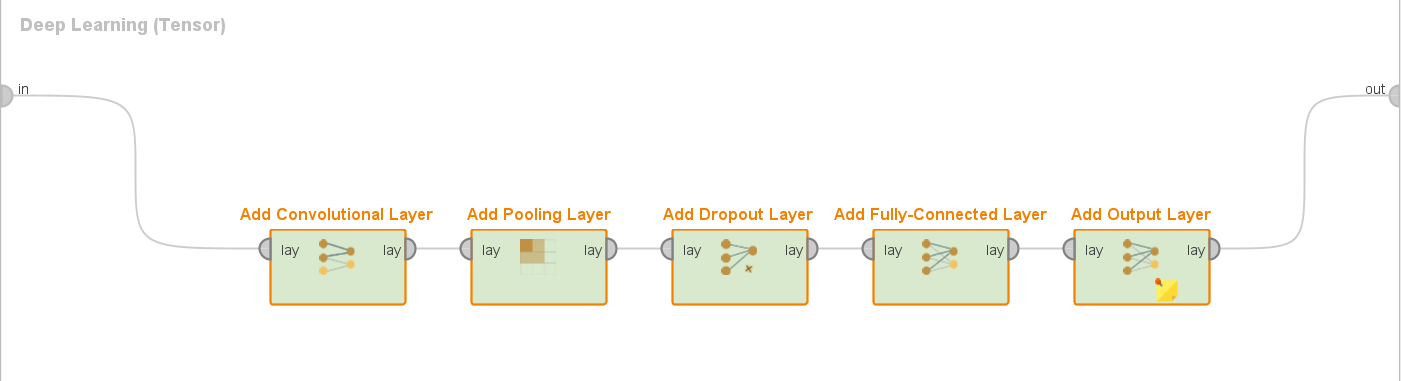
*Convolutional Layer*: number of activation maps = 32; kernel size = (3, 3); activation function = ReLU;

*Pooling Layer*: Pooling Method = max; Kernel Size = (2,2); Stride Size = (2,2);

*Dropout Layer*: dropout rate=0.25;

*Fully-Connected Layer*: neurons=128; activation function = ReLU; check use dropout box [to do it, click on the “Show advanced parameters” link]; dropout rate = 0.25;

*Output Layer*: output type = FullyConnected; loss function = Multiclass Cross Entropy (Classification); neurons = 10 [to do it, click on the “Show advanced parameters” link]; activation function = Softmax;



**Figure 5. Flowchart of the Deep Learning CNN Process.**

*Performance (Classification)*: Check the “accuracy” box.

(b) **Running the Model and Evaluation of the Performance Results***.* Run the model by clicking on the “Run” button and report the performance results (by clicking on the “Performance Vector” tab in the “Results” panel) and the running time of the model. Examine the confusion matrix and the accuracy rate and comment on the performance results, i.e., how good they are and what they mean in the context of this prediction problem.

# **Deliverables:**

Your submission file should contain two parts:

1. A pdf report that contains:
   1. Model Design (flowcharts similar to the examples shown in the figures above)
   2. Performance outputs, as requested in each question.
   3. Your comments on the model performance, as asked in each question.
2. A replicability folder in the .zip format that contains:
   1. The AI studio processes. To collect them, click on the File menu at the top left corner of AI Studio and then click on the Export Process option. Export your process file in the .rmp format as follows:
      1. Save Embedding Process as Embedding.rmp
      2. Save Feedforward Neural Networks Process as FNN.rmp
      3. Save Convolutional Neural Networks Process as CNN.rmp
   2. The “trip\_advisor\_embedding.xlsx” file