```
%% Training MLP Model %%
%%%%%%%%%%%%%%%%%%%%%%%%

%% Group work %%
% Supanut Sookkho (MSc Data Science / 230024841) & Yumi Heo (Msc Data Science / 
230003122)

%% clear command, workspace, and figures
clc;
clear;
close all;

%% Import data from the CSV file
X_train = readmatrix('X_train_ndarray.csv');
X_test = readmatrix('X_test_ndarray.csv');
X_vali = readmatrix('X_vali_ndarray.csv');
y_train = readmatrix('y_train_ndarray.csv');
y_test = readmatrix('y_test_ndarray.csv');
y_vali = readmatrix('y_vali_ndarray.csv');

%% Data preparation
% For a fair comparison, we use the same training, validation, and test set made 
from Python and input them in Matlab.

% Transpose the matrices and assign them back to the same variable.
X_train = X_train';
X_test = X_test';
X_vali = X_vali';
y_train = y_train';
y_test = y_test';
y_vali = y_vali';

% For reproducibility.
rng(123)

%% Set the Hyperparameter for the model

% For the sake of the comparison, we declared all the variables in MATLAB to be the 
same name as the variables we declared in Python.
% Although it is not 100% apples-to-apples, we compare the process between these 2 
platforms step by step.

% Declare the structure of the neural network model.
output_features = 1;
activation_function_hidden = 'poslin';
activation_function_output = 'logsig';

% The reason we specified 'poslin' to activation_function_hidden in hidden layer is
% that we can't use ReLU activation directly in Matlab with the NNstart function.
% The alternative solution has been brought from Mathworks's staff,
% which is to use 'poslin' activation.
% Reference: https://uk.mathworks.com/matlabcentral/answers/1848003-how-can-i-
apply-relu-activation-function-in-levenberg_marquardt-algorithm-for-training-
```

```
neural-networ

% Declare the Learning rate.
learning_rates_grid = [0.01, 0.1, 0.2, 0.3];

% Declare the grid search.
hidden_neurons_grid = [10, 25, 50, 100, 200];

% Training function: Update all the wieghts with the gradient descent.
trainFcn = 'traingd';

%% Define variables to plot later in Python after training model in MATLAB.

% Define null variables to store the result of the best model's hyperparameters.
best_accuracy = 0.0;
best_hyperparameters = struct('learning_rate', 0, 'hidden_neurons', 0);
best_model = [];

% Define the empty list to store the result of every combination of
hyperparameters.
learning_rates_list_heatmap = [];
hidden_neurons_list_heatmap = [];
accuracy_list_heatmap = [];

% Define the empty list to store the training time in each epoch.
training_time_bestmodel = [];

% Define the empty list to store the loss value in each epoch.
train_loss_for_plotting = [];
vali_loss_for_plotting = [];

% Set the number of epochs.
number_of_epochs = 999 % In MATLAB, the first epoch starts from 0.

%% Build a neural network model

for learning_rate = learning_rates_grid
    for hidden_neurons = hidden_neurons_grid
        fprintf('Grid search > training the NNet model with learning rate=%.4f and
hidden_neurons=%d\n', learning_rate, hidden_neurons);

        % Create the object for neural network model
        Credit_Churn_NN_Model = patternnet(hidden_neurons, trainFcn);
        Credit_Churn_NN_Model.layers{1}.transferFcn = activation_function_hidden;
        Credit_Churn_NN_Model.layers{2}.transferFcn = activation_function_output;

        % We already split the data into training, validation, and testing sets in
Python
        % and imported them to MATLAB.
        % Set the ratio to 100:0:0 to prevent for MATLAB to split them again.
        net.divideParam.trainRatio = 100/100;
        net.divideParam.valRatio = 0/100;
        net.divideParam.testRatio = 0/100;
```

```
        % Set the number of learning_rate for this loop
        Credit_Churn_NN_Model.trainParam.lr = learning_rate;

        % Set the number of epochs for this loop
        Credit_Churn_NN_Model.trainParam.epochs = number_of_epochs;

        % By default, MATLAB will stop the training automatically when some ↙
conditions are met.
        % For the fair comparison to PyTorch, set 'max_fail' to be one billion to ↙
disable early stopping.
        % And make sure that the model will run until the 1000 epochs.
        Credit_Churn_NN_Model.trainParam.max_fail = 1000000000;

        % Train the neural network model with training data set
        [trained_model,training_record] = train(Credit_Churn_NN_Model, X_train, ↙
y_train);

        % Forward propagation.
        % Get predictions from the training set.
        y_train_pred = trained_model(X_train);

        % Round the probability results to get binary classes.
        y_train_pred = round(y_train_pred);

        % Evaluate the model.
        % Calculate the accuracy rate of the training model.
        accuracy_train = sum(y_train_pred == y_train) / numel(y_train);
        fprintf('Accuracy rate (train set) of this combination = %.2f%%\n',↙
accuracy_train * 100);

        % Validate the model with validation set.
        y_vali_pred = trained_model(X_vali);
        y_vali_pred = round(y_vali_pred);
        accuracy_vali = sum(y_vali_pred == y_vali) / numel(y_vali);
        fprintf('Accuracy rate (Validation set) of this combination = %.2f%%\n',↙
accuracy_vali * 100);

        learning_rates_list_heatmap = [learning_rates_list_heatmap, learning_rate];
        hidden_neurons_list_heatmap = [hidden_neurons_list_heatmap, ↙
hidden_neurons];
        accuracy_list_heatmap = [accuracy_list_heatmap, accuracy_vali];

        % Check if the current hyperparameters yield a better performance.
        if accuracy_vali > best_accuracy
            best_accuracy = accuracy_vali;
            best_hyperparameters.learning_rate = learning_rate;
            best_hyperparameters.hidden_neurons = hidden_neurons;
            best_model = trained_model;
            training_time_bestmodel = training_record.time;
            train_loss_for_plotting = training_record.perf;
            vali_loss_for_plotting = training_record.vperf;
        end
```

```
    end
end

%% Evaluate the best-trained model with the testing dataset

% Predict the result by using the testing set.
y_test_pred = best_model(X_test);
y_test_pred = round(y_test_pred);

% Calculate the accuracy rate of the the best model.
accuracy_test = sum(y_test_pred == y_test) / numel(y_test);
fprintf('Accuracy rate (Test set) of this combination = %.2f%%\n', accuracy_test *
100);

%% Confusion matrix
% As y_test_pred and y_test are column vectors, pick each vector to
% calculate True Positive (TP), True Nagative (TN), False Positive (FP), False
Nagative (FN).
TP = sum((round(y_test_pred) == 1) & (y_test == 1));
TN = sum((round(y_test_pred) == 0) & (y_test == 0));
FP = sum((round(y_test_pred) == 1) & (y_test == 0));
FN = sum((round(y_test_pred) == 0) & (y_test == 1));

% Calculate precision, recall, and F1 score
precision = TP / (TP + FP) * (TP + FP > 0);
recall = TP / (TP + FN) * (TP + FN > 0);
f1 = 2 * (precision * recall) / (precision + recall) * (precision + recall > 0);

fprintf('The Precision of the best model = %.2f%%\n', precision * 100);
fprintf('The Recall of the best model = %.2f%%\n', recall * 100);
fprintf('The F1 score of the best model = %.2f%%\n', f1 * 100);

%% Show the confusion chart
C = confusionmat(y_test,y_test_pred)
confusionchart(y_test,y_test_pred)

%% Export to the recorded lists to CSV. Those will be used to plot in Python
% This block of the codes is commented out as we already exported those CSV
% files.

%csvwrite('matlab_training_time_bestmodel.csv', training_time_bestmodel);
%csvwrite('matlab_train_loss_for_plotting.csv', train_loss_for_plotting);
%csvwrite('matlab_vali_loss_for_plotting.csv', vali_loss_for_plotting);
```