FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Técnicas e Algoritmos em Ciência de Dados
Assessed Coursework 2

This assignment must be submitted by June 4th, 2022 at 8:00 am.
Late submissions penalties: 10% of the total value of the coursework for each hour of delay, or fraction of it.

Learning outcomes assessed

This coursework will test the fundamentals of neural network training, and random forests implementation.

Instructions

Identifier
Please choose a random number of 6 digits and write it in the first cell of the notebook. Make sure that you keep a copy of that number as it will be used to provide the feedback (please avoid trivial numbers, such as 000000 or 123456 – thank you).

Submission
Submit your files through ECLASS. The files you submit cannot be overwritten by anyone else, and they cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submissions after the deadline will incur in a penalty: 10% of the total value of the coursework for each hour of delay, or fraction of it.

If you have issues, at the very last minute, email your coursework as an attachment at alberto.paccanaro@fgv.br with the subject "URGENT – COURSEWORK 2 SUBMISSION". IN the body of the message, explain the reason for not submitting through ECLASS.

IMPORTANT

- Your submission will consist of a single Python notebook implementing your solutions.
- The name of the file will be the random number that identifies you (for example 568423.ipynb)
- This coursework consists of 4 parts. Please make sure that the 4 parts are clearly separated and identifiable in your Python Notebook.
- DO NOT SUBMIT ANY DATASET, only the code.
- Any utility function that you will use should be included in the notebook – do not submit utility scripts, they will NOT be run.
- Python notebooks can become unordered when writing the code. If a cell in your notebook depends on another cell that is BELOW it, marks will be deducted.

ADVICE ON BONUS EXERCISES

The value of each exercise in percentages is provided. However, note that the total sum of the marks for the exercises is 110. This is because 10 extra points are given in this coursework.

All the work you submit should be solely your own work. Coursework submissions will be checked for this.

Marking Criteria

This coursework is assessed and mandatory and is worth 25% of your total final grade for this course. In order to obtain full marks for each question, you must answer it correctly but also completely. Marks will be given for writing well structure code.

COURSEWORK

Part 1 – Multi Layer Perceptron for regression – value of this section: 15 %

For this part, download the file "Part1.tsv" from the ECLASS platform.

Here are the steps that you will need to implement:

a) Load the data into a pandas DataFrame, and get a scikit-learn compatible dataset. Use the "target" column as the target variable.
b) Make a 70%/30% split of the dataset for training and testing respectively.
c) Using numpy, create a scikit-learn regressor that implements a multilayer perceptron architecture with 5 hidden layers.
    - The dimensionality of each layer is your decision.
    - Each hidden layer must have a bias unit.
    - All activations should be the sigmoid function.
    - You must use the backpropagation algorithm to calculate the derivatives.
    - Use mini-batch gradient descent to update the weights.
    - The parameters of the estimator are the following:
        - **learning_rate:** A float number that determines the learning rate used for updating the weights on the update step of the gradient descent.
        - **batch_size:** An integer that determines the number of datapoints that are included in each mini-batch.
        - **epochs:** An integer that determines the number of times the training goes through all the datapoints.
d) Train the estimator you implemented using the training set. Use the trained estimator to predict values for the test set.
e) Use the scikit-learn MLPRegressor estimator. Train it on the training test and generate predictions for the test set.
f) Compare the performance of both models using the mean squared error metric from scikit-learn.

Part 2 – Multi Layer Perceptron for classification – value of this section: 40 %

For this part, download the file "Part2.tsv" from the ECLASS platform.

Here are the steps that you will need to implement:

a) Load the data into a pandas DataFrame, and get a scikit-learn compatible dataset. Use the "target" column as the target variable.
b) Make a 70%/30% split of the dataset for training and testing respectively.
c) Using numpy, create a scikit-learn classifier that implements a multilayer perceptron with the following parameters:
    - Each hidden layer must have a bias unit.
    - All activations should be the sigmoid function.

- You must use the backpropagation algorithm to calculate the derivatives.
- Use mini-batch gradient descent to update the weights.
- The parameters of the estimator are the following:
  - **hidden_layers_dimensions:** A list of integers that determines the number and dimensionality of the hidden layers. The number of items on the list determine the number of hidden layers. The first element of the list (at index 0) is the dimensionality of the first hidden layer (connected to the input). The last element is the dimensionality of the hidden layer (connected to the output layer). The dimensionality does not include the bias term. The dimensionality of the input and output layers should be inferred from the dimensionality of the data.
    For example: A list [4,3,2] will generate 3 hidden layers with dimensions 4, 3, and 2, respectively. If we count the bias units, the dimensions are 5, 4, 3.
  - **learning_rate:** A float number that determines the learning rate used for updating the weights in gradient descent.
  - **batch_size:** An integer that determines the number of datapoints that are included in each mini-batch.
  - **epochs:** An integer that determines the number of times the training goes through all the datapoints.

d) Train the estimator you implemented using the training set. Use the trained estimator to predict values for the test set.
   - During training, use the following parameters
     - hidden_layers_dimensions = [4,4,4,4]
     - learning_rate = 0.0001
     - batch_size = 32
     - epochs = 100

e) Evaluate the trained model on the test set using the following metrics:
   - Accuracy
   - AUC-PR
   - AUC-ROC

f) During each epoch of training, collect the loss, and make a plot with the epoch number in the X axis, and the loss on the Y axis.

g) Print the confusion matrix related to the test set.

Part 3 – Random forest for classification – value of this section: 35 %

Download the Census Income dataset (https://archive.ics.uci.edu/ml/datasets/Adult). In this part, you will implement random forests. Be aware that some of the variables in this dataset are nominal (or categorical).

Here are the steps that you will need to do:

a) Load the data into a pandas DataFrame, and get a scikit-learn compatible dataset.

b) Make a 70%/30% split of the dataset for training and testing respectively.

c) An implementation of the Random Forest algorithm, as described in Section 8.2.2 of the Witten, James, Hastie & Tibshirani book. You should add the following options:
    - A parameter **num_features** for the number of predictors to consider at each split
    - A parameter **num_trees** to control the number of trees in the forest
    - Parameters for controlling the growth of trees, you need to implement **at least** two of the following:
        - Maximum level of the tree
        - Minimum number of observations in a node
        - Stopping criterion based on the proportion of classes in the node

Part 4 – Performance comparison – value of this section: 20 %

This part relies on the models you implemented in Parts 2 and 3. You should use the dataset from Part 2.

Here are the steps that you will need to do:

a) Use your classifiers (multi-layer perceptron classifier and random forest).
    - Train them on the training set.
    - Make predictions on the test set
b) Choose any 3 classifiers from scikit-learn and repeat steps in a) for each classifier. You can find a list of classifiers on the scikit-learn documentation.
c) You will have obtained 5 sets of predictions from all the models you trained. Use these predictions and the labels from the test set to calculate the performance of each model using the following performance metrics:
    - Precision
    - Recall
    - AUC-ROC
    - AUC-PR
d) Create a comparison report. Your code will:
    - Print a table with values of the performance metric (columns) for each model (rows) on the test set.
    - Make a single figure containing 2 subplots. In the first subplot, you will plot the ROC curves for each model. In the second subplot, you will plot the PR curves for each model.

[Tip for the entire coursework: You DO NOT need to use the entire dataset while programming, test on a subset if your computer (or Google Colab) is taking very long to train the models. Marks are assigned to the code, NO MARKS ARE ASSIGNED FOR PERFORMANCE.]