MIE 524 Data Mining
Assignment 2: Machine Learning

This assignment allows students to build machine learning models.

- Programming language: Python (Google Colab Environment)
- Due Date: October 7, 2024, 11:59PM EST

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in a Python notebook (named `MIE524_A2.ipynb`) committed, together with any associated output files, to the student's assignment repository, and (3) attendance in the post-assignment lab quiz.

Please note the plagiarism policy in the syllabus. If you borrow or modify any multiline snippets of code from the web, you are required to cite the URL in a comment above the code that you used. You do not need to cite tutorials or reference content that demonstrate how to use packages – you should certainly be making use of such content. *The use of generative AI solutions to generate code or text answers to the assignment questions is not allowed.*

What/how to submit your work:

1. All your code should be included in a notebook named `MIE524_A2.ipynb` that is provided in the cloned assignment repository.
2. Commit and push your work to your GitHub repository in order to submit it. Your last commit and push before the assignment deadline will be considered to be your submission. You can check your repository online to make sure that all required files have actually been committed and pushed to your repository.
3. Your committed notebook should include all the computed outputs by first running it from top to bottom on Google Colab and then exporting the notebook.
4. A link to create a personal repository for this assignment is posted on Quercus.

This assignment has 4 points in total and the point allocation is shown below:

- Q1: 1 points
- Q2: 3 points

Note: Partial points may be given for partial answers. Points will be deducted for missing or incomplete answers. Points could also be deducted for styling.

Q1 - Gradient Boosted Trees

Implement a gradient-boosted trees model from scratch for regression. You may use scikit-learn's decision tree implementation and scikit-learn's out-of-the-box model performance metrics. You may handle the data using `Pandas` and `Numpy`. Please use `MIE524_A2.ipynb` to complete this question.

Task

a) Implement `my_GradientBoosting` and `my_GradientBoostingRegressor` classes. Train the regressor on the provided Recipes dataset [1] with your choice of hyperparameters, predicting the recipe's rating. You do not need to use the string columns from the dataset as features.

b) Evaluate your regressor using the provided test datasets. Report on the regressor's RMSE[2].

c) Evaluate your regressor's performance against both scikit-learn's[3] and Spark's[4] implementation of gradient boosted trees with the **same** set of hyperparameters you have chosen for your own implementation. Compare both the evaluation metrics and model training time.

d) Implement a random forest (consisting of regression trees from scikit-learn [5]) regressor to perform regression on the dataset. Compare the performance of this random forest using at least 3 different sets of hyper-parameters to the best performance achieved with your gradient boosted tree.

Q2 - Autoencoder

Implement a simple feedforward antoencoder for images using PyTorch from scratch. Your autoencoder should have **two hidden layers** for both the encoder and decoder. The latent space should have size 64. You may choose the size for the hidden layers. Please use `MIE524_A2.ipynb` to complete this question.

[1] `https://www.kaggle.com/datasets/hugodarwood/epirecipes`

[2] `https://en.wikipedia.org/wiki/Root-mean-square_deviation`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`

[4] `https://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-tree-regression`

[5] `https://scikit-learn.org/stable/modules/tree.html`

Task

a) Implement `my_AutoEncoder` class using **MSE** loss and **Adam** optimizer. Complete the code to train the autoencoder on the provided CIFAR10 data for 5 epochs. Report on the final MSE.

b) Extract the trained latent representations of the training data. Train a simple scikit-learn's random forest model on the representations. Evaluate your random forest model using the provided test data and report on the accuracy and f1 scores. How does this random forest model compare to the random forest models we have seen during the lab? Provide a short explanation of why you think it performs better or worse (no more than 5 sentences). Write your answers the provided text box.

c) Retrain your autoencoder with three different latent space dimensions and compare the performance of the different models.