Assignment 1
Machine Learning Spring 2023

- DEADLINE: One week from the start date.
- No late submissions would be entertained.
- Plagiarism will be handled very strictly. Do not share your work with anyone. Both matching assignments will be marked zero.

Task1: Regression

The goal in this task is to develop a fully connected neural network that can predict compressive strength of concrete given 8 features using regression.

You are given the following files:

- **TrainData.csv**: It contains 824 training examples. Each row contains 8 values.
- **TrainLabels.csv**: This file contains true labels for the examples in TrainExamples.csv
- **TestData.csv**: This file contains test examples.
- **TestLabels.csv:** This file contains true labels for the examples in TestExamples.csv
- You can load train and test data using the following code:

```
Xtr=np.loadtxt("TrainData.csv")
Ytr=np.loadtxt("TrainLabels.csv")
Xts=np.loadtxt("TestData.csv")
```

Sub-tasks:

i. Develop a **multi-layered** neural network (using Pytorch). Train it using the given training data.

ii. Optimize hyper parameters for the model using backpropagation.

iii. Use Root Mean Squared Error (RMSE) as loss function. (Give Formula for loss function)

iv. After choosing the best design parameters, use the complete training dataset to train the final model. Dump the model in the file named "rollno_Model". You can take help from https://pytorch.org/tutorials/beginner/saving_loading_models.html for saving your model.

v. Create a file rollno_test.py. It should read the test examples from TestExamples.csv.

- **DO NOT CHANGE THE ORDER OF EXAMPLES**.
- Generate predictions for the test examples and save them to a csv file named "<your roll no.>_Predictions.csv". You can use the following code to save your results:

```
np.savetxt("MSDS20018_Predictions.csv", Yts)
```

- **Additional task: Add the use of L1 norm weights as regularization and show its impact using the learning curve. Pick the best lambda parameter.**

You have to submit the following files:

- Files containing the code used for parameter selection.
- Training.py: The file containing the code to train and dump the final model using the chosen hyper parameters.
- Testing.py: The file in which you load the saved model and generate and save predictions for test data.
- <your roll no.>_Predictions.csv: The csv file containing your predicted labels for the **test set.**

A pdf report that contains the following information:

i. Network design choices (number of layers and size of each layer) and experimental setting (batch size, learning rate, momentum, optimizer, number of epoches) did you use for the final model? What other network design choices did you try?

ii. Plot training loss variation with increasing epochs.

iii. Multiple training cycles in neural networks can result in different final model weights and hence different predictions. Take at least 5-runs and report the mean and standard deviation of the RMSE. Plot test RMSE variation with epochs.

iv. Comment if you observe over-fitting, under-fitting, and experiment with varying learning rates.

Task2: Classification

The goal in this task is to develop a machine learning model that can classify between images of T-shirts and dress-shirts.

You are given the following files:

- **TrainData.csv**: It contains 12000 training examples. Each row contains 784 values. The dataset has been derived from Fashion-MNIST dataset. Each example is a flattened 28x28 pixel gray-scale image. You can reshape the examples to visualize what each image looks like.
- **TrainLabels.csv**: This file contains true labels for the examples in TrainExamples.csv
- **TestData.csv**: This file contains test examples.
- **TestLabels.csv:** This file contains true labels for the examples in TestExamples.csv
- You can load train and test data using the following code:

```
Xtr=np.loadtxt("TrainData.csv")
Ytr=np.loadtxt("TrainLabels.csv")
Xts=np.loadtxt("TestData.csv")
```

- To visualize an example (say training example at index 30, you can use the following code):

```
import matplotlib.pyplot as plt
plt.imshow(Xtr[30].reshape([28,28])
```

Sub-tasks

i. Write a method named extract_features which are the pixel values.
ii. You are supposed to train a model using your extracted features.
iii. Use fully connected neural network.
iv. Implement K-Nearest Neighbor classification technique, compare its performance with neural network.
v. Optimize experimental settings for the models. Since the dataset is balanced, you can use classification accuracy as the performance metric.
vi. Use binary cross entropy as loss function.
vii. Additional task: Add the use L1 of weights as regularization and the show its impact using the learning curve. Pick the best lambda parameter.
viii. After choosing the best parameters, use the complete training dataset to train the final model. Dump the model in the file named "myrollnumber_Model.pkl". You can take help from https://scikit-learn.org/stable/modules/model_persistence.html for saving your model.
ix. Create a file myrollno_test.py. It should read the test examples from TestExamples.csv. **DO NOT CHANGE THE ORDER OF EXAMPLES**. Generate predictions for the test examples and save them to a csv file named "myPredictions.csv". You can use the following code to save your results:

```
np.savetxt("MSDS20018_Predictions.csv", Yts)
```

You have to submit the following files:

- Files containing the cross-validation code used for design parameters selection.
- Training.py: The file containing the code to train and dump the final model using the chosen hyper parameters.
- Testing.py: The file in which you load the saved model and generate and save predictions for test data.
- MSDS20018_Predictions.csv.: The csv file containing your predicted labels for the **test set.**

A pdf report that contains the following information:

i. Network design choices (number of layers and size of each layer) and experimental setting (batch size, learning rate, momentum, optimizer, number of epoches) did you use for the final model? What other network design choices did you try?

ii. Plot training loss variation with increasing epochs.

iii. Multiple training cycles in neural networks can result in different final model weights and hence different predictions. Take at least 5-runs and report the mean and standard deviation of the RMSE.

iv. Plot test RMSE variation with epochs. Comment if you observe over-fitting, under-fitting, and experiment with varying learning rates.

NOTE:

You must submit two folders named Regression and Classification and then zip these before submitting in a folder named after your roll number. Do not submit datasets.