CS6220 BDS 2020 Fall (Released on Monday of Aug. 24, 2020)

Homework Assignment 1
(Programming Category)

Student Name:______________________

Student Session: cs6220-A

You are given three types of programming problems in the first homework assignment.

- **Problem 1** is on learning with hand-on experience of using a deep learning (DL) model from a popular DL framework. (Pages 2-8)
- **Problem 2** is on data-driven deep learning system tuning. (Pages 8 – 14)
- **Program 3** is an application software development problem that requires you to write code. (Pages 14-15)

Each problem is given you two or more subproblems as options and you only need to choose one option (subproblem) from one of the three Problems as your first homework. This provides sufficient diversity for students with different knowledge and educational backgrounds.

For those of you who is the beginner of machine learning to choose the shallow learning algorithm in this first assignment (Option 1.1 from Problem 1) and you will have opportunity to choose deep learning algorithms such as different neural network algorithms in the subsequent assignments.

For those of you who would like to get datasets from real world and then employ one of your favorite algorithms to produce prediction model and prediction results, you are given the problem 3. For Problem 3, feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

Post Date: on Monday of Week 2
Due Date: midnight on Friday of Week 3

Problem 1. Hand on experience with Supervised Learning (Traditional v.s. Deep Neural Network)

This assignment gets you some hand-on experience with either traditional machine learning or deep neural network learning as a user or an application developer.

(1)Software download.
For this homework, you may choose one of your favorite machine learning (ML) algorithms, such as SVM, Decision Tree (Random Forest), or neural network model, such as DenseNet, ResNet, LeNet, etc. from a popular machine learning framework of your choice.

The popular software frameworks are Scikit-Learn, TensorFlow (google), Caffe, Torch/PyTorch, OpenCV (Intel), CNTK (Microsoft).
https://scikit-learn.org/
https://www.tensorflow.org.
https://caffe.berkeleyvision.org
http://torch.ch or https://pytorch.org
https://docs.opencv.org/master/index.html
https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine
https://keras.io

You may also use Keras, a high-level neural networks API, written in **Python**, for this homework as it is capable of running on top of TensorFlow, CNTK, Theano, etc..

In the rest of the problem description, I will use TensorFlow (TF) to describe the homework but you can replace TF with one of your favorite ML/DL frameworks.

(2) Dataset Preparation
You are asked to use an image dataset of at least two types of objects, such as Cat v.s. Dog, to train a machine learning model to perform a binary classification task, with high accuracy, using either a neural network model or a traditional ML algorithm.

You may find such image datasets in the public domain such as Kaggle, ImageNet, or creating your own.

Some example datasets are
(i) Color rich single object images like color photos of cats and dogs, or color photos of people. Such as Kaggle (https://www.kaggle.com/c/dogs-vs-cats), or CIFAR10, (https://www.cs.toronto.edu/~kriz/cifar.html), or ImageNet

(https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet). You may also choose simple grey-scale images. One example is MNIST. (http://yann.lecun.com/exdb/mnist/)

(ii) If you want to use a traditional ML algorithm, such as SVM, Decision Tree (Random Forest), you will need to find the image dataset with CSV format instead of raw pixel format. For example, you can find MNIST CSV or AT&T Face CSV from https://github.com/git-disl/GTDLBench under datasets.

(iii) You are asked to create two versions of the same dataset. Here are some example approaches: You can create two versions by using two different resolutions (e.g., 64x64, 32x32, or 28x28). You may also create two datasets with different sizes, such as two digits from MNIST dataset of size 1000, each digit has 500 images, or 2000, each digit has 1000 images.

Training v.s. Test data partition: You can use 80:20, namely using 80% of the images as the training dataset (say 40 pictures of cat and 40 pictures of dog images in a dataset of 100) for training and 20% (10 pictures of cat and 10 pictures of dog images) as the test dataset. Once you learned to use TensorFlow to train the model, you will know training a model over 100 images, 1000 images or 5000 images will be very fast.

This programming homework provides two options. You are required to choose one option.

If you have no hand-on experience with any deep learning software framework, then you are recommended to choose the first option of this problem.

Problem 1.1 Hand-on Experience for Training and Testing a K-class Image Classifier (K=2 or 5 or 10 …)

Requirement.

(1) If you choose traditional ML algorithms from e.g., Scikit-Learn repository Choose to use a conventional machine learning algorithm, such as Decision Tree (Random Forest), Support Vector Machine SVM to train a binary classifier. then these algorithms require input dataset is N-dimensional vector representation. Hence, you will need to either download the image dataset that has CVS format for each image or use an online function to convert an image of resolution, say 32x32 into a CVS format. Then go to step (3).

(2) For those of you who wish to gain experience with a deep Neural Network algorithm, such as CNN, ResNet, LeNet, etc., you will need to choose

your favorite deep learning framework, say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(3) Using 80% of the images as the training dataset (say 40 pictures of cat and 40 pictures of dog) and 20% (10 pictures of cat and 10 pictures of dog) as the test dataset.

(4) You are asked to show how TensorFlow was used to train a binary classification model, from input to training in CNN layers, to the output of the trained K-class model, including the storage size of the model in MB, the training accuracy and training time.

If you use SVM or Decision Tree/Random Forest, then you are asked to show how your training makes progress (see detail in the deliverable)

(5) Then you use your testing images to test your binary classifier trained by TensorFlow CNN or by a traditional ML algorithm, and report the average test accuracy and test time for the two datasets you chose (either different dataset sizes or two different resolutions).

(6) **Outlier Test Scenario**. When performing testing on your binary classifier that you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 5 or more images that do not belong to the binary class classification task. For example, if you choose dog and cat as your binary classification task, then create 5 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. For each of the 5 outlier queries, run 3 times each and report your results, such as if 3 times of the same outlier query returns the same prediction result or not and you are asked to elaborate on why.

(7) **Option:** If you have time and still want to learn more on CNN, then you are asked to use one of the deep learning frameworks of your choice, e.g., TensorFlow, to produce another two K-class CNN classifier with K= 10 for example, and two versions of the same dataset with each class has the same #training examples as your binary classifier. You are asked to measure the training and testing accuracy and time and compare with your binary CNN classifier (K=2).

Hint: Here are some example datasets.

MNIST dataset. http://yann.lecun.com/exdb/mnist/.
USPS dataset. https://www.kaggle.com/bistaumanga/usps-dataset
Traffic Sign Recognition. https://www.kaggle.com/c/traffic-sign-recognition
LISA dataset: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
p2-trafficsigns. https://github.com/ vxy10/p2-TrafficSigns
The face database.
https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
You can also find most of them at
https://git-disl.github.io/GTDLBench/datasets/

Deliverable:

(1) provide URL of your open source code package and dataset download.
(2) Screen shots of your execution process/environments
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
  a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
  b. choose and show 2 sample images per class for all K classes in each of the four datasets.
  c. the training v.s. testing data split ratio and size used in your classifier training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
  d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.

     If you are using traditional ML algorithm such as SVM or Decision Tree/Random Forest, then you are asked to show your training process in terms of #iterations and display the output of training in two intermediate stage (#iterations) and the final stage when the training converges (terminates), report the storage size of the model, the training accuracy and training time.

  e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence.

     If you use traditional ML algorithm such as SVM or Decision Tree (Random Forest), then you are asked to show the default configuration of your algorithm, such as the input data structure, the default error threshold or the total #iterations, and the model code and the storage size of the model.

(4) **Output Analysis:** *Report the performance comparison and analysis of your K-class CNN classifier:*

a. You are asked to provide a table (ideally in an excel file, but not required) to compare the two classifiers (each is trained from one of the two datasets) in terms of training time, training accuracy, testing time and testing accuracy.
b. You are asked to record the trained model size in MB for the two classifiers in the above table.
c. You are asked to add the test accuracy and time for your outliner dataset in requirement (5) by testing using the two classifiers.
d. You are asked to make at least three observations from your experimental comparison of the two classifiers.

Hint: If you also did (7) the option, then your comparison will be four classifiers.

Problem 1.2. Improving CNN Classifier Performance with Optimizations

Requirement.

(1) Choose your favorite deep learning framework, say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(2) Using 80% of the images as the training dataset (say 40 pictures of cat and 40 pictures of dog) and 20% (10 pictures of cat and 10 pictures of dog) as the test dataset.

(3) You are asked to show how TensorFlow was used to train a binary classification model, from input to training in CNN layers, to the output of the trained K-class model, including the storage size of the model in MB, the training accuracy and training time.

(4) Then you use your testing images to test your binary CNN classifier trained by TensorFlow and report the average test accuracy and test time.

(5) **Outlier Test Scenario**. When performing testing on your binary CNN classifier that you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 5 or

more images that do not belong to the binary class classification task. For example, if you choose dog and cat as your binary classification task, then create 5 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. For each of the 5 outlier queries, run 3 times each and report your results, such as if 3 times of the same outlier query returns the same prediction result or not and make an attempt to elaborate on why.

(6) Change the deep neural network you used to train the above binary classifier (say LeNet) to another 3 different models, such as ResNet32, ResNt64, VGG16, and using the same training dataset to train another three binary classifiers and compare their performance in training time, training accuracy, test time and test accuracy.

Hint: Here are some example datasets.

MNIST dataset. http://yann.lecun.com/exdb/mnist/.
USPS dataset. https://www.kaggle.com/bistaumanga/usps-dataset
Traffic Sign Recognition. https://www.kaggle.com/c/traffic-sign-recognition
LISA dataset: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
p2-trafficsigns. https://github.com/ vxy10/p2-TrafficSigns
The face database.
https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
You can also find most of them at
http://yanzhaowu.me/GTDLBench/datasets/.

Deliverable:

(1) provide URL of your open source code package and the dataset download.
(2) Screen shots of your execution process/environments
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
    a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
    b. choose and show 4 sample images per class for all K classes in the dataset.
    c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 100 images with 8:2 training v.s. testing ratio, then 80 for training and 20 for testing.)
    d. You are asked to record the default neural network (NN) model, such as LeNet, or ResNet, or DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the configuration you use to train your CNN classifiers on the two different datasets.

e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence.

(4) **Output Analysis:** *Report the performance comparison and analysis of your K-class CNN classifier:*

a. You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (two are trained on each of the two datasets, with one optimized CNN classifier for each dataset), in terms of training time, training accuracy, testing time and testing accuracy.
b. You are asked to record the trained model size in MB for both classifiers in the above table for all 4 CNN classifiers
c. You are asked to make at least three observations from your experimental comparison of the 4 CNN binary classification models.

Problem 2 Learning Deep Neural Networks for Performance Improvement

This assignment provides you with experience in tuning machine learning model configurations for training a DNN model with high accuracy on a benchmark dataset or a dataset of your own choice.

(1)Software download.
For this homework, you may choose one of your favorite machine learning (ML) algorithms, such as DenseNet, LeNet, etc., from a popular deep learning (DL) framework of your choice.

The popular DL frameworks are TensorFlow (google), Caffe, Torch/PyTorch, OpenCV (Intel), CNTK (Microsoft).
https://www.tensorflow.org.
https://caffe.berkeleyvision.org
http://torch.ch or https://pytorch.org
https://docs.opencv.org/master/index.html
https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine
https://keras.io

You may also use Keras, a high-level neural networks API, written in **Python**, for this homework as it is capable of running on top of TensorFlow, CNTK, Theano, etc..

In the rest of the problem description, I will use TensorFlow (TF) to describe the homework but you can replace TF with one of your favorite ML/DL frameworks.

(2) Dataset Preparation
You are asked to use an image dataset of at least two types of objects, such as Cat v.s. Dog, to train a neural network to learn a binary classification task, such as detecting cat or dog with high accuracy.

You may find such image datasets in the public domain such as Kaggle, ImageNet, or creating your own.

Some example datasets are

(i) Color rich single object images like color photos of cats and dogs, or color photos of people. Such as Kaggle (https://www.kaggle.com/c/dogs-vs-cats), or CIFAR10, (https://www.cs.toronto.edu/~kriz/cifar.html), or ImageNet (https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet). You may also choose simple grey-scale images. One example is MNIST. (http://yann.lecun.com/exdb/mnist/)

(ii) You are asked to create two versions of the same dataset. Here are some example approaches: You can create two versions by using two different resolutions (e.g., 64x64, 32x32, or 28x28). You may also create two datasets with different sizes, such as two digits from MNIST dataset of size 1000, each digit has 500 images, or 2000, each digit has 1000 images.

Training v.s. Test data partition: You can use 80:20, namely using 80% of the images as the training dataset (say 40 pictures of cat and 40 pictures of dog images in a dataset of 100) for training and 20% (10 pictures of cat and 10 pictures of dog images) as the test dataset. Once you learned to use TensorFlow to train the model, you will know training a model over 100 images, 1000 images or 5000 images will be very fast.

This programming homework provides two options. You are required to choose one option.

Problem 2.1 Comparing TensorFlow Trained CNN classifier with varying dataset complexity

Goal. The goal of this problem is to get you to learn how different complexity and variations of datasets may impact on the performance of deep neural network trained classifiers.

You only need to work with an off-shell deep learning software framework such as TensorFlow, PyTorch or one of your favorite DL framework. It does not require you to write or modify any program code. Your job is simply to use TensorFlow to generate 4 CNN classifiers, one per dataset configuration. Then compare their training and testing performance (time and accuracy).

For a beginner, you are recommended to train a binary classification model. Feel free to create a classification task with k classes (k=2, 10, 100 for example), depending on your machine capacity for running the TensorFlow training.

Requirements: You are asked to vary a set of dataset-specific parameters in your training phase and to make observations and comparisons:

(1) Datasets:
   a. the input datasets in terms of varying volumes (at least one small and one 10x larger). The number of images per class should be no smaller than 100.
   b. the input datasets in terms of varying resolutions (at least two different resolutions, e.g., 28x28, 64x64)

   You may name your datasets as D100, D1000, D-28x28, D-64x64.
   Hint: make sure you divide your dataset into training and testing.

(2) For each of the four datasets from the same domain (say color images from ImageNet, e.g., cat v.s. dog), you are asked to use TensorFlow to train 4 different CNN classifiers and to make comparison on their training time, accuracy and test time and accuracy

(3) **Outlier Test Scenario**. When performing testing on each of the 4 CNN classifiers you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 images that are outside the binary classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set in your report. Suggestion: For each of these 10 outlier photos, perform each query 5 times and report whether the results are the same or different and analyze why.

(4) Optional task 1 (only if you have more time and interest): You may choose one or two of the following 4 subtasks in the deliverable of Problem 1.1.
   a. Choose the one binary CNN classifier that has the highest test accuracy among the 4.
   b. For this CNN classifier and its training dataset, you are asked to use two different ratios of training size v.s. testing size and make a comparison. Such as 7:3. 9:1 …
   c. For this CNN classifier and its training dataset, you are asked to change the default setting of #epoch to be 5x and 10x larger. Report your comparison.
   Note: if your training dataset has B mini-batches, then you will need about #iterations = #epochs x B.

d. For this CNN classifier and its training dataset, you can create two models using two different weight filters, such as 3 by 3 and 5 by 5 for an image grid size of 28 x 28.

Deliverables:

(1) Provide URL of your open source code packages

(2) Example screen shots of your execution process/environments

(3) Input Analysis: *Use a table to report your setup parameters for* ***each*** *training model built*:

a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).

b. Choose 5 sample images and show each in their two resolution versions.

c. the training v.s. testing data split ratio and the size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)

d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, Densenet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for your training and testing experiments.

e. You are asked to report the error threshold used in the TensorFlow default configuration for backward propagation in each step. Usually it is documented for MNIST and CIFAR-10.

(4) **Output Analysis:** *Report the performance comparison and analysis of your 4 CNN classifiers*

(a) You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (each is trained from one dataset configuration), in terms of training time, training accuracy, testing time and testing accuracy.

(b) You are asked to record the trained model size in MB for each of the four classifiers in a table.

(c) You are asked to provide a table of the same format but this time your test is performed on your outlier test dataset in requirement (4).

(d) You are asked to make at least three observations from your experimental comparison of the 4 CNN classifiers.

Problem 2.2 Comparing TensorFlow Trained CNN classifier with varying training parameter configurations

Goal. The goal of this problem is to get you to learn the impact of different CNN training configurations on the performance of deep neural network trained classifiers.

You only need to work with an off-shell deep learning software framework such as TensorFlow as an application developer. It does not require you to write or modify any program code. But you will need to make changes to the TensorFlow CNN configuration default on a small selection of parameters.

Your job is to compare the TensorFlow default setting with at least 3 alternative configurations you made and generate 4 CNN classifiers as executables for performance comparison on training and testing performance (time and accuracy).

For simplicity, you are recommended to create either a binary classification task or use a classification task of 10 classes. If you have powerful GPU on your machine for running the TensorFlow training, you may choose a classification task with a larger #classes (e.g., 2, 10, 100).

Requirements:

(1) **Dataset:** You are asked to create a color image dataset of no smaller than 100 images from ImageNet, or CIFAR-10, CIFAR-100 or from your own collection, according to your classification task. For example, for a binary task, you may collect 50 (or 100, 500, 1000) photos of one object and 50 (or 100, 500, 1000) photos of another object.

(2) CNN training by default configuration. You are asked to use TensorFlow default configuration for CIFAR-10 (which comes with the download) to train a CNN classifier for your binary classification task. Record training time and accuracy and testing time and accuracy.

(3) You are asked to make three modifications on the TensorFlow default configuration you used to train your first binary classifier.
   a. Change the default DNN model (e.g., change LeNet to ResNet-52)
   b. Change the default CNN structure of a given DNN model, say LeNet from CNN-2 (2 hidden layers) to CNN-5.
   c. Varying the #epochs used for training CNN classifier in 4-5 values, scuh as 1 epoch (which process the dataset only one pass), 2 epochs, 5 epochs, 10 epochs. Make sure to include the default one you used to produce the CNN classifier using default configurations in (2). Record the performance in training/testing time and accuracy in a table.

(4) **Outlier Test Scenario**. When performing testing on each of the 4 CNN classifiers you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 images that do not belong to the binary classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test

on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set.

(5) Optional tasks (only if you have more time and interest, you may choose to do c) or d) instead of both):

a) Choose the one binary CNN classifier that has the highest test accuracy among the 4 you have.
b) For this CNN classifier and its training dataset, you are asked to use 2x, 5x increase of your datasets. Then run test performance on your trained model
c) Retain your CNN classifier with the training and test split on the dataset of 2x bigger, 5x bigger. Compare the performance of these 3 classifiers in terms of training/testing time and accuracy in a table.
d) For this CNN classifier and its training dataset, you are asked to evaluate the impact of different resolution scales or different sizes of the input image set on the CNN classifier.

Deliverables:

(1) Provide URL of your open source code packages
(2) Example screen shots of your execution process/environments for CNN classifier training and testing scenarios.
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
   a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
   b. choose and show 5 sample images per class.
   c. the training v.s. testing data split ratio and size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
   d. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.
   e. You are asked to report the default error threshold used in the TensorFlow default configuration for convergence. Usually it can also be found from some documentations for MNIST and CIFAR-10.

(5) **Output Analysis:** *Report the performance comparison and analysis of your 4 CNN classifiers*
   a. You are asked to provide a table (ideally in an excel file, but not required) to compare the 4 CNN classifiers (each is trained from one

dataset configuration), in terms of training time, training accuracy, testing time and testing accuracy.

b. You are asked to record the trained model size in MB for each of the four classifiers in a table.
c. You are asked to provide a table of the same format but this time your test is performed on your outlier test dataset in requirement (4).
d. You are asked to make at least three observations from your experimental comparison of the 4 CNN classifiers.

Problem 3: Using ML or Deep Learning frameworks to Solve a real world big data problem.

This problem is designed for those students who are familiar with ML and AI algorithms and may have prior hand on experiences with ML or DL software frameworks, such as Spark ML library, Hadoop Mahout, Scikit-learn (http://scikit-learn.org/), Weka (http://www.cs.waikato.ac.nz/ml/weka/) or R https://bigdata-madesimple.com/10-r-packages-machine-learning/.

You are asked to work on solving a data analytic problem using your favorite ML or DL software frameworks. You may choose your own dataset and the problem on mining your dataset or use the following problems and the associated datasets:

Problem 3.1 Mining the Kaggle million songs dataset

The million songs dataset is a big data mining challenge. One of the earlier Kaggle challenges. It contains (1) the full listening history for 1M users, (2) half of the listening history for 110K users (10K validation set, 100K test set). You are asked to predict the missing half.

http://www.kaggle.com/c/msdchallenge
Statistics of the dataset:

- Large Data Set
  - 1,019,318 users
  - 384,546 MSD songs
  - 48,373,586 (user, song, count)
- Kaggle Competition: offline evaluation
  - Predict songs a user will listen
  - Training: 1M user listening history
  - Validation: 110K users

This is a well-known big data challenge and there are software postings in the Public domain. Feel free to use any of them you found helpful.

Deliverable:

(1) URL of the dataset
(2) The subset of the data you used in your program or model training and testing.
(3) The software packages and tools or library you use in your programs
(4) The open source code you leveraged in developing your MillionSongMiner.
(5) The experiments you conducted to report your mining accuracy, ideally over three different datasets extracted from the million-songs dataset in three different sizes (e.g., 2x, 5x, 10x of your initial smaller dataset, say 1000, or 10,000 songs.)
(6) Analysis of your hand-on experiences, with three lessons learned or three observations you wish to make.

Problem 3.2 Mining the White House visitor Log dataset

The White House Visitor Log dataset available at http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Released-0827.csv . The attributes in this dataset are described at: http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Key-1209.txt . A spreadsheet of the data can be found at: http://www.whitehouse.gov/briefing-room/disclosures/visitor-records .

You are required to write an efficient program (such as using MapReduce) to find the following information:

(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST, NAMEMID) to the White House.

(ii) The 10 most frequently visited people (visitee_namelast, visitee_namefirst) in the White House.

(iii) The 10 most frequent visitor-visitee combinations.

(iv) Some other interesting statistics that you can think of.

Throughout this programming assignment, do not limit your programs to run with a single design choice (such as one reduce task only).

Deliverable:
(1) URL of the dataset
(2) The software packages and tools or library you use in your programs
(3) The open source code you leveraged in developing your visitor log miner
(4) The experiments you conducted to report your mining accuracy and time, ideally over three different datasets in three different sizes (e.g., 2x, 5x, 10x of your initial dataset.)
(5) Analysis of your hand-on experiences, with three observations you wish to make.