Mini-project 1

- This mini-project is to be completed in groups of up to three. You will submit your assignment on MyCourses as a group. You must register your group on MyCourses.
- You are free to use libraries with general utilities, such as matplotlib, numpy, pandas and scipy for Python. **However, you should implement everything (e.g., the models and evaluation functions) yourself, which means you should not use pre-existing implementations of the algorithms or functions as found in SciKit learn, etc.**

Introduction

The goal of this homework is to gain experience on implementing the well-known linear classifier *logistic regression*, from scratch, as are discussed in the class lectures. You are required to perform two-class classification on two datasets Hepatitis and Bankruptcy by using Logistic Regression classifier. Datasets are attached.

Dataset Hepatitis consists of various features for hepatitis patients. The main aim of the Hepatitis dataset is to discriminate two classes: survivors and patients for whom the hepatitis proved terminal. For the purpose of this project we will use Hepatitis.csv to build observation matrix X and its associated class label vector y.
Dataset Bankruptcy contains various econometric attributes for bankruptcy status prediction. For the purpose of this project we

will use Bankrupcy.csv to build observation matrix X and its associated class label vector y.

Your Tasks

- You need to download the corresponding data files discussed in Section Introduction and load each dataset into numpy objects (arrays or matrices) in Python.
- Perform some statistical analysis on the datasets e.g. what are the distribution of the two classes? what are the distribution of some of the features? etc. You may visualize your results using histogram plots.
- Implement the linear classifiers logistic regression from scratch, by following the equations discussed in class lectures and apply your implemented algorithms to the datasets. To clarify, for example, basic functions such as transpose, shuffle and panda's .mean() and .sum() are okay to be used, but using the train_test_split from sklearn is **not** ok.
    - You are free to implement the method in anyway you want, however we recommend to implement both models as python classes (use of constructor is recommended). Each of your model class should have at least two functions: `fit` and `predict`. The function `fit` takes the training data X and its corresponding labels vector y as well as other hyperparameters (such as learning rate) as input, and execute the model training through modifying the model parameters (i.e. W). `predict` takes a set of test data as input and outputs predicted labels for the input points. Note that you need to convert probabilities to binary 0-1 predictions by thresholding

the output at 0.5. The ground-truth labels should also be converted to binary 0-1.

- Define a function `Accu-eval` to evaluate the models' accuracy. `Accu-eval` takes the predicted labels and the true labels as input and outputs the accuracy score.
- Implement k-fold cross validation from scratch as a Python class. Use 10-fold cross validation to estimate performance in all of your experiments and you should evaluate performance using accuracy. For model selection, if you have T different models, you should run 10-fold cross validation T times and compare the results.
- At least, complete the followings: test different learning rates for your logistic regression, discuss the run time and accuracy of your logistic regression on both datasets, explore if the accuracy can be improved by a subset of features and/or by inserting new features to the dataset.

Report (Maximum 5 pages of content and Maximum 2 pages of references.):

We are flexible on how you report your results, but you must adhere to the following structure:

- **Abstract** (100-250 words): provide a summary of the project task and highlight your most important findings. For example, include sentences like "In this project we investigated the performance of linear classification models on two benchmark datasets", "We found that the logistic regression approach was achieved worse/better accuracy when some features were removed, added."

- **Introduction** (at least one paragraph): Summarize the project task, the two datasets and your important findings. This is similar to the abstract but you should provide more details.
- **Datasets** (at least one paragraph): Briefly describe the dataset and its characteristics such as number of samples, features, type of features, etc. Describe the new features you come up with in detail. *Note:* You do not need to explicitly verify that the data satisfies the i.i.d. assumption (or any of the other formal assumptions for logistic regression).
- **Results**: Describe the results of all your experiments; including tables and/or figures will be a great help when discussing your results and findings. At a minimum discuss how the logistic regression performance (e.g. convergence speed) depends on the learning rate and demonstrate if the new features and/or the feature subsets you used will improve the performance. You should properly cite and acknowledge previous works/publications that you use or build on.
- **Discussion and conclusion**: Discuss and summarize the key takeaways from the project and possible directions for future investigation.
- **Statement of Contributions** (1-3 sentences) State the breakdown of the workload across the team members.
- **Appendix** To facilitate the grading process, attach the codes for your implementations to the end of your report. This does *not* count towards the page limit of the report.

  Please keep your analysis as short as possible while still covering the requirements of the assignment. The analysis report is limited to 5 pages (single-spaced, minimum font size

of 11 and 1 inch minimum margin each side). We highly recommend to use LaTeX for preparing your report. We recommend to use NeurIPS 2020 template. Your report should look technical. Imagine you are writing a paper for a major machine learning conference.

Deliverables

- **report.pdf**: Your report (including Appendix) as a single pdf file.}
- **code.zip**: Your codes (e.g. .py, .ipynb, etc.) must work with Python 3.6 in Colab. Include a *readme* file and provide instruction for TA on how to replicate your results on Colab. All the results must be reproducible in Colab using the submitted *code.zip*. Points will be deducted if we have a hard time reading or understanding the structure of your code.

  The report should be self contained. TA's will do the grading mainly based on the `report.pdf`, and will not be obliged to consult the supplementary codes.

Evaluation

This assignment is out of 100 points.
Your report should be both thorough and concise. It will be judged based on its scientific quality including but not limited to:

- Does the report include all the required experiments?
- Is the report technically sound? (i.e. do the steps taken make sense? Are the results in an acceptable range?)

- How thorough/rigorous is the experimental validation?
- Is the report well-organized and coherent?
- Is the report clear and free of grammatical errors and typos?
- Does the report contain sufficient and appropriate references and related work?

All members of a group will receive the same mark.

To get an A grade you need to go beyond the requested steps/parts. For example, you might investigate different stopping criteria for the gradient descent in logistic regression or develop an automated approach to select a good subset of features. Try to offer and implement solutions for solving the classification problem better or increase the performance of your model. Or maybe explore the data in such a thorough manner that you can justify removing the features you might have removed. You do not need to necessarily do all/any of these, but you should demonstrate creativity, rigour, and an understanding of the course material in how you run your chosen experiments and how you report on them in your write-up.

About Colab

The easiest way to set up your environment is to use Colab. Colaboratory (also known as Colab) is a free Jupyter notebook environment from Google that runs in the cloud and stores its notebooks on Google Drive. It is more suitable for interactive jobs rather than long runs. It is free. Upload the data into your GoogleDrive. Create a Jupyter notebook. Mount your

GoogleDrive using the following commands and follow the instructions (authorization code, etc.).

```
from google.colab import drive
drive.mount('/content/gdrive')
```

You can read the data using the Pandas dataframes:

```
import pandas as pd, numpy as np
df_data = pd.read_csv('./dataset.csv')
```

You can see a glimpse of the training data using df_data.head(). It returns the first few rows of data.

Final remarks

You are expected to display initiative, creativity, scientific rigour, critical thinking, and good communication skills. You don't need to restrict yourself to the requirements listed above - feel free to go beyond, and explore further.
You can discuss methods and technical issues with members of other teams, but you cannot share any code or data with other teams. Any team found to cheat (e.g. use external information, use resources without proper references) on either the code, predictions or written report will receive a score of 0 for all components of the project.