# CS209 Machine Learning Lab

Note: 1. Python with scikit-learn library is recommended for implementing lab exercises.
2. Use UCI repository or any standard repository for dataset:
https://archive.ics.uci.edu/ml/index.php
3. Install Anaconda (Python v3.x); https://www.continuum.io/downloads.

**Exercise 0: Python Tutorials (you may use other programming language/tools of your choice)**

a) In this task you have to write a word count program (using IPython). Your program should read a text document. You should save your session and it should include Headings and comments at some important steps to explain the working of code.

b) Create a matrix A of dimensions n × m, where n = 100 and m = 20. Initialize Matrix A. Create a vector v of dimension m × 1. Initialize the matrix with a random values and vector with normal distribution using $\mu = 2$ and $\sigma = 0.01$ (use numpy).

Perform following operation on them:
i. Iterative multiply (element-wise) each row of matrix A with vector v and sum the result of each iteration in another vector c.
ii. Find mean and standard deviation of the new vector c.
iii. Plot histogram of vector c using 5 bins.

**Exercise 1: Linear Regression**

a) Implement the linear regression using least square method in your own code. Also implement the linear regression by using existing library (scikit-learn). Compare the performance of both implementations.
b) Implement the linear regression using Gradient Descent method in your own code. Also implement the linear regression by using existing library (scikit-learn). Compare the performance of both implementations.

Note: You can use any dataset from UCI repository.

**Exercise 2: Logistic Regression**

Implement the logistic regression in your own code. Also implement the logistic regression by using existing library (e.g. scikit-learn). Compare the performance of both implementations and show the results.

Use the following evaluation metrics: (a) Mean Squared Error (MSE) (b) Root Mean Squared Error (RMSE) (c) Mean Absolute Error (MAE) (d) R Squared ($R^2$).

**Exercise 3: Linear Discriminant Analysis**

Implement the Linear Discriminant Analysis algorithm in your own code for class classification. Also implement the Discriminant Analysis algorithm by using existing library (e.g. scikit-learn). Compare the performance of both implementations and show the results.

**Exercise 4: Implement Decision Tree**

**Classification Datasets:** You can use one of the two datasets (or optionally, both datasets).
(a) Car Evaluation dataset D1: Target attribute safety :{low, med, high}.
https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
(b) Iris dataset D2: Target attribute class :{Iris Setosa, Iris Versicolour, Iris Virginica}.
https://archive.ics.uci.edu/ml/datasets/Iris

Implement Decision Tree. In this task you will implement a decision tree. As a starting point you can implement your complete decision tree model for classification tasks.

You have to split data into two parts train and test (70% and 30% respectively). Using the train data you will build a decision tree. Use Cross Entropy as a Quality-criterion. You have to provide information on the learning process that includes.

1. Define an appropriate stopping criteria i.e. max depth, gain is too small or reduction in cost is small
2. At each decision step (or split) present the probability of each class using histogram (properly labeled figure)
3. At each decision step, plot the Cross Entropy of each attribute.
4. Note down the Information Gain at each new node created, you can store it in node structure or class. Display it at the end.
5. Print your tree using a breath first tree traversal. (you can also print node hierarchical level, information gain and decision rule, etc).
6. On a test set measure the cross entropy loss (i.e. logloss, note that this time problem is not binary classification).

**Exercise 5: Random Forest Regression**

Implement the random forest regression model. Evaluate your model using k-fold cross validation. Use the following evaluation metrics: (a) Mean Squared Error (MSE) (b) Root Mean Squared Error (RMSE) (c) Mean Absolute Error (MAE) (d) R Squared ($R^2$).

**Exercise 6: Naïve Bayes Classification:**

Dataset:

You can use iris dataset (i.e. D2 dataset in exercise 2) or any dataset of your choice.

Implement Naïve Bayes Classification algorithm and show the results. Split data into a train and a test split (70% and 30% respectively).

**Exercise 7: K-Nearest Neighbor (KNN) Classification:**

Datasets:

1. Classification Datasets: You can use one of the two datasets (or optionally, both datasets).
(a) Iris dataset D2 (i.e. D2 dataset in exercise 2):
Target attribute class :{Iris Setosa, Iris Versicolour, Iris Virginica}.
(b) Wine Quality dataset D3 : Target attribute quality:{0 to 10}.
https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Implement K-Nearest Neighbor (KNN) Classification:
Your task is to implement KNN Classification algorithm. To implement KNN you have to

- Split data into a train and a test split (70% and 30% respectively).
- Implement a similarity (or a distance) measure. To begin with you can implement the Euclidean Distance.
- Implement a function that returns top K Nearest Neighbors for a given query (data point).
- You should provide the prediction for a given query (use majority voting for classification).
- Measure the quality of your prediction. [Hint: You have to choose a quality criterion].

**Exercise 8: Implement K Means clustering algorithm**

Document dataset:
(a) IRIS dataset D4: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ multiclass/iris.scale
(b) rcv1v2 (topics; subsets D5:

https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/ datasets/multilabel.html#rcv1v2(topics;subsets)

1: Implement K Means clustering algorithm (using any software library). You should use D4 or D5 datasets. Also, you should also choose a criterion for selecting an optimal value of k (number of clusters).
2: Cluster the data set D4 or D5 using your own implementation of K-means algorithm. Show the results.

**Exercise 9: Classification using (Deep) Neural Networks**

Implement simple Neural Network model in your own code for classification task. Also implement the Popular Deep Learning models (CNN, RNN) by using existing library. Evaluate the developed models and compare the performances.

**Exercise 10: A spam filter using SVM:**
**UCI dataset:** Spambase D6: https://archive.ics.uci.edu/ml/datasets/Spambase

Build a spam filter using a pre-processed dataset. A spam filter to classify an email to be Ham or Spam, using the content of an email as features. Build a basic spam filter using SVM. You have to use libsvm https://github.com/cjlin1/libsvm/tree/master/python.
libsvm accepts data in a libsvm format.
Each data row in a libsvm format is given as
<label> <index1>:<value1> <index2>:<value2> ...
Convert dataset D3 into a libsvm format. Follow the readme document given on the libsvm link to see how you can use it to solve your problem. You have to learn a spam classifier on train part of the dataset and evaluate it on test dataset. Also optimize the hyper parameter i.e. value of C. [hint: when choosing the range of hyperparameter its always useful to check a diverse range i.e. C ={1,2,3,4} is not a good range to check for optimal value, you might want to check a broader range going from 0.1 to 100 etc.]. Present your results in form of graphs and tables, listing details. You have to choose a quality criterion according to the given problem i.e. classification.

[Note:] If you are not able to use libsvm you can replace it with scikit learn. But you have to convert your data into libsvm format.
---