1. **Supervised, Semi-Supervised, and Unsupervised Learning**

   (a) Download the Breast Cancer Wisconsin (Diagnostic) Data Set from: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`. Download the data in `https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data`, which has IDs, classes (Benign=B, Malignant=M), and 30 attributes. This data has two output classes. Use the first 20% of the positive and negative classes in the file as the test set and the rest as the training set.

   (b) **Monte-Carlo Simulation**: Repeat the following procedures for supervised, unsupervised, and semi-supervised learning $M = 30$ times, and use randomly selected train and test data (make sure you use 20% of both the positve and negative classes as the test set). Then compare the *average* scores (accuracy, precision, recall, F-score, and AUC) that you obtain from each algorithm.

      i. **Supervised Learning:** Train an $\mathscr{L}_1$-penalized SVM to classify the data. Use 5 fold cross validation to choose the penalty parameter. Use normalized data. Report the average accuracy, precision, recall, F-score, and AUC, for both training and test sets over your M runs. Plot the ROC and report the confusion matrix for training and testing in one of the runs.

      ii. **Semi-Supervised Learning/ Self-training:** select 50% of the positive class along with 50% of the negative class in the training set as *labeled data* and the rest as *unlabelled data*. You can select them randomly.

         A. Train an $\mathscr{L}_1$-penalized SVM to classify the labeled data Use normalized data. Choose the penalty parameter using 5 fold cross validation.

         B. Find the unlabeled data point that is the farthest to the decision boundary of the SVM. Let the SVM label it (ignore its true label), and add it to the labeled data, and retrain the SVM. Continue this process until all unlabeled data are used. Test the final SVM on the test data andthe average accuracy, precision, recall, F-score, and AUC, for both training and test sets over your M runs. Plot the ROC and report the confusion matrix for training and testing in one of the runs.

      iii. **Unsupervised Learning:** Run k-means algorithm on the whole training set. Ignore the labels of the data, and assume $k = 2$.

         A. Run the k-means algorithm multiple times. Make sure that you initialize the algoritm randomly. How do you make sure that the algorithm was not trapped in a local minimum?

         B. Compute the centers of the two clusters and find the closest 30 data points to each center. Read the true labels of those 30 data points and take a majority poll within them. The majority poll becomes the label predicted by k-means for the members of each cluster. Then compare the labels provided by k-means with the true labels of the training data and report the average accuracy, precision, recall, F-score, and AUC over M

runs, and ROC and the confusion matrix for one of the runs.[1]

C. Classify test data based on their proximity to the centers of the clusters. Report the average accuracy, precision, recall, F-score, and AUC over M runs, and ROC and the confusion matrix for one of the runs for the test data.

iv. **Spectral Clustering**: Repeat 1(b)iii using *spectral clustering*, which is clustering based on kernels. Research what spectral clustering is. Use RBF kernel with gamma=1 or find a gamma that the two clutsres have the same balance as the one in original data set (if the positive class has p and the negative class has n samples, the two clusters must have p and n members). Do not label data based on their proximity to cluster center, because spectral clustering may give you non-convex clusters . Instead, use $\texttt{fit} - \texttt{predict}$ method.

v. One can expect that supervised learning on the full data set works better than semi-supervised learning with half of the data set labeled.O ne can expects that unsupervised learning underperforms in such situations. Compare the results you obtained by those methods.

2. **Active Learning Using Support Vector Machines**

(a) Download the banknote authentication Data Set from: `https://archive.ics.uci.edu/ml/datasets/banknote+authentication`. Choose 472 data points randomly as the test set, and the remaining 900 points as the training set. This is a binary classification problem.

(b) Repeat each of the following two procedures 50 times. You will have 50 errors for 90 SVMs per each procedure.

i. Train a SVM with a pool of 10 randomly selected data points from the training set using linear kernel and $\mathscr{L}_1$ penalty. Select the penalty parameter using 10-fold cross validation.[2] Repeat this process by adding 10 other randomly selected data points to the pool, until you use all the 900 points. Do NOT replace the samples back into the training set at each step. Calculate the test error for each SVM. You will have 90 SVMs that were trained using 10, 20, 30, ... , 900 data points and their 90 test errors. You have implemented *passive learning*.

[1]Here we are using k-means as a classifier. The closest 30 data points to each center are labeled by *experts*, so as to use k-means for classification. Obviously, this is a naïve approach.

[2]How to choose parameter ranges for SVMs? One can use wide ranges for the parameters and a fine grid (e.g. 1000 points) for cross validation; however,this method may be computationally expensive. An alternative way is to train the SVM with very large and very small parameters on the whole training data and find very large and very small parameters for which the training accuracy is not below a threshold (e.g., 70%). Then one can select a fixed number of parameters (e.g., 20) between those points for cross validation. For the penalty parameter, usually one has to consider increments in $\log(\lambda)$. For example, if one found that the accuracy of a support vector machine will not be below 70% for $\lambda = 10^{-3}$ and $\lambda = 10^{6}$, one has to choose $\log(\lambda) \in \{-3, -2, \ldots, 4, 5, 6\}$. For the Gaussian Kernel parameter, one usually chooses linear increments,e.g. $\sigma \in \{.1, .2, \ldots, 2\}$. When both σ and λ are to be chosen using cross-validation, combinations of very small and very large λ's and σ's that keep the accuracy above a threshold (e.g.70%) can be used to determine the ranges for σ and λ. Please note that these are very rough rules of thumb, not general procedures.

ii. Train a SVM with a pool of 10 randomly selected data points from the training set[3] using linear kernel and $\mathscr{L}_1$ penalty. Select the parameters of the SVM with 10-fold cross validation. Choose the 10 closest data points in the training set to the hyperplane of the SVM[4] and add them to the pool. Do not replace the samples back into the training set. Train a new SVM using the pool. Repeat this process until all training data is used. You will have 90 SVMs that were trained using 10, 20, 30,..., 900 data points and their 90 test errors. You have implemented *active learning*.

(c) Average the 50 test errors for each of the incrementally trained 90 SVMs in 2(b)i and 2(b)ii. By doing so, you are performing a Monte Carlo simulation. Plot average test error versus number of training instances for both active and passive learners on the same figure and report your conclusions. Here, you are actually obtaining a learning curve by Monte-Carlo simulation.

[3]If all selected data points are from one class, select another set of 10 data points randomly.

[4]You may use the result from linear algebra about the distance of a point from a hyperplane.