1. **Time Series Classification**

   An interesting task in machine learning is classification of time series. In this problem, we will classify the activities of humans based on time series obtained by a Wireless Sensor Network.

   (a) Download the AReM data from: `https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+system+based+on+Multisensor+data+fusion+\%28AReM\%29` . The dataset contains 7 folders that represent seven types of activities. In each folder, there are multiple files each of which represents an instant of a human performing an activity. Each file containis 6 time series collected from activities of the same person. There are 88 instances in the dataset, each of which contains 6 time series and each time series has 480 consecutive values.

   (b) Keep datasets 1 and 2 in folders bending1 and bending 2, as well as datasets 1, 2, and 3 in other folders as test data and other datasets as train data.

   (c) Feature Extraction

   Classification of time series usually needs extracting features from them. In this problem, we focus on time-domain features.

   i. Research what types of time-domain features are usually used in time series classification and list them (examples are minimum, maximum, mean, etc).

   ii. Extract the time-domain features for all of the 6 time series in each instance. You are free to normalize/standardize features or use them directly. [1]

   iii. Estimate the standard deviation of each of the time-domain features you extracted from the data. Then, use Python's bootstrapped or any other method to build a bootsrap confidence interval for the standard deviation of each feature.

   iv. Use your judgement to select the three most important time-domain features (one option may be min, mean, and max).

   (d) Binary Classification Using Logistic Regression

   i. Assume that you want to use the training set to classify bending from other activities, i.e. you have a binary classification problem. Depict scatter plots of the features you specified in 1(c)iv extracted from time series 1, 2, and 6 of each instance, and use color to distinguish bending vs. other activities. (See p. 120 of the textbook).[2]

   ii. Break each time series in your training set into two (approximately) equal length time series and repeat the experiment in 1(d)i. Do you see any considerable difference in the results with those of 1(d)i?

   iii. Break each time series in your training set into $l \in \{1, 2, \ldots, 10\}$[3] time series of approximately equal length and use logistic regression to solve the binary

[1] You are welcome to experiment to see if they make a difference.

[2] You are welcome to repeat this experiment with other features as well as with time series 3, 4, and 5 in each instance.

[3] You are welcome to try $l \in \{11, 12, \ldots, 20\}$ as well.

classification problem, using time-domain features. Calculate the p-values for your logistic regression parameters and refit a logistic regression model using your pruned set of features.[4] Alternatively, you can use backward selection using sklearn.feature_selection or glm in R. Use 5-fold cross-validation to determine the best value of l. Explain what the right way and the wrong way are to perform cross-validation in this problem. Obviously, use the right way!

do this first

iv. Report the confusion matrix and show the ROC and AUC for your classifier. Report the parameters of your logistic regression β_i's as well as the p-values associated with them.

v. Test the classifier on the test set. Remember to break the time series in your test set into the same number of time series into which you broke your training set. Remember that the classifier has to be tested using the features extracted from the test set. Compare the accuracy on the test set with the cross-validation accuracy you obtained previously.

vi. Do your classes seem to be well-separated to cause instability in calculating logistic regression parameters?

vii. From the confusion matrices you obtained, do you see imbalanced classes? If yes, build a logistic regression model based on case-control sampling and adjust its parameters. Report the confusion matrix, ROC, and AUC of the model.

(e) Binary Classification Using $\mathscr{L}_1$-penalized logistic regression

i. Repeat 1(d)iii using $\mathscr{L}_1$-penalized logistic regression, i.e. instead of using p-values for variable selection, use $\mathscr{L}_1$ regularization. Note that in this problem, you have to cross-validate for both l, the number of time series into which you break each of your instances, and C, the weight of $\mathscr{L}_1$ penalty in your logistic regression objective function. Packages usually perform cross-validation for C automatically.[5]

ii. Compare the $\mathscr{L}_1$-penalized with variable selection using p-values. Which one performs better? Which one is easier to implement?

(f) Multi-class Classification (The Realistic Case)

i. Use the best l you found[6] from 1(e)i to build an $\mathscr{L}_1$-penalized multinomial regression model to classify all activities in your training set.[7] Report your test error. Research how confusion matrices and ROC curves are defined for multiclass classification and show them for this problem if possible.[8]

ii. Repeat 1(f)i using a Naïve Bayes' classifier. Use both Gaussian and Multinomial priors and compare the results.

iii. Which method is better for multi-class classification in this problem?

[4] R calculates the p-values for logistic regression automatically. One way of calculating them in Python is to call R within Python. There are other ways to obtain the p-values as well.

[5] Using the package Liblinear is strongly recommended.

[6] You are welcome to use cross-validation for this problem

[7] New versions of scikit learn allow using $\mathscr{L}_1$-penalty for multinomial regression.

[8] For example, the pROC package in R does the job.

2. ISLR 3.7.4
3. ISLR, 4.7.3
4. ISLR 4.7.7