University of Waterloo
ECE 457B: Fundamentals of Computational Intelligence
Winter 2025
Assignment 1 - Data Pre-processing, Experimental Setup, and Non-Parametric Classification Algorithms
Submission Due: February 7, 2025 by 11:59pm

Overview

Collaboration/Groups: You may do your work individually or with a partner. If you are working with a partner, you must sign up for an `Assignment Group` in LEARN and Crowdmark. If you are working alone, you should still join a group in LEARN with one member. You can also collaborate with other classmates on the right tools to use and setting up your programming environment, but your submitted worked must be only from members of your group.

Submission: Hand in one report per person, or group, to Crowdmark and LEARN. Your report should be submitted as two files:

- *To your group dropbox on LEARN:* Your code in the form of a jupyter notebook that has the code, and results already generated on the provided data in a readable way.
- *To Crowdmark:* Your report in the form of a pdf that is *no more than about 10 pages*. You can use the jupyter notebook to generate this report document, but you need to **remove all the code and preliminary processing** from it. Only keep what is actually essential to explain what you did, why you did it and what your results show. It might be easier to copy the text, headings, plots and tables out into a separate document and save that.

Your group on LEARN has an associated **dropbox**.

Presentation of Results (Descriptive, Concise, Clear)

- **Report document:** All tasks and questions below should be carried out or answered for both datasets. The report is a pdf document with your answers for the questions, description of your process, plots and tables for results. ***It does not contain code!***
- **Essential Plots and Tables:** Along the way, minimize the number of different plots shown to the *essentials* for the reader to understand what you did.
- **Summary Table:** At the very end you will produce a summary table of all the accuracy results for the different experiments.

- **Code:** You must submit your code, in a self-contained python jupyter notebook to LEARN. Your, you should collapse most of your code before printing to improve readability. You only need to show critical code which relates to the central task of the question or a point you are highlighting in your text.

Tools: You can use libraries available for python which we discuss in class or provide as links. You need to mention explicitly which libraries you are using, any blogs or papers you used to figure out how to carry out your calculations.

Specific objectives:

- Establish your software stack to carry out data analysis assignments for the rest of the course.
- Load datasets, run a simple classification algorithm and generate some exploratory plots and tables. Ensure all plots and tables have labels for axis, titles and short captions explaining them.
- Practice how to apply the methods discussed in class.
- Demonstrate understanding by making well reasoned design choices, and *explaining* any result you obtain in straightforward, short, text. Remember, there isn't always one particular right answer to how to do something, but you must justify what you did and demonstrate how it worked.

Dataset

The first data set is the Wine Quality Data Set:

- Wine Quality Dataset:
  `https://archive.ics.uci.edu/ml/datasets/wine+quality`
- The dataset original consists of one datafile for red wine and one datafile for white wine. The provided jupyter notebook loads the data and adds an additional column for wine `color` where `color` $= 0$ indicates white wine and `color` $= 1$ indicates red wine.
- **Classification task:** predict 'color' from the other features, including 'quality'.

The second dataset is the Abalone dataset about the abalone fish and its physical characteristics.

- An `abalone.csv` file will be available in the asg1 directory on learn.
- Original Dataset:
  `https://archive.ics.uci.edu/dataset/1/abalone`
- **Classification task:** predict 'Rings' from the other features, this feature is essentially the age of the fish. Note that one of the features is categorical.

1 Assessment of Data and Applying Normalization

For each dataset, first you need to look at the dataset as a whole and analyse if we need to carry out any preprocessing. Load the dataset and explore the features and their ranges and distribution. Show just a few highlights of the approaches you use. Normalize the data using `z-score` normalization (ie. standarization) as a preprocessing step. Be sure to answer the following questions along the way:

1. Is there any missing data?
2. Compute the moments or summarization statistics on the data features (mean, median, variance, skew, kurtosis). Do these highlight anything interesting about the different features?
3. Use a pairsplot (the seaborn library has a nice one, for example) to look at the whole of the dataset. Choose a subset, just some features, and show it in your report to highlight some features that seem important.
4. Is this a balanced dataset? If not, what kind of correction could we apply?

2 Classification with KNN

Classify the data using a KNN classifier using the `KNeighborsClassifier` class in the python `SciKit Learn` library. You will explore some parameters of the KNN classifier, plot the different validation accuracies against the values of the parameter, select the best parameter to fit the model and report the resulting accuracy.

Carry out the following activities and reporting:

1. Start by training the model with the classifier's default parameters using the training set and test the model with the test set.
2. Note that different values of k will lead to different results. Note that Abalone is a hard dataset to get high accuracy on, with kNN accuracy the accuracy could be below 50%.
3. To find the best value for k, you need to compute accuracy for a range of values of k so you can "tune" the classifier.
4. First divide the data into Training and Testing sets in the ratio 80%/20%.
5. You then need to test a range of values for the KNN parameter k using **5-fold cross validation** on the Training set. So each round of cross-validation will use 4 of the folds for training, and 1 of the folds for computing accuracy to observe how well that k does.

6. All of these 5-folds can be plotted, analysed, and averaged to determine the best value of k to use for KNN. You can decide which information to show, but at least produce one plot of the mean validation accuracy vs. k across all folds.

7. Once a promising value of k is chosen, you can retrain KNN using that k on all the training data and calculate accuracy on the held out Test set and report this result.

8. **Improving on KNN:** We can also try to improve on our classification results using the method of *weighted* KNN. The `KNeighborsClassifier` class has an option for *weighted* KNN where points that are nearby to the query point are more important for the classification than others.

    - using your chosen value of k above, compute the values on your Test set using some other weighting method.

3 Decision Trees Classifier

You will now do classification on your datasets using Decision Trees. Decision Trees have a number of parameters that can effect peformance. You can use the `GridSearchCV` function for this question.

1. Use 5-fold cross validation and a range of parameter values to evaluate the best settings for classification on each dataset.

    - the maximum depth of trees

2. Produce a plot showing the mean accuracy vs. relative to tree depth.

3. **Interpretability:** Use the decision tree library functions, to examine the final resulting splitting rules used for the trees. Do they indicate any interesting patterns that explain the data? Can you find support for this from any analysis you've done or see on this dataset previously? For this part, *use the original raw feature space only*, not the PCA/LDA space. (Why not?)

    - Relevant decision tree visualizers, whichever one you use, make sure it is readable in useful way, don't show information that isn't helpful:
    - `tree.plot_tree()`: the built-in tree plot function for
    - `sklearn.tree.DecisionTree` `tree.export_graphviz` : another simple visualizer
    - `sklearn.tree.export_text` : text view of the tree data

4 Random Forest Classifier

You will now do classification on your datasets using Random Forests. Random Forests have a number of parameters that can effect performance. You can use the `GridSearchCV` function for this question.

1. Use 5-fold cross validation and a range of parameter values to evaluate the best settings for classification on each dataset.
    - the maximum depth of trees, you can try values as low as 2 or 3 and as high as needed, decision trees have an upper limit on how deep they can go determine by the size of the dataset.
    - the number of trees, try values at regular intervals, you can go as low as 3 and as high as a few hundred trees.
2. Produce a plot showing the mean accuracy vs. the above parameter settings. This can be individually or using a **heat plot** showing a grid of mean accuracies for different combinations of the two parameters.

NOTE: *do not produce a tree plot or export for each tree in the forest!*
Include summary accuracy scores on all datasets in the table in the last question.

5 Final Results

In this question, summarize your findings concisely in words and tables.

- Comment on which pipeline resulted in the best classification accuracy overall, or for each dataset.
- Feel free to make additional observations about the results beyond these.
- Produce results tables summarizing all the final results in the following general form:

	setting(*)	wine	abalone
kNN	best setting k=?, etc.	acc	acc
Decision Tree	best settings	acc	acc
Random Forest	best settings	acc	acc

Notes

You might find the following links are useful to solve this assignment:

- `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`
- `https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation`

- https://scikit-learn.org/1.5/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier