COMP2200/COMP6200 Practical Exercise: Understanding Training, Validation, and Test Sets

1 Introduction

In this exercise, you will explore the concepts of training, validation, and test datasets by manually deriving classification rules from real-world data. This will be a hands-on group activity designed to help you understand the role of each dataset in machine learning.

2 Learning Objectives

By the end of this exercise, you should:

- Understand the distinction between training, validation, and test datasets.
- Be able to derive simple classification rules from observed data.
- Recognize the importance of validation data in refining rules.
- Understand the role of test data in final model evaluation.
- Learn basic Git version control skills and commit work to a GitHub Classroom repository.
- Gain familiarity with using scikit-learn to load and process data.

3 Setup

You will be working in groups of three, with each person playing a specific role:

- **Training Person**: Observes the training dataset and comes up with classification rules.
- **Validation Person**: Applies the given rules to the validation dataset.
- **Test Person**: Evaluates the final chosen rule on the test dataset.

If you can't form a group of three, you can share a validation person or test person between teams.

We will use the **Wisconsin Breast Cancer Dataset** from scikit-learn, a binary classification dataset with real-world medical data. These are measurements of different kinds of cancers (cheap and easy to do), then they did an expensive biopsy to determine whether it was malignant or benign.

Your job is to find some rules based on the measurements which could do a good job of predicting whether they were malignant or benign *without needing to do the biopsy.*

4 Loading and Splitting the Data in Python

Download `Practical1.ipynb` and run it. It should give you output like this:

```
Training set size: (549, 31)
Validation set size: (10, 31)
Test set size: (10, 31)
```

If you are the **training** person, run this code in another cell:

```
train_df
```

If you are the **validation** person, run this code in another cell:

```
val_df
```

If you are the **test** person, run this code in another cell:

```
test_df
```

4.1 Navigating Dataframes

Note that you can't see all the data. Here are some commands that you could put in a cell in the Jupyter notebook to see different parts of the data frame:

`train_df.columns` Get a list of all the column names

`train_df['concave points error']` Show one particular column

`train_df[['mean error', 'malignant']]` Double square brackets to show several columns at once

`train_df.sample(10)` Take a random sample of 10 rows, which will be different every time you run it

`train_df.head(15)` Show the first 15 rows

`train_df.tail(15)` Show the last 15 rows

Replace `train_df` with `val_df` or `test_df` depending on your role. There are more methods listed in the Pandas Cheat Sheet on iLearn if you are interested.

5 Procedure

Scroll to the right and you will see a column called "malignant". The labels indicate whether the tumor is benign (0) or malignant (1).

Each group will go through multiple rounds, progressively refining their classification rules:

5.1 Round 1: Silly Rules

The training person should come up with the most absurd rule they can think of to classify the data (e.g., "If the sample index is odd, classify as malignant"). Ask the validation person to apply this rule to the validation data and record how many this rule got right.

5.2 Round 2: Sensible Rules

The training person should look at the data more carefully and try to come up with a simple, reasonable rule based on the features you observe (e.g., "If mean radius is greater than 5, classify as malignant").

Again, ask the validation person to apply this rule to the validation set and record the results.

5.3 Round 3: More Sophisticated Rules

Attempt to refine your rule by incorporating multiple features (e.g., "If mean radius is greater than 2 and texture score is below 3, classify as benign").

Again, test this rule on the validation set and record the results.

Repeat this as many times as you want, until the validation isn't seeing much improvement.

5.4 Final Round: Choose the Best Rule

After evaluating your different rules, select the one you believe performs best, based on the results from validation.

Now ask the test person to apply that rule. They will evaluate it on the test dataset. This is the real measure of how good your rule is!

6 Deliverables

Each individual must create a `results.txt` file containing:

- All rules generated across the rounds.
- The corresponding validation results.
- The final rule chosen for the test set.

- The test set evaluation results.
- A short reflection on what was learned about training, validation, and test datasets.

Commit this file to your assigned GitHub Classroom repository. You will not be marked on the accuracy of your rules but rather on your understanding of when to use each dataset and your ability to use Git.

(But if your team did have the best results, feel free to boast about it.)