CITS5508: Assignment 2

Semester 1, 2025

Assessed, worth **20%**. Due **11:59 pm Friday 16th May 2025**.

1 Outline

This assignment is divided into two parts: a *Voting Classifier* on the *Diagnostic Wisconsin Breast Cancer Database* and a *Random Forest* on the *Labeled Faces in the Wild* dataset.

Detailed descriptions of each task are given below.

While this assignment is centred around programming, you should treat your submission as a report. A specific set of tasks is given for both **Part 1**, and **Part 2** however, your assignment is to perform an investigation. Do not expect to receive full marks for one-word answers when asked to comment on your results. Explain your thoughts in detail.

2 Submission

- You should submit your assignment via LMS. The submission should be a single Jupyter Notebook file (with an `.ipynb` extension).
- Your notebook should include all code, outputs, descriptions and comments. Treat it as a report – introduce the topic, describe your methodology, present your results and conclusions.
  - Make use of Markdown cells inserted at appropriate places to explain your code. In addition to Markdown cells, some short comments can be put alongside the Python code
  - Dividing the assignment into suitable sections and subsections (with section and subsection numbers and meaningful headings) would make your portfolio easier to follow.
  - Please see `sample.ipynb` under the Week 2 content on LMS for an example of how to structure your notebook.
- Your notebook should already include all output, but must also run without errors when using the CITS5508 Anaconda environment. Note that Google Colaboratory will produce identical outputs to our environment.

3 Part 1 – Voting Classifier

Your first task is to build a **Voting Classifier** for the *Diagnostic Wisconsin Breast Cancer Database* from your first assignment. The dataset can be downloaded from the UC Irvine Machine Learning Repository via:

https://doi.org/10.24432/C5DW2B

Please download the dataset and place the extracted files in the same directory as your notebook. Do not use the "import in Python" option from the website.

3.1 Tasks

1. Load the dataset and make sure all columns are named correctly. Drop any unnecessary columns.
2. Split the dataset into a training and testing set. Use 80% of the data for training, and 20% for testing. Make sure to stratify your split.
3. Fit three base models to your training data using scikit-learn:
   - Logistic Regression
   - Random Forest
   - Support Vector Machine
4. Create a Voting Classifier using the three base models with scikit-learn. Use soft-voting.
5. Evaluate the performance of all four models (base models and voting classifier) on the test set.
   (a) Create a table of precision, recall, and F1-scores for each model.
   (b) Display a confusion matrix for each model.
6. Comment on your results in detail. Investigate the following ideas:
   (a) How do the models compare to each other? Which model performs best? Is this what you expected?
   (b) What are the advantages and disadvantages of using a voting classifier instead of just using one of the base models?

4 Part 2 – Labeled Faces in the Wild

Your second task is to build a **Random Forest** classifier for the *Labeled Faces in the Wild* dataset. This dataset is a collection of pictures of famous people collected from the internet.

You can load the dataset directly from scikit-learn using the following code:

```python
from sklearn.datasets import fetch_lfw_people

lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.5)

n_samples, h, w = lfw_people.images.shape

X = lfw_people.data
n_features = X.shape[1]

y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
```

You can learn more about the dataset by running `lfw_people['DESCR']` after running the code above.

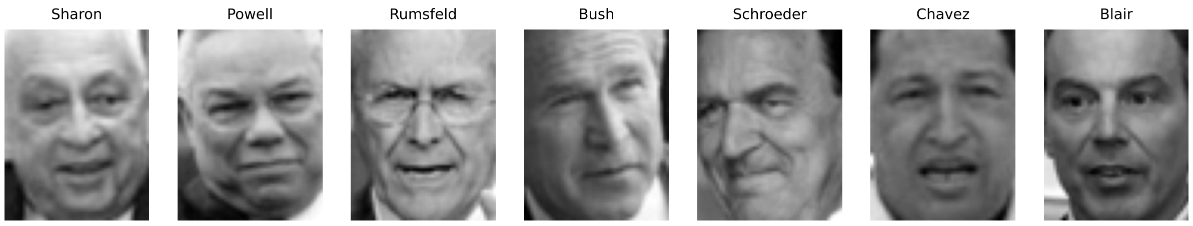

Figure 1: Example images from the dataset.

4.1 Tasks

1. Load the Labeled Faces in the Wild (LFW) people dataset using the code above. Briefly describe the dataset:
   (a) How many examples are there in the dataset? What do the examples represent?
   (b) Describe the features of the dataset. What do they represent? How many features are there?
   (c) Describe the target of the dataset. How many classes are there?
2. Split the dataset into a training and testing set. Use 80% of the data for training, and 20% for testing. Make sure to stratify your split.
3. Sample a random example for each class from the training set and display the image. Ensure your images are labelled with the name of the class.
4. Create a Random Forest classifier using scikit-learn. Use 1000 trees in the forest.
   (a) Create a table of precision, recall, and F1-scores for the classifier. Display the confusion matrix.
   (b) Plot the feature importance of the Random Forest classifier. Briefly comment on the plot.
5. Perform Principal Component Analysis (PCA) on the training data using scikit-learn. Keep 150 principal components.
   (a) How many dimensions does the data have before and after PCA?
   (b) What is the explained variance ratio of the 150 principal components?
   (c) Display the first 10 *eigenfaces*. *Eigenfaces* are the principal axes in the feature space of the PCA transformed data and are often used face recognition tasks.
   (d) Explain the meaning of the eigenfaces. How do they relate to the original images? What do they represent?
6. Create a Random Forest classifier using the PCA transformed data. Once again, use 1000 trees in the forest.
   (a) Create a table of precision, recall, and F1-scores for the classifier.
   (b) Display the confusion matrix for the classifier.
7. Compare the Random Forest classifier on the PCA transformed data to the Random Forest classifier on the original data. Comment on the following:
   (a) How do the models compare to each other? Which model performs best? Is this what you expected?
   (b) Comment on the time taken to train each model. How does PCA affect the training time? Why do you think this is the case?
   (c) Compare the feature importances of the two models. Is this what you expected?