# CITS 2401
## Computer Analysis and Visualisation

THE UNIVERSITY OF WESTERN AUSTRALIA

# Project 1

### Cars data analysis system

**Worth:** 10% of the unit
**Submission Type**: Answer the questions on the quiz server (Moodle). Details in Submission section of this document.
**Deadline**: **18th April 2025 11:59 pm**
**Late submissions:** late submissions attract a 5% raw penalty per day up to 7 days (i.e., 25th April 2025 11:59 pm). After that, the mark will be 0 (zero). Also, any plagiarised work will be marked zero.

## 1. Outline

XYZ automobile company has a collection of cars that vary in make and year. The sales and marketing department has to analyse the data of car collections to predict their sales for the individual brand and their model year. The company usually stores information about each car in a database. They can export the data into a CSV (comma-separated value) file for further analysis. They use Python as the programming language for this analysis task.

You are working as an analyst in their analysis team. You are responsible for implementing a Car Sale Analysis System. In this implementation, you are required to develop a Python program to read the datasets and complete some data analysis tasks.

To complete this project, you must refer to the lectures up to week 5 Files (recommended week 6 Sets and Dictionary for easier data management), along with their relevant labs. The dataset for this project is provided on the quiz server (Moodle) `cars_data.csv` which contains information regarding cars data. The first row of the file contains the header: `Brand, Price, Body, Mileage, EngineV, Engine Type, Registration, Year, Model, Traffic Incident,` and `Fuel Economy`.

## Submission Policy:

You are expected to have read and understood the University's **guidelines on academic conduct**. In accordance with this policy, this is an individual project, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learned little and will therefore, likely, fail the final exam. It is also important to keep in mind that ChatGPT and other similar tools are limited in their ability to generate outputs, and it is easy to detect if you use their outputs without understanding the underlying principles. The main goal of this project is to demonstrate your understanding of programming principles and how they can be applied in practical contexts.

You must submit your project before the deadline listed above. Following UWA policy, a late penalty of 5% will be deducted for each day i.e., 24 hours after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

## 2. Tasks

Once you commence the quiz on the quiz server, you will have access to sample test cases for these tasks.

**Section 1. Data Preprocessing**
You will initially preprocess the data before you start analysis.

# CITS 2401
## Computer Analysis and Visualisation

THE UNIVERSITY OF WESTERN AUSTRALIA

**Task 1: Import data [2 marks]**

The primary objective of Task 1 is to collect data from the provided dataset. The dataset file presented in CSV format, along with instructions and a quick preview below, will be provided to you (on the Quiz Server).

Now, your task is to write a function `read_data(filename)` that retrieves the data from the given file `filename` (the file is in CSV format) by returning every entry (row) INCLUDING the headings (the headings always exist in given files, this will be used later for finding indices of columns) as a 2-dimension list. Each entry in this 2-dimension list should be a list containing data in string format separated by a comma (one list per row, and any new line characters at the end of each row should be stripped off). It's important to note that no imports should be used for this task. You may assume there are no errors in the given and tested files.

Sample test case:

```
data = read_data('cars_data.csv')

print(data[0])
```

Sample output:

```
['Brand', 'Price', 'Body', 'Mileage', 'EngineV', 'Engine Type',
'Registration', 'Year', 'Model', 'Traffic Incident', 'Fuel Economy']
```

**Task 2: Process data to retrieve only the relevant columns [5 marks]**

In this task, you are to write a function `process_data(data)` that takes the `data` output from `read_data(filename)` function, collects the relevant information for each car in a **tuple**, row by row, while only considering the columns you need for completing the future tasks; you should focus on the essential ones while disregarding the others. This function saves all the tuples (one per row) in a list called `preprocessed_data` and returns it. Remember, this list does NOT contain the heading row and there are more columns than necessary for this project in the CSV file. The data from each row are collected in a tuple, and the list contains the tuples. The columns that you need to consider are: `"Brand", "Price", "Mileage", "EngineV", "Year", "Engine Type"`. The data from these columns should appear in the above order in the tuple; see also the sample test case below.

Please note that, the order of columns in the provided CSV file may differ from the file we use to test your code. So you should not assume that the column will be in the same order as the sample csv file. You should use the header row (the first row in the CSV file) to identify the correct column based on its name, rather than its position in the file. Ensure that entries are stored in appropriate data type i.e. numbers are stored in integers or floats depending on the data type and so on.

Sample test case:

```
preprocessed_data = process_data(data)

print(preprocessed_data[0])
```

Sample output:

```
('BMW', '34900.0', '35', '3.0', '2011', 'Petrol')
```

**Task 3: Filter the cars data [6 marks]**

Write a function `filter_data(preprocessed_data)` that takes the pre-processed data from `process_data(data)` function in task 2 as input. This function will return two lists: `brand_names` – a list containing all the names of unique car brands, and `grouped_data` – a nested list (i.e., a list of lists containing relevant car information for each brand. The `grouped_data` list should have as many lists as there are unique car bands – For example, if there are seven cars brand, there will be seven sub-lists in `grouped_data`. Each of these sub-lists should also contain multiple lists, one for each car of that respective brand. These lists should contain all the information of that respective car in `preprocessed_data`. For example, if there are 300 BMW cars in `preprocessed_data`, the list corresponding to this brand in `grouped_data` should contain 300 lists. Each of these lists should contain the `"Brand", "Price", "Mileage", "EngineV", "Year", "Engine Type"` data of the car from `preprocessed_data`.

Sample test case:

```
brand_list, grouped_data = filter_data(preprocessed_data)

print(brand_list)

print(grouped_data[0][0])
```

Sample output:

```
['Audi', 'BMW', 'Mercedes-Benz', 'Mitsubishi', 'Renault', 'Toyota',
'Volkswagen']

['Audi', '3650.0', '240', '2.5', '2000', 'Diesel']
```

**Task 4: Calculate Statistics [15 marks]**

In this task you will complete some statistical computation using the filtered data from Task 3. Write functions for each of the sub-tasks. Each sub-task make use of `grouped_data`.

*NOTE: For the following tasks, you might find it useful to define two extra functions: one function to calculate the cosine similarity between two vectors (e.g.* `cosine_similarity(a,b)`*) and one function to calculate the standard deviation of a vector (e.g.* `st_dev(a)`*). The formulas for cosine similarity and standard deviation can be found at the end of this document.*

i. **[3 marks]** Write a function `average_stdev_price(grouped_data)` that takes the relevant data from task 3 to calculate the (1) average and (2) standard deviation of car prices for each individual brand store them into a list. This function should return two lists: `list_average_prices` and `list_stdev_prices`, both having one element for each brand (for example, if there are seven brands, both lists should have seven elements). **The values in the lists should be rounded to two decimal places.**

   Sample test case:

   ```
   list_avg_prices,list_stdev_prices = average_stdev_price(grouped_data)
   print(list_average_prices)
   print(list_stdev_prices)
   ```

Sample Output:

```
[17768.83, 23520.65, 28411.84, 11529.25, 8328.73, 20839.65, 13068.16]
```

```
[19284.15, 22344.92, 33891.76, 6132.84, 3655.8, 19592.95, 10020.83]
```

ii. **[3 marks]** In this task, write a function `price_range(grouped_data)` to select only the cars newer than the year 2000 (inclusive), and calculate the price range of these cars, for each brand (i.e., the difference between the maximum and minimum price). This function should return a list of price ranges, one value for each car brand, **rounded to two decimal places**.

Sample test case:

```
price_ranges = price_range(grouped_data)
print(price_ranges)
```

Sample output:

```
[95900.0, 135250.0, 198009.0, 24830.0, 19196.24, 162001.0, 53155.0]
```

**iii.** **[3 marks]** In this task, write a function `cosine_similarity_by_brand(grouped_data)` to calculate the cosine similarity between car price and engine size, **only for the Petrol cars** of each brand. This function should return a list of cosine similarity values, one value for each car brand**, rounded to two decimal places.**

Sample test case:

```
cosine_similarities = cosine_similarity_by_brand(grouped_data)
print(cosine_similarities)
```

Sample output:

```
[0.81, 0.6, 0.23, 0.92, 0.88, 0.94, 0.85]
```

iv. **[3 marks]** Write a function `average_price_engine_types(grouped_data)` that calculates the average price of **ALL** Petrol, Diesel and Gas cars (across all brands) with mileage below 100(exclusive). **Note that if the engine type of a car is assigned as "Other", it should be considered together with the cars that have gas engines**. Unlike tasks *i-iii* above, this function should not return a separate result for each brand; rather, it should consider all brands together and return a list of 3 values, **rounded to two decimal places**.

Sample test case:

```
average_price_engines = average_price_engine_types(grouped_data)
print(average_price_engines)
```

Sample output:

```
[19843.11, 20216.8, 13392.65]
```

v. **[3 marks]** In this task, create a function called `brand_engine_vol(grouped_data, brand_list)` to calculate the average engine volume, for each brand. This function should return a list of tuples, one tuple for each brand. Each tuple should contain two values: the first value should be the brand name in string format (taken from `brand_list`) and the second value should be the average engine volume for the cars belonging to the respective brand, **rounded to two decimal places**.

Sample test case:
```
avg_engine_volumes = brand_engine_vol(grouped_data, brand_list)

print(avg_engine_volumes)
```

Sample output:
```
[('Audi', 2.83), ('BMW', 3.67), ('Mercedes-Benz', 3.45),
('Mitsubishi', 2.34), ('Renault', 2.8), ('Toyota', 2.69),
('Volkswagen', 1.94)]
```

**This concludes section 1, at the end of which you will get a large amount of car statistics. Now you can move onto car affordability prediction task!**

**Section 2. Car affordability prediction using statistical analysis**

In this section, you will use statistical analysis as a simple method to predict the affordability of a car. The approach involves predicting affordability based on the engine volume and mileage. For example, if the engine volume and car mileage are below and above a set of thresholds, then it is likely that the car will be quite cheap. You will now proceed to implement this method.

***NOTE: This part is the most complex one in this project, so please make sure to read it carefully.***

**Task 5: Predict Rain [6 marks]**

You will write a function `affordability_prediction(grouped_data, threshold1=2.6, threshold2=100)` that returns two lists: `predicted and actual`; containing "Yes" (for affordable) and "No" (for not affordable). The `actual` list is constructed by looking at all the car entries in `grouped_data` (across all brands; you do not need to make any separation based on brands at this point) and assigning "No" if the price is above 15 000 (exclusive), and "Yes" otherwise. The predicted list is constructed by considering the engine volume and mileage of each car. If the engine **volume** is **below the value of threshold1** (by default, 2.6; not inclusive) and the **mileage** is **above the value for threshold2** (by default, 100; not inclusive), the prediction is "Yes". Otherwise, the prediction will be "No".

The input parameters:

- `grouped_data`: The output from Task 3
- `threshold1(for engine volume) and threashhold2(for mileage)`: These are the thresholds for the engine volume and mileage used to predict a car's affordability. The default values are set to 2.6 liters for engine volume and 100 (thousands km) for mileage. If the engine volume is less than `threshold1` and the mileage is larger than `threshold2`, then the prediction for affordability is "Yes".

Sample test case:

```
actual, predicted = affordability_prediction(grouped_data)

print(len(actual), len(predicted))

print(actual[0], predicted[0])
```

Sample output:

```
1000 1000

Yes Yes
```

**Task 6: Evaluating the Detection System [6 marks]**

Our final task is to evaluate how well our affordability detection system works. This feat is done by calculating some performance metrics. For our evaluation, we will calculate four metrics – accuracy, precision, recall and F1 score. The calculation formulae for these metrics are all available on this page (you can do your own research in addition to reading this):

https://en.wikipedia.org/wiki/F-score

- The accuracy measures how well the detection system has performed.
- The precision measures the ratio of affordability predicted YES out of the all the labels.
- The recall measures the ratio of affordability predicted YES out of all the YES labels.
- F1 score measure is a balanced representation of both precision and recall.

Formulas for the metrices are given below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where **True Negatives (TN)** refers to the number of affordability predictions that are correctly made to be negative (actual No and predicted No), **True Positives (TP)** refers to the number of affordability predictions that are correctly made to be positive (actual Yes and predicted Yes), **False Negatives (FN)** refers to the number of affordability predictions that are incorrectly made to be negative (actual Yes and predicted No), and **False Positives (FP)** refers to the number of predictions that are incorrectly made to be positive (actual No and predicted Yes).

Write a function `result_analysis(actual, predicted)` that takes the two lists from the previous task and **prints (NOT returns)** the statistical analysis results for your cars data analysis system. The outputs should be in the following order and format:

- **Accuracy**: output text “Accuracy” followed by a colon “:” and the result formatted to four (4) decimal places.

- **Precision, Recall and F1 Score** are printed in this order with the same formatting, formatted to four (4) decimal places.

With the default threshold values for the given dataset `cars_data.csv` you should get the following result.

Sample test case:

```
result_analysis(actual, predicted)
```

Sample output:

```
Accuracy:  0.753
Precision:  0.8914
Recall:  0.6916
F1 Score:  0.7789
```

## 3. Submission

You will answer the questions related to the tasks above on the quiz server (Moodle) by the due date – **18th April 2025 at 11:59 pm** (drop dead due date **25th April 2025** with 5% raw penalty per day).

You are also required to submit your Python code in the quiz answer box containing all functions to solve the above tasks on the quiz server, as well as attach the python file containing all the code you wrote for this project in "Final Submission" section. You must name the file as `P1_[student id].py`. For example, if your student ID is 12345678, then your file name is `P1_12345678.py`.

Failure to follow these instructions will result in a **penalty of 50%.**

**Code Format:**

Make sure you have the module docstring for your project code, indicating your name, your student ID number along with the description of the project. You

**Important Note:**

i) We may use different CSV file to test your code on quiz server; The order of the column is different, zero values, larger dataset etc. Please consider all these boundary cases in your solution. You will find different sample test cases in quiz server from this document.

ii) You must round the floating-point numbers to the number of decimal places when you return or display the result required by each task (two decimal places in Task 4 and four decimal point in Task 6)

iii) You MUST not import any library/module in this project. However, you are going to import library/module in your Project 2.

iv) There will be no partial mark for any questions. You solution of each question must pass all test cases.

*v) This project is worth a total of 50 marks. Please note that your submission will also be manually checked including coding format and style before we release the final marks.*

**Penalty:**

**Be careful about multiple submissions of a question, as each resubmission will attract a 10% penalty i.e. 1st resubmission 10%, 2nd 20% and so on. Test your code in your machine before you submit.**

## 4. Rubrics

| | Criteria | Highly Satisfactory (D, HD) | Satisfactory (P, CR) | Unsatisfactory (N) |
|---|---|---|---|---|
| Python functions (40) | • Understand the use of Python functions.<br>• Demonstrate the ability to write and execute Python functions. | **Demonstrated the ability to use Python functions fluently:**<br>• Correct use of Python functions as appropriate.<br>• Code is error free. | **Demonstrated the ability to use Python functions:**<br>• Mostly correct uses of Python functions.<br>• Code is error free. | **Failed to demonstrate the ability to use Python functions:**<br>• Only a few correct use of Python functions.<br>• Code has errors. |
| Coding Style (10) | • Code is written in accordance with the style guideline.<br>• Code is written legibly and is of high standard. | **Demonstrated the ability to present Python code comprehensively:**<br>• The coding style conforms to the style guideline with attention to details. | **Demonstrated the ability to present Python code:**<br>• The coding style conforms to the style guideline. | **Failed to demonstrate the ability to present Python code:**<br>• The coding style does not conform to the style guideline. |

# CITS 2401
## Computer Analysis and Visualisation

THE UNIVERSITY OF WESTERN AUSTRALIA

**Necessary formulas:**

1. **Cosine similarity**

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Where $A_i$ is car price and $B_i$ is the engine size.

2. **Standard deviation**

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}},$$

Where $x_1, x_2, x_3 \ldots\ldots x_n$ are observed value in sample data. $\bar{x}$ is the mean value of observations and $N$ is the number of sample observations.