Assessment 1: Set Exercises

Set Exercise 1: worth 40% of element mark

During Lecture 1 you were asked to suggest applications of artificial intelligence. Your task is to **select one of these applications** and **write a blog post describing it**. The topics will be available on the Moodle shortly after the week 1 sessions.

Your blog post must:

- Introduce the topic – you should explain the purpose behind the AI. What will it mean we can now do that we couldn't before?
- Discuss whether the AI has been successful – what has the impact of its introduction been.
- Discuss the future of the AI – will it continue to be used? Are there any improvements required to make it better?

The word limit for your blog post is **700 words**.

For information on how to write a good blog post, have a look at the resources posted on the Moodle.

You must **submit your blog post in PDF form**.

Assessment Criteria

A rubric will be used to **assess** and **provide feedback** on your submission. This is provided below:

	Fail (< 40%)	>= 40	>= 50	>= 60	>= 70	>= 80
AI application is introduced (25%)	Requirements not met	Vague introduction with little detail	Some introduction but some omissions	Application idea introduce but more detail on its purpose needed	Generally good introduction – AI and its purpose are discussed. Some minor omissions.	The AI and its purpose are fully introduced. No omissions.
The AI's success is discussed (30%)	Requirements not met	Vague description of AI's impact with little detail. No discussion.	Some description of AI's impact in some detail. No discussion.	Good description of the AI's impact. Some analytical discussion.	The AI's impact is well described. Good analytical discussion but lacking detail in a few places.	Excellent discussion of the AI's impact and strong analytical discussion.
The AI's future is discussed (30%)	Requirements not met	Vague description of the AI's future with little detail. No discussion.	Some description of the AI's future in some detail. No discussion.	Good description of the AI's future. Some analytical discussion.	The AI's future is well described. Good analytical discussion but lacking detail in a few places.	Excellent discussion of the AI's future and strong analytical discussion.
Style and language (15%)	Requirements not met	Poor spelling and grammar. Inappropriate structure.	Poor spelling and grammar. Some improvements to structure required.	Some minor spelling or grammatical errors. Some improvements to structure required.	Some minor spelling or grammatical errors. A suitable structure is followed.	Excellent. Very few spelling or grammatical errors. A suitable structure is followed.

Assessment 1: Set Exercises

Set Exercise 2: worth 60% of element mark

Your task here is to produce a technical poster about the design of an AI-based assistant to be used by within a hospital. You must outline the AI components that will go into the assistant, explain how a user will interact with it.

Your poster post must:

- Describe the AI components – how will you use machine learning, evolutionary computation, knowledge representation and NLP? (You might not use all of these, it's up to you to decide). How and why will each be used?
- Indicate how the components are assembled into the final system.
- Outline the UI – how does the user input requests, and how is the information presented?
- Include any ethical considerations.

The poster must be submitted as a PDF file to the Moodle and you may use a single page only. A template has been provided for you on the Moodle.

For information on how to write a good scientific poster, have a look at the resources posted on the Moodle.

Assessment Criteria

A rubric will be used to **assess** and **provide feedback** on your submission. This is provided below:

	Fail (< 40%)	**>= 40**	**>= 50**	**>= 60**	**>= 70**	**>= 80**
Selection and justification of AI components (50%)	Not clear which AI will be used or how, no rationale behind their selection.	Some of the AIs' use is described but there is little rationale.	The AI to be used is mostly described. Some aspects are justified but more detail is needed.	The AI to be used is described. Most areas are justified but justification could be strengthened in places.	The AI to be used is described and is mostly justified.	All of the AI to be used is described and the rationale behind the choices is clear.
UI design (30%)	Little consideration to the UI is given, and there is no rationale behind the design.	There is some consideration to the UI and some reasonable choices have been made. No justification.	Reasonable UI choices have been made and justification has been attempted.	The UI is mostly well designed. The design is generally justified but more depth is needed.	The UI is well designed. The rationale behind the design is mostly clear but could be enhanced in one or two areas.	The UI is well designed and the rationale behind the design is clear.
Style and language (20%)	Poor spelling and grammar. Inappropriate structure. Poster content is too brief.	Some spelling and grammar errors. Inappropriate structure. Poster content is too brief.	Some spelling and grammar errors. Minor improvements to structure required. Poster content is too brief.	Some minor spelling or grammatical errors. Minor improvements to structure required.	Some minor spelling or grammatical errors. A suitable structure is followed.	Excellent. Very few spelling or grammatical errors. A suitable structure is followed.

Assessment 2: Machine Learning and Optimisation

This assignment contributes **70%** of the overall module mark for COMP2002 and is an **individual assignment**. You must submit the deliverables to the Moodle by the specified submission dates.

The coursework has two parts – one is a **machine learning exercise** and the second is about **evolutionary computation**. You must **complete and submit both parts**. Each part is worth 50% of the coursework mark. A Jupyter notebook has been placed on the Moodle for you to use. You should download it and use it to implement the code you need to complete the tasks below.

PART 1 – MACHINE LEARNING

You have been provided with datasets relating to the energy efficiency of buildings. Your task is to train regression models that predict two indicators of energy efficiency based on 8 numerical inputs.

You must complete the following tasks:

Task 1.1 – Data preparation (10% of total mark)

The first phase of the work requires you to load the data you have been provided with into your Python program. Before the data can be used to train and test your models you must first prepare it – this means that the inputs must be normalized. There is no missing data in the dataset.

Task 1.2 – Regression (20% of total mark)

Having prepared the data you must now build a regression tool that can predict new points. Use the following regression implementations within the *scikit-learn* package to construct predictors for the dataset:

- Random Forest (*sklearn.ensemble.RandomForestRegressor*)
- Neural Network (*sklearn.neural_network.MLPRegressor*)
- Support Vector Machine (*sklearn.svm.SVR*)

You must demonstrate that each regressor is capable of providing a prediction for a given input.

Task 1.3 – Assessment of regression (20% of total mark)

The regression models you have used in the previous task must be assessed. To do this you are required to assess the *mean square error* rate for each model. You may use the MSE implementation available in *scikit-learn* to do this. It is not sufficient to report a single MSE rate. You must use cross validation to report training and testing results and report these values using a boxplot.

PART 2 – OPTIMISATION

The second part of this assignment requires you to implement an optimiser to solve a timetabling problem for a university. You have been provided with a file that describes modules, and lists modules against which they cannot be scheduled. A module consists of one lecture per week and one or more lab sessions.

Your task is to optimise a timetable so that session is scheduled once per week in such a way that timetable constraints are minimised. There are 20 sessions per week. You have one lecture theatre and two labs available.

Relevant constraints for this assessment are:

- A session cannot be scheduled for a time when any of its students or staff are in another session (**concurrence constraints**). The sessions for a module that clash are shown in the data file.
- A lab session cannot occur in the week before its corresponding lecture has taken place (**precedence constraints**).

You have been provided with a file describing 17 modules, the number of lab sessions, and the modules that a module cannot conflict with. You must design and implement a fitness function by taking the **number of the concurrence constraints** and multiplying them with the **number of precedence constraints**. This fitness function should be minimised – the ideal timetable is one with no constraint violations at all, in which case the function will return 0.

Task 2.1 – Generation of random solutions (10% of total mark)

Your first task is to implement the fitness function. Your code should read the file provided and given a timetable it should return the quality of the schedule in terms of the constraint violations, as described above. You should call your function and print out the fitness of a random route.

Task 2.2 – Algorithm implementation (25% of total mark)

You should implement a hillclimber (as described in the lectures) to optimise the problem implemented in Task 2.1. Your algorithm must have the following features:

- It should be possible to use one of two mutation operators – one (**session replace**) mutates a solution by placing a session into another slot in the timetable, and the other is a **ruin-and-recreate** operator (which generates a completely new solution at random).
- At each iteration your single **parent** solution should be used to generate a single **child** operator by using **one** of the mutation operators (either the swap or the ruin-and-recreate for the entire run of the algorithm).
- At the end of an iteration the algorithm should retain the parent or child that has the best fitness.
- At the end of an iteration the best (parent or child) solution's fitness should be added to a list of the best fitnesses, which is returned along with the best solution at the end of the optimisation.

Task 3.3 – Visualisation of results (15% of total mark)

You should run the algorithm twice for 500 iterations – once for each mutation operator. Repeat this 30 times, so that you get 30 fitness lists for the swap operator and 30 fitness lists for the ruin-and-recreate operator. Plot the **average**, **maximum** and **minimum** fitness at each iteration for each operator. You should plot them on the same graph so that they can be compared. You should be able to see which optimiser is best – state which in the notebook, and say why.

COURSEWORK DELIVERABLES

A Jupyter notebook has been provided on the Moodle for you to use for this coursework. You should implement your code in it, and submit it to the Moodle ahead of the deadline specified in the submission dates earlier in this document. **Please indicate which task each section of the notebook refers to using a Markdown cell**.

Please check your submitted files are correct by downloading them again and checking that they work. **You will receive a confirmation receipt by email when your work has been properly submitted** – if you do not receive this email then your work has not been submitted.

ASSESSMENT CRITERIA

Your work will be assessed according to the rubric found in Tables 2 and 3. Your mark for this piece of coursework will be based on an aggregation of the marks for each category. Marks will be awarded based on **both the demonstration and the** report.

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%	
Data preparation (10%)	Data is badly loaded into the program and there is no real preparation of the data.	The data is loaded correctly, though inefficiently, but there is no preparation.	The data is efficiently loaded into the program and there is an attempt at normalisation.	The data is loaded and normalised, with some inefficiencies.	The data is loaded and normalised, and the code to do so is efficiently written.	/10
Regression (20%)	Very little or no attempt at training or testing regression models. Poor organisation of code.	There has been an attempt at training one or more of the models, but there is no testing. The organisation of the code is poor.	All three models are trained, but there are problems with the testing. The organisation of the code is poor.	All three models are trained and tested correctly. The code to do so is repetitive and not well organised.	All three models are trained and tested correctly. The code used to do so is well organised and efficient.	/20
Assessment of results (20%)	Very little or no attempt at assessing the regression results.	There has been an attempt at using mean absolute error, but little or no use of cross validation.	Mean absolute error is used in combination with cross validation. The presentation of results should be improved.	Mean absolute error is used in combination with cross validation, and the results are presented with a boxplot. Some inefficiencies in the code organisation.	Mean absolute error is used in combination with cross validation, and the results are presented with a boxplot. The code is efficient and well organised.	/20

Table 2: Feedback Template for Assessment 2 (Machine Learning part)

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%	
Generation of random solutions (10%)	The fitness function is incorrect. Poor attempt at loading the data. Generation of random solutions does not work.	The data is loaded. There has been an attempt at the fitness function, but it is incorrect. Generation of random solutions does not work.	The data is loaded and the fitness function is close to correct. Generation of random solutions does not work.	The data is loaded and the fitness function is close to correct. Solution generation is close to working. Code is inefficiently designed.	The data is loaded and the fitness function is correct. Random solutions are generated correctly. The code to do so is efficiently structured.	/10
Algorithm implementation (25%)	Mutation operators are mostly incorrect or missing. The algorithm is incorrect – selection is missing or not working and the fitness archive is missing.	One mutation operator has been attempted. Some of the algorithm is present, but it is incomplete or incorrect. The fitness archive has been attempted.	Two mutation operators have been attempted but are incomplete or incorrect. The algorithm is partially correct and the fitness archive works correctly.	The mutation operators work and the algorithm is mostly implemented correctly. The fitness archive stores fitnesses correctly.	The mutation operators and algorithm have been implemented correctly. The fitness archive stores fitnesses correctly. The code is efficiently structured and well organised.	/25
Visualisation of results (15%)	No attempt at producing a plot, or the plot is incorrect. Experiments are not run for the correct number of repeats.	A graph shows the partial results required. The experiment was run incorrectly.	A graph shows the correct results required but the experiment was run incorrectly.	A graph shows the correct results and the correct experimental setup has been followed. No analysis of the results.	A graph shows the correct results and the correct experimental setup has been followed. The best mutation operator is identified and justified.	/15

Table 3: Feedback Template for Assessment 2 (Evolutionary Computation part)