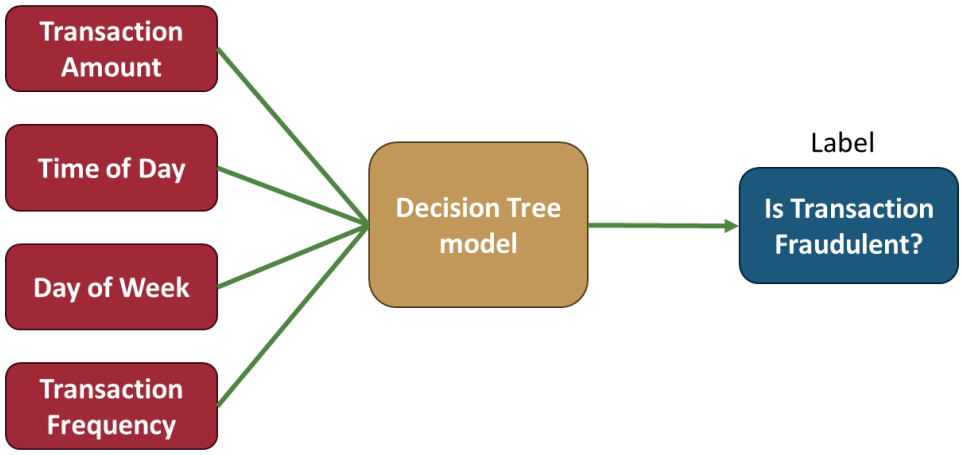
Assessment Brief

Module title	Artificial Intelligence
Module code	COMP2611
Assignment title	Coursework #2 - Decision trees
Assignment type and description	A Programming assignment on decision trees using Python via the Scikit-Learn package.
Rationale	This assignment will help to have a good understanding of how to work with decision trees and use them for real-world applications.
Tasks to be completed	10 Python programming tasks
Weighting	Marked from 100, worth 20% of the overall module grade.
Submission deadline	25th April
Submission method	Online submission via Gradescope
Feedback provision	Will be delivered via Autograder
Learning outcomes assessed	-Employ artificial intelligence techniques to solve real-world problems, in this instance: decision trees.
Module leader	Brandon Bennet (Lead), Arash Rabbani (Co-Lead)

1. Assignment guidance

This coursework focuses on implementing various tasks related to decision tree classification using the scikit-learn library in Python. The goal is to process and analyze a dataset, train decision tree models, and optimize their performance. We are going to build a model that can detect fraudulent transactions. You will start by reading and filtering the data, splitting it into train and test groups and training a decision tree. Then it will be pruned by increasing the cost complexity parameter. This parameter is then optimized and finally the depths of all trees are compared. The deeper the tree, means it is a more complex model, but it is not necessarily a more accurate one.

The dataset used is a series of transactions and associated features. Each row corresponds to a transaction, and the columns include the transaction amount, time of day, transaction frequency, day of the week, and a binary label indicating whether the transaction is fraudulent (label 1) or not (label 0). Here is the list of dataset headers and description:
transactionAmount: The amount involved in the transaction.
timeOfDay: The time of day when the transaction occurred.
transactionFrequency: How many times a transaction happened in a row.
dayOfWeek: The day of the week when the transaction took place.
labels: Binary labels where 1 indicates a fraudulent transaction and 0 indicates a legitimate one.



You are provided with a template python file that has several functions that need to be completed to carry out the abovementioned decision tree training and testing. In addition, there is a section at the end of the template that starts with **if __name__ == "__main__"** and is called **Main block.** This block demonstrates the usage of functions with the provided dataset. Also, this section is for your local tests to see if the model works reasonably well. For example, it is expected that pruning will maintain or reduce the depth of the model and if that is not the case, it means something is going wrong in your code.

2. Assessment tasks

Download the "CW2.py" and "DT.csv" files from Minerva, under the section for assessment #2. "CW2.py" will act as the template of your code and you should fill out the following functions each of which is presented as a task:

Load the Data (Task 1): [10 marks]
Function: load_data(file_path, delimiter=',')
Description: Load data from a CSV file and return the number of rows, data array, and header list.
Instructions: Provide the file path, specify the delimiter as given. Use pandas to read the csv but the output variable 'data' should be a *numpy* array with rows representing different samples and columns showing different features. 'num_rows' variable is an integer and 'header_list' is a python list containing all the headers.
Filter the data (Task 2): [10 marks]
Function: *filter_data(data)*
Description: Remove rows containing *missing* values from the dataset and give the filtered dataset back as output. The missing values are not actually empty, they have values of "-99".
Instructions: use *numpy*.
Data statistics (Task 3): [10 marks]
Function: *statistics_data(data)*
Description: Write a function that calculates the coefficient of variation for each feature.
Instructions: use *numpy*. Coefficient of variation is standard deviation divided by the average of a feature. 'coefficient_of_variation' variable is a float number.
Split the data (Task 4): [10 marks]
Function: *split_data(data, test_size=0.3, random_state=1)*
Description: Write a function that splits the dataset into training and testing sets while maintaining label ratios.
Instructions: Use train_test_split of sklearn to make sure that the sampling is stratified, meaning that the ratio between 0 and 1 in the label column stays the same in train and test groups. Keep the *random_state* equal to 1 as it already is.
Train a decision tree (Task 5): [10 marks]
Function: *train_decision_tree(x_train, y_train, ccp_alpha=0)*
Description: write a function that trains a decision tree model with a specified cost complexity parameter.
Instructions: from *sklearn* use *DecisionTreeClassifier* class and its *fit* method.
Make predictions (Task 6): [10 marks]
Function: *make_predictions(model, X_test)*
Description: write a function that takes a trained decision tree model and a set of features (*x_test*) and gives back the predicted label (*y_test_predicted)*.
Evaluate Model (Task 7): [10 marks]
Function: *evaluate_model(model, x, y)*
Description: Write a function that evaluates the performance of the model on the test dataset, returning accuracy and recall as two float numbers.
Optimal Cost Complexity Parameter (Task 8): [10 marks]
Function: *optimal_ccp_alpha(x_train, y_train, x_test, y_test)*
Description: Write a function to determine the optimal cost complexity parameter that balances model simplicity and accuracy. The optimal cost complexity parameter leads to a simpler model but shows similar (+-1%) accuracy compared to the unpruned model.

Instructions: Start with *ccp_alpha=0* and gradually increase it while monitoring the obtained accuracy, as soon as it dropped more than 1% from the unpruned accuracy, stop the training and report the last found *ccp_alpha* (float number).

Decision Tree Depth (Task 9): [10 marks]

Function: *tree_depths(model)*

Description: Write a function that gets a decision tree model and gives back the depth of it. So, in the test stage we can use this function to compare the depth of the initial, pruned, and optimized decision trees.

Feature importance (Task 10): [10 marks]

Function: *important_feature(x_train, y_train, header_list)*

Description: train a decision tree model and increase the **Cost Complexity Parameter** until the depth of the tree reaches 1. Give back the name of the remaining feature as a **string**. This feature can be interpreted as the most important feature to predict fraudulent transactions in the studied dataset.

3. General guidance and study support

Please refer to the module handbook, teaching slides, and skills@library.

4. Assessment criteria and marking process

Your submission will be marked via AutoGrader and will go through 10 tests to evaluate the 10 tasks. The tests are similar to the function calls in the "main block" of the template that is shared with you (CW2.py) and will fail if values or size of the requested output does not match to the solution.

When you submit your code, you will get example feedback for Tasks 1 to 5. That gives you a good idea showing your code is on the right track and you have secured a pass mark. Feedback on the rest of the tasks is reserved for the final marking of the coursework and will not be available during the submission period. For the final marking the used dataset will change so the answers cannot be hard coded or reversed engineered. As a note, the "Main block" of your submission won't be marked. Marking will be done only by calling the "functions" and comparing their output with the solution code.

5. Presentation and referencing

Include proper recognition and references for any external sources, such as code snippets, papers or repositories, if used. You should cite the external sources in the form of comments at the end of your submission. There is an example at the end of the template.

6. Submission requirements

This is an individual assessment, and each student should make their own submission. You need to submit only the file named "CW2.py" and put your **name** and **email** as a comment at the top of the file in the designated lines. The submitted file should keep its original name as it is "CW2.py", you can download the template file from Minerva under Assessment #2 section, as well as the required dataset which is a csv file.

Please do not use any additional Python packages other than those already imported in the provided code. The required/allowed packages are **pandas**, **numpy**, and **scikit-learn**. (**warnings** and **os** packages are there as well for some housekeeping, keep them as it is).

Please do not change the names of the functions provided in the code, nor their input/output argument name and order. Use the exact function names as specified in the template.
Please do not alter the names of the Python file or any associated files provided in the coursework. Changing name of the functions and files will cause losing the mark for affected tasks or even for the entire coursework.

7. Academic misconduct and plagiarism

Leeds students are part of an academic community that shares ideas and develops new ones.

You need to learn how to work with others, how to interpret and present other people's ideas, and how to produce your own independent academic work. It is essential that you can distinguish between other people's work and your own, and correctly acknowledge other people's work.

All students new to the University are expected to complete an online Academic Integrity tutorial and test, and all Leeds students should ensure that they are aware of the principles of Academic integrity.

When you submit work for assessment it is expected that it will meet the University's academic integrity standards.

If you do not understand what these standards are, or how they apply to your work, then please ask the module teaching staff for further guidance.

By submitting this assignment, you are confirming that the work is a true expression of your own work and ideas and that you have given credit to others where their work has contributed to yours.

8. Assessment/ marking criteria grid

Tasks are worth 10 marks each, making a total of 100 marks equivalent to 20% of the module final grade.