Data Mining and Decision Systems
Workshop 6
Association Rules & Decision Trees

Aims of the workshop

The first part of today's workshop is to consider an example of association rules and how to calculate important metrics surrounding them to cement the knowledge from last week's lecture.

Having covered the creation of a Linear Regression model with Scikit learn, you should be familiar with model initialisation, fitting, and predicting. From this you should be able to produce error metrics for both training and test datasets, as well as knowing how to calculate confusion matrix plots.

This week we will look at Decision Trees within Scikit learn, and how these can be created and visualised for explainability of your trained model. Many of these steps are near identical to what you already know from Workshop 5.

Please see 'Useful Information' below on how to lookup certain Python functionality. The concept behind this workshop is about discovery, and experimentation surrounding topics covered so far.

Feel free to discuss the work with peers, or with any member of the teaching staff.

Useful Information

Seaborn

Seaborn is a high-level plotting library for Python. Written on top of Matplotlib it enables rich and intricate plots whilst maintaining readability and ease of use. You can access the library's web page at the following link:
https://seaborn.pydata.org/

For API Reference, consider: https://seaborn.pydata.org/api.html

You may find https://seaborn.pydata.org/tutorial.html useful for understanding more functionality, and introducing you to each aspect.

Matplotlib

Matplotlib is a powerful 2D plotting library for Python. It is the de-facto standard for the community. Whilst matplotlib is considered powerful, this comes at the expense of verbosity. Those wishing to further manipulate their diagrams and plots may wish to seek out matplotlib documentation (https://matplotlib.org/3.1.1/api/index.html) to enable more advanced graphics manipulation.

Scikit Learn (Sklearn)

Scikit-learn is a machine learning library for Python, providing simple tools to work with complex models and to facilitate data analysis. Sklearn is built on top of common frameworks such as numpy (powerful arrays), scipy (scientific python library), and matplotlib(2D plotting).
Most of the techniques covered within lectures will be available in some form through Sklearn, such as regression/classification models, and cross validation.

Functions of note: load_iris, train_test_split, DecisionTreeClassifier, fit, predict, plot_tree, plot_confusion_matrix.

Reminder

We encourage you to discuss the contents of the workshop with the delivery team, and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

The contents of this workshop are not intended to be 100% complete within the 2 hours; as such it's expected that some of this work be completed outside of the session. Exercises herein represent an example of what to do; feel free to expand upon this.

Remember the magic: `%matplotlib inline`
Remember common imports if you need them (you may not):

Dataframe as df
Seaborn as sns
Matplotlib.pyplot as plt
Numpy as np

Association Rules

*These association rule exercises are **not** programming tasks. You are expected to calculate these by hand, and to produce these tables. You may wish to use Microsoft Word to create these.*

Exercise 1: A popular games distribution platform is interested in the types of games its users are purchasing. Below is a small sample of data held on their users.

User	Games
U1	GTA V, Red Dead Redemption 2
U2	Cities Skylines, SimCity
U3	GTA V
U4	Cities Skylines, GTA V
U5	GTA V, Factorio
U6	Cities Skylines, Factorio
U7	GTA V, Cities Skylines, Factorio

Using the systematic approach from last week's lecture produce a support table. Remember to consider:

i) Singular items
E.g Support(GTA V) = 5 / 7
It is in U1, U3, U4, U5, U7.
ii) Multiple Item sets (E.g GTA V, Factorio). (2-item set, 3-item set, etc)
E.g How many User records does GTA V and Factorio appear in together?
U5 & U7 = 2/7.

Only include entries in your support table if the support for that given itemset is > 0%

Exercise 2: Now you have a full support table for all singular and multiple itemsets with support above 0, produce a list of association rules which lead to the decision on whether a user has GTA V purchased. E.g (X -> GTA V) or (X, Y -> GTA V). Looking at your support table will help you know which itemsets to use (as they have support > 0 and exist). This is

any combination of items which lead to GTA V, singular or multiple. Remember X -> Z or X, Y -> Z.

Exercise 3: From these generated association rules, produce confidence metrics for each of them.

Exercise 4: Finally, calculate the improvement over random of each of the generated association rules. What do these values tell you about the generated association rules?

Check the answers to the above exercises with a member of the delivery team.

Decision Trees

These exercises for Decision Trees are Python Notebook exercises.

Exercise 5: From Sklearn datasets import load_iris and load the iris dataset. This dataset has some attributes such as data, feature_names, target, and target_names which you may find useful.

Exercise 6: Create a random training-test split, of 70-30, by using shuffle=True. Ensure that the 4 arrays returned are correct. You can print the length of these by passing them to the len function. E.g len(my_training_x). Or, alternatively, you can inspect the first element my_training_x[0] to see if it is what you expect. ***Don't mix up your training X, training Y, testing X, testing Y datasets.***

Exercise 7: Create a DecisionTreeClassifier from sklearn, assigned to a variable called model. For now leave parameters as default.

Exercise 8: Fit the newly created model to your training data from Ex 6.

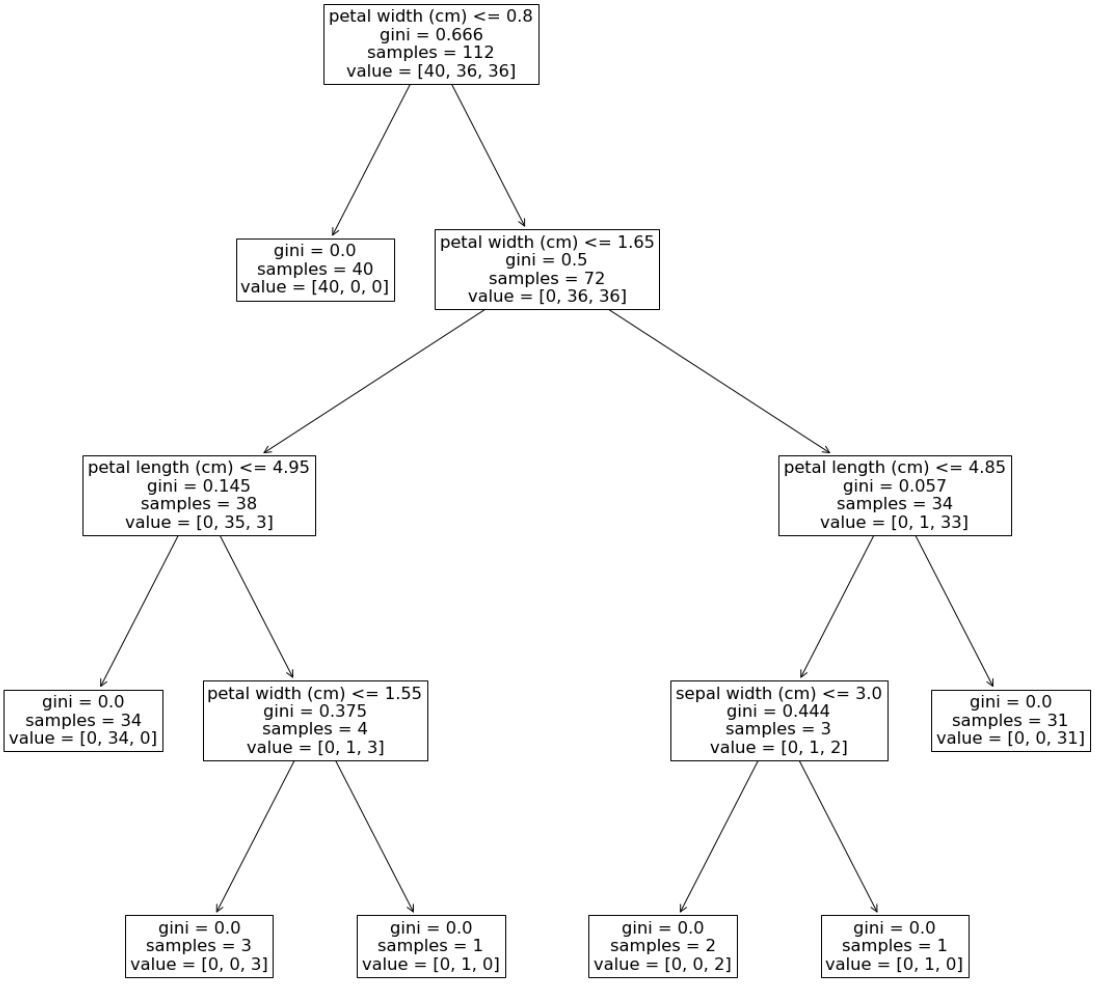
Exercise 9: Using the plot_tree from sklearn (newly added), plot your trained Decision Tree. You can alter the size of the produced graphic using matplotlib rcParams. In this case 20x20 should suffice; however, you can change these.

```
from sklearn.tree import plot_tree # Only available in Scikit-learn >=
0.21

import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (20,20)

plot_tree( model, feature_names=iris_dataset.feature_names )
```

You should get something like the following (it will not be identical, this is fine):



Exercise 10: Now you have a trained model you can make predictions on unseen data. Using the test set input data (X), invoke predict on your trained model to obtain predicted class outputs. Sample output should look something like [0, 0, 1, 2, 0, …, 2, 1]. Assign this to a variable (notably pred_y).

Exercise 11: Using sklearn confusion matrices, use the test set predicted output, and the test set ground truth data to plot a confusion matrix. Hint: You did this last workshop for dummy data.

Exercise 12: Reload the cell containing your shuffled/random train-test split, and re-run code up to Ex 11. Does the structure of your Decision Tree change? Is it deeper? Does the accuracy of the trained model change? I.e Are some classes confused now, when they weren't before?

Exercise 13: Explore the additional parameters when creating the DecisionTreeClassifier. E.g Max_depth. How does this impact the accuracy of the model? What risks does this pose if you create a very deep tree in terms of generalisation? What if it's a shallow tree?

The Extended Exercises are optional, and are offered as an advanced supplement for those who have completed the existing work and wish to expand on their knowledge and challenge themselves further.

Extended Exercise 1: Select any 2 of the 4 input features of the iris dataset (remember feature_names may help you know what each index means). Using a newly trained DecisionTreeClassifier on these features, and its nodes, overlay these linear decision boundaries on a 2D plot. Remember to use Hue as a 3rd dimension to show class membership.

Caution: This exercise is extremely difficult, and requires recursive tree traversing with a sklearn object and very advanced matplotlib usage. "*Down the rabbit hole* away little Alice goes. Follow her at *your own peril*, but beware *of* the world you are about to *enter*."