COMP809 – Lab 7: Decision Tree Classifier Part II

This lab is divided in two parts.

- Part I is to evaluate the performance of the classification tree on Iris dataset.
- Part II, Decision Tree Classifier is used to predict car safety using criterion 'gini index'.

Part I:

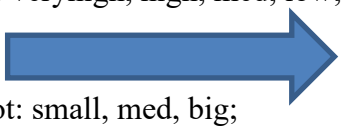
Task 1: load the iris dataset, create the data frame and build a Decision tree assuming '*max_depth*' and '*max_leaf_nodes*' are 3 and 4 respectively. Once your model is fitted, then the make prediction.
Task 2: Create the confusion matrix and explain the findings.
Task 3: Generate the model performance summary and explain your findings.

Part II:
Decision Tree Classifier is used to predict car safety using criterion 'gini index'. The Decision Tree Classification is implemented with Python and Scikit-Learn.

The Car Evaluation Dataset:

The Car Evaluation dataset was derived from a simple hierarchical decision model originally developed for the demonstration of Expert system for decision making (Bohanec & Rajkovič, 1990). It contains examples with the structural information removed.
The target attribute (*class*) directly relates the six input attributes namely:

- ***buying***: veryhigh, high, med, low;
- ***maint* (maintenance)**: veryhigh, high, med, low;
- ***doors***: 2, 3, 4, 5more;
- ***persons***: 2, 4, more,
- ***lug_boot*** (luggage boot: small, med, big;
- ***safety***: low, med, high.



Both ***doors*** and ***persons*** will also be treated as categorical variables

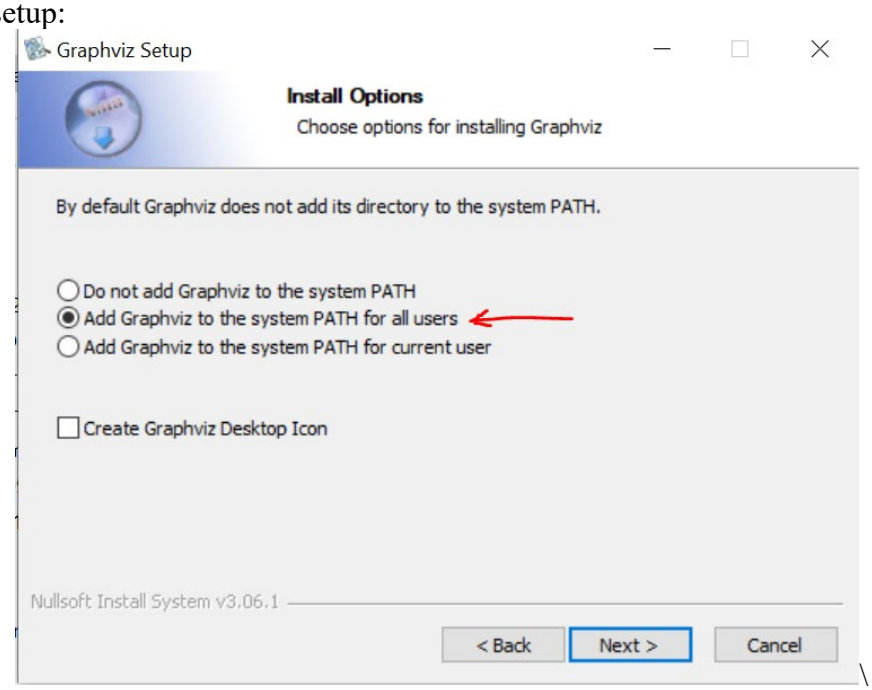
Class values are categorised as:

- unacc (unacceptable),
- acc(acceptable),
- good,
- vgood (very good)

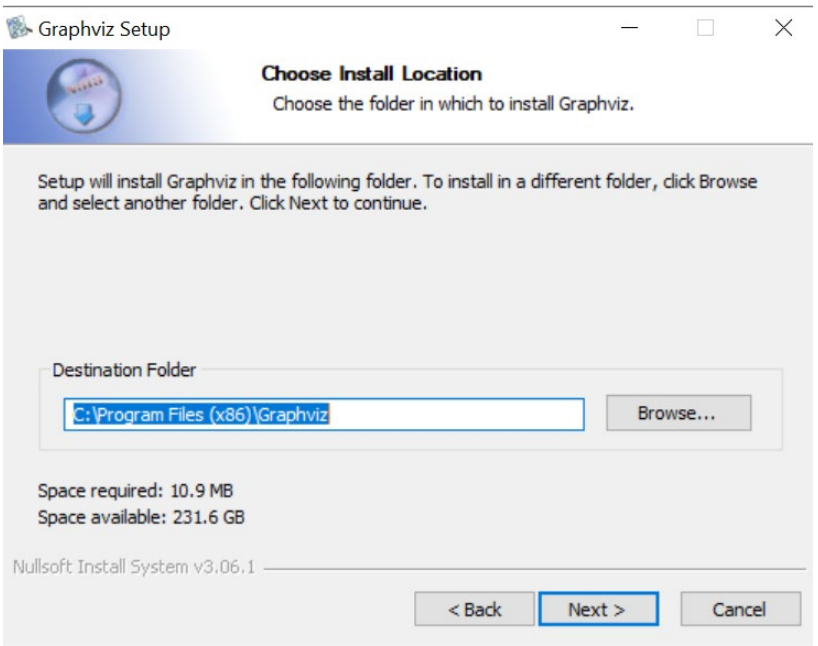
Because of known underlying concept structure, this database may be particularly useful for testing constructive induction and structure discovery methods. More information about the dataset (metadata) are provided in Appendix I.

Download Files:
To start with download lab 6 files: Iris dataset (car_evaluation. csv) and the python code file (lab7Part1and2.ipynb).

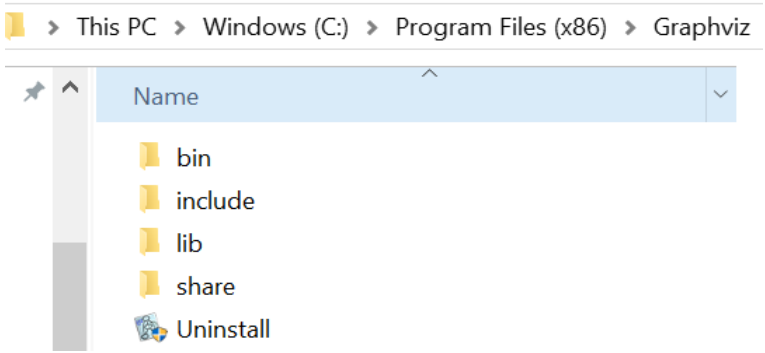
Download and Install 'graphviz':
Follow the link to download and install the appropriate 'graphviz' package. In below example you see 'graphviz' package for wind32 is being installed.
-Install option setup:



-Note the path:



-Once installation is finished you should be able to see the related folders:



-Add the package to your system path (already added in 'Lab06_DTree.ipynb' file).

```
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz/bin/'
```

Import Libraries:

Configuring the classifier will be achieved using Python's sklearn library. Evaluation will be done via sklearn but use a dedicated set of methods designed specifically for computing metrics.

Create a Data Frame:
Create the data frame (*mycar*), check the dimension and preview the data frame:

In [2]:
```
#Import dataset
mycar = pd.read_csv('car_evaluation.csv', header=None)

# view dimensions of dataset
print(mycar.shape)

# preview the dataframe
mycar.head()
```

```
(1728, 7)
```

Out[2]:

	0	1	2	3	4	5	6
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

The data frame column names are not informative. Using the metadata, add meaningful name to the columns. Then review the data frame and check for the possible of missing values. Are there any missing values in your dataset?

```
'''
Rename column names: give proper column name
'''
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
mycar.columns = col_names

#  preview the dataframe
print(mycar.head())
#Check the number of missing values
mycar.isnull().sum()
```

Create the Decision Tree Classifier

1. Data preparation:

Task 1.1: Declare the input and target variable and create the splits (test (30%)). How many instances are in each train and test set (check the shape of train and test sets)?

Task 1.2: Perform feature transformation to 'Encode' categorical variables. This can be done through ordinal encoding.

2. Create the Classification Tree Model

Task 2.1: Set up a classification tree model ('*mytree_gini'*) with criterion '*gini*' index using *max_depth*=3. The value of *max_depth* (3) is an arbitrary number (you should be able to tune this parameter following lab 05 instruction). How many nodes are used in your baseline model? Print the tree to visually inspect it. Explain the tree structure.

Task 2.2: Fit the model and make prediction using the test set. Calculate and report the accuracy of your model.

Task 2.3: Explore the outcome of the prediction. How many unique classes are 'predicted'? How is it different from the 'actual' test set?

TODO Tasks

Create the Confusion Matrix

Q1: Provide the confusion matrix and explain the classification model performance. According to 'sklearn documentation' *Wikipedia and other references may use a different convention for axes*. To make sure your interpretation is correct use the result of **Task 2.3** as a guide.

Q2: Provide the evaluation metrics including 'accuracy, precision, recall, and f1-score). Explain your findings.

References

Bohanec, M., & Rajkovič, V. (1990). DEX: An Expert System Shell for Decision Support. *Sistemica*, 145-157.

Appendix I

The car evaluation dataset structure:

- Buying: price
- maint: price of the maintenance
- Doors: number of doors
- Persons: capacity in terms of persons to carry
- lug_boot: the size of luggage boot
- safety: estimated safety of the car

Attribute Information:

- Class Values:
    - unacc, acc, good, vgood
- Attributes:
    - buying:
        - vhigh, high, med, low.
    - maint:
        - vhigh, high, med, low.
    - doors:
        - 2, 3, 4, 5more.
    - persons:
        - 2, 4, more.
    - lug_boot:
        - small, med, big.
    - safety:
        - low, med, high.