Module imports

In [63]:
```
import numpy
import scipy
import pandas
import matplotlib.pyplot as plt
import sklearn
```

This week we'll focus on how to generate random problems and apply CART and Random Forest.

# CART

Random problem generators and data splitter:

In [64]:
```
from sklearn.datasets import make_regression
from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split
```

We can use graphviz to visualize decision trees but its installation requires some updates to path variables, so it's commented out.

In [65]:
```
# import graphviz
```

Generating the data using a random problem generator:

In [66]:
```
X, y = make_classification(n_samples=500, n_features=10, n_informative=6, n_re
dundant=4, random_state=433,
                           n_classes=2)
```

These generators are useful for understanding how a model builder works without dealing with data.

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=433)

from sklearn.tree import DecisionTreeClassifier
# from sklearn.tree import export_graphviz

model = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=4,
class_weight=None)
model.fit(X_train, y_train)

train_prediction = model.predict(X_train)
test_prediction = model.predict(X_test)

# graph_data = export_graphviz(model, filled=True, rounded=True)  # if graphvi
z is installed
# graph = graphviz.Source(graph_data)

# graph
```

We can take a quick look at the model summary (Dobilas 2022)

In [68]:
```
from sklearn.metrics import classification_report # for model evaluation metri
cs

print('*************** Tree Summary ***************')
print('Classes: ', model.classes_)
print('Tree Depth: ', model.tree_.max_depth)
print('No. of leaves: ', model.tree_.n_leaves)
print('No. of features: ', model.n_features_in_)
print('--------------------------------------------------------')
print("")

print('*************** Evaluation on Test Data ***************')
score_te = model.score(X_test, y_test)
print('Accuracy Score: ', score_te)
# Look at classification report to evaluate the model
print(classification_report(y_test, test_prediction))
print('--------------------------------------------------------')
print("")

print('*************** Evaluation on Training Data ***************')
score_tr = model.score(X_train, y_train)
print('Accuracy Score: ', score_tr)
# Look at classification report to evaluate the model
print(classification_report(y_train, train_prediction))
print('--------------------------------------------------------')
```

```
*************** Tree Summary ***************
Classes:  [0 1]
Tree Depth:  4
No. of leaves:  14
No. of features:  10
--------------------------------------------------------

*************** Evaluation on Test Data ***************
Accuracy Score:  0.7272727272727273
              precision    recall  f1-score   support

           0       0.63      0.92      0.75        73
           1       0.90      0.58      0.70        92

    accuracy                           0.73       165
   macro avg       0.77      0.75      0.73       165
weighted avg       0.78      0.73      0.72       165

--------------------------------------------------------

*************** Evaluation on Training Data ***************
Accuracy Score:  0.826865671641791
              precision    recall  f1-score   support

           0       0.77      0.96      0.85       178
           1       0.93      0.68      0.79       157

    accuracy                           0.83       335
   macro avg       0.85      0.82      0.82       335
weighted avg       0.85      0.83      0.82       335

--------------------------------------------------------
```

We can follow a similar process to fit a linear model:

In [69]:
```
X, y = make_regression(n_samples=400, n_features=5, n_informative=3, random_st
ate=433)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=433)

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor(criterion='squared_error', splitter='best', max_
depth=4)
model.fit(X_train, y_train)

train_prediction = model.predict(X_train)
test_prediction = model.predict(X_test)

print('*************** Tree Summary ***************')
print('Tree Depth: ', model.tree_.max_depth)
print('No. of leaves: ', model.tree_.n_leaves)
print('No. of features: ', model.n_features_in_)
print('------------------------------------------------------------')
print("")

print('*************** Evaluation on Test Data ***************')
score_te = model.score(X_test, y_test)
print('Accuracy Score: ', score_te)
print('------------------------------------------------------------')
print("")

print('*************** Evaluation on Training Data ***************')
score_tr = model.score(X_train, y_train)
print('Accuracy Score: ', score_tr)
print('------------------------------------------------------------')
```

```
*************** Tree Summary ***************
Tree Depth:  4
No. of leaves:  16
No. of features:  5
------------------------------------------------------------

*************** Evaluation on Test Data ***************
Accuracy Score:  0.742429801079534
------------------------------------------------------------

*************** Evaluation on Training Data ***************
Accuracy Score:  0.8596269596674244
------------------------------------------------------------
```

# Random Forest

Using Random Forest for a classification problem:

In [70]:
```
from sklearn.ensemble import RandomForestClassifier

X, y = make_classification(n_samples=1000, n_features=8, n_informative=5, rand
om_state=433)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=433)

model = RandomForestClassifier()  # your results will vary if you don't set a
 random_state!
model.fit(X_train, y_train)
# make a single prediction
prediction = model.predict([X_test[200]])  # predict expects a list of rows
print(f'Row 200, Prediction: {prediction[0]:.2f}, Actual: {y_test[200]:.2f}'
      f', R2: {model.score(X_test, y_test)*100:.2f}%')
```

```
Row 200, Prediction: 1.00, Actual: 1.00, R2: 92.42%
```

Similarly, we can build a regression model with:

In [71]:
```
from sklearn.ensemble import RandomForestRegressor

X, y = make_regression(n_samples=1000, n_features=8, n_informative=5, random_s
tate=433)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=433)

model = RandomForestRegressor(random_state=433)
model.fit(X_train, y_train)
# make a single prediction
prediction = model.predict([X_test[155]])  # be careful to pass an array of ro
ws
print(f'Row 155, Prediction: {prediction[0]:.2f}, Actual: {y_test[155]:.2f}'
      f', R2: {model.score(X_test, y_test)*100:.2f}%')
```

```
Row 155, Prediction: 134.22, Actual: 235.03, R2: 89.00%
```