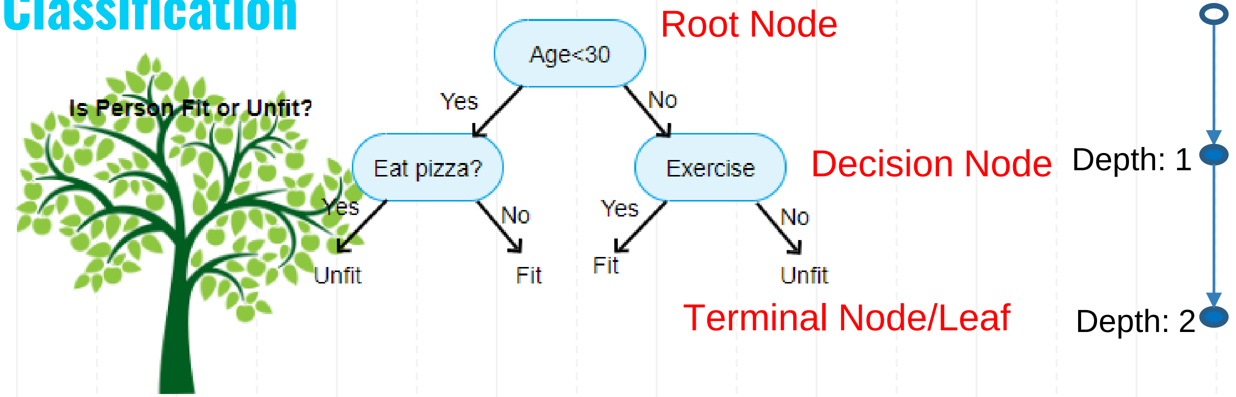
EE2211 Introduction to Machine Learning

T14 & T22, Chua Dingjuan elechuad@nus.edu.sg
Materials @ tiny.cc/ee2211tut

Tutorial 9

Decision Trees….

Decision Trees - Classification



- Regression Tree
  - (ypred) $\hat{y}_m = \frac{1}{J_m}\sum_{j=1}^{J_m} y_j$
  - MSE
    - MSE = $S_m = \frac{1}{J_m}\sum_{j=1}^{J_m}(y_j - \hat{y}_m)^2$
- Classification Trees
  - Gini
    - $Q_m = 1 - \Sigma_{i=1}^{K}$
  - Entropy
    - $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i$
  - Misclassification Rate
    - $Q_m = 1 - \max_i p_i$

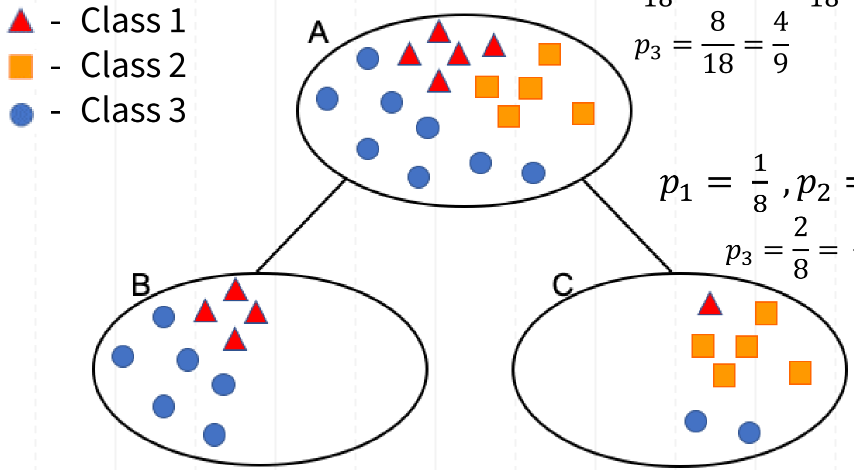
- Random Forest
  - Bootstrapping

Discussion of Solutions 9

Q1,2,3,4

Question1

Compute the Gini impurity, entropy, misclassification rate for nodes A, B and C, as well as at depth 1 of the decision tree shown below.



$p_1 = \frac{5}{18}, \; p_2 = \frac{5}{18},$

$p_3 = \frac{8}{18} = \frac{4}{9}$

$p_1 = \frac{1}{8}, p_2 = \frac{5}{8},$

$p_3 = \frac{2}{8} = \frac{1}{4}$

$p_1 = \frac{4}{10} = \frac{2}{5}, p_2 = \frac{0}{10} = 0, \; p_3 = \frac{6}{10} = \frac{3}{5}$

(a) SOLUTION

GINI IMPURITY $= 1 - \Sigma_{i=1}^{K} p_i^2$

Node A: $Q_A = 1 - p_1^2 - p_2^2 - p_3^2$

$= 1 - \left(\frac{5}{18}\right)^2 - \left(\frac{5}{18}\right)^2 - \left(\frac{4}{9}\right)^2 = 0.6481$

Node B: $Q_B = 1 - \left(\frac{2}{5}\right)^2 - (0)^2 - \left(\frac{3}{5}\right)^2 = 0.48$

Node C: $Q_C = 1 - \left(\frac{1}{8}\right)^2 - \left(\frac{5}{8}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.5312$

Overall Gini at Depth 1 :

proportion of data samples in B x Q_B + proportion x Q_C

$Q_1 = \left(\frac{10}{18}\right) 0.48 + \left(\frac{8}{18}\right) 0.5312 = 0.5028$

Total 10 samples in B, out of total samples 18 (in A)

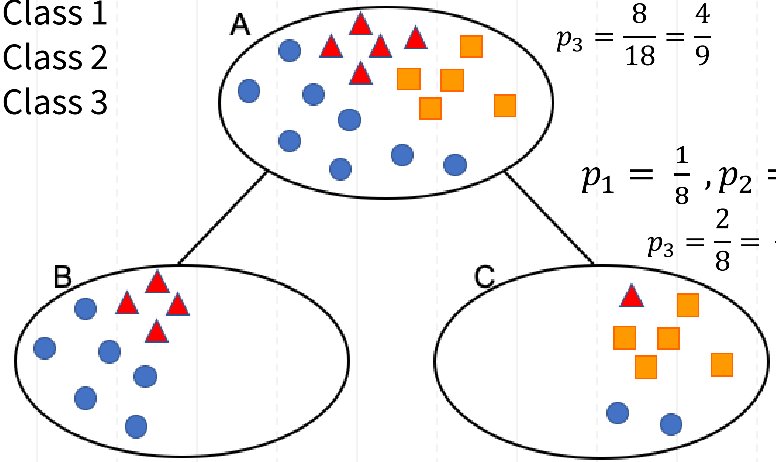
Question1

Compute the Gini impurity, entropy, misclassification rate for nodes A, B and C, as well as at depth 1 of the decision tree shown below.

- ▲ - Class 1
- ■ - Class 2
- ● - Class 3

$p_1 = \frac{5}{18}, \; p_2 = \frac{5}{18},$

$p_3 = \frac{8}{18} = \frac{4}{9}$



$p_1 = \frac{1}{8}, p_2 = \frac{5}{8},$

$p_3 = \frac{2}{8} = \frac{1}{4}$

$p_1 = \frac{4}{10} = \frac{2}{5}, p_2 = \frac{0}{10} = 0, \; p_3 = \frac{6}{10} = \frac{3}{5}$

(b) SOLUTION

Entropy $= -\sum_i p_i \log_2 p_i$

Node A: $Q_A = -\left(\frac{5}{18}\right)\log_2\left(\frac{5}{18}\right) - \left(\frac{5}{18}\right)\log_2\left(\frac{5}{18}\right) - \left(\frac{4}{9}\right)\log_2\left(\frac{4}{9}\right) = 1.5466$

Node B: $Q_B = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - (0)\log_2(0) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.9710$

Node C: $Q_C = -\left(\frac{1}{8}\right)\log_2\left(\frac{1}{8}\right) - \left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) = 1.2988$

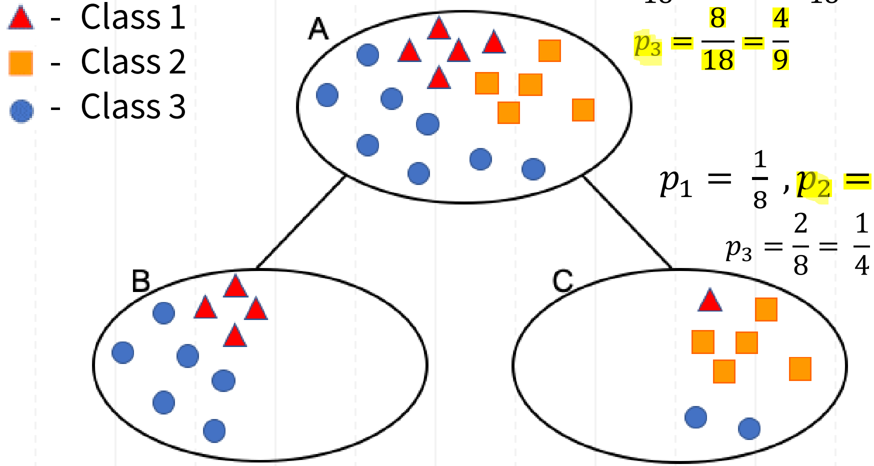
Overall Entropy at Depth 1 :

proportion of data samples in B x Q_B + proportion x Q_C

$Q_1 = \left(\frac{10}{18}\right) 0.9710 + \left(\frac{8}{18}\right) 1.2988 = 1.1167$

Question1

Compute the Gini impurity, entropy, misclassification rate for nodes A, B and C, as well as at depth 1 of the decision tree shown below.



$p_1 = \frac{5}{18}, \; p_2 = \frac{5}{18}, \; p_3 = \frac{8}{18} = \frac{4}{9}$

$p_1 = \frac{1}{8}, \; p_2 = \frac{5}{8}, \; p_3 = \frac{2}{8} = \frac{1}{4}$

$p_1 = \frac{4}{10} = \frac{2}{5}, \; p_2 = \frac{0}{10} = 0, \; p_3 = \frac{6}{10} = \frac{3}{5}$

(c) SOLUTION

Misclassification Rate $= 1 - \max_i p_i$

Node A: $Q_A = 1 - \max\left(\left(\frac{5}{18}\right), \left(\frac{5}{18}\right), \left(\frac{4}{9}\right)\right) = 1 - \left(\frac{4}{9}\right) = \frac{5}{9}$

Node B: $Q_B = 1 - \max\left(\left(\frac{2}{5}\right), 0, \left(\frac{3}{5}\right)\right) = 1 - \left(\frac{3}{5}\right) = \frac{2}{5}$

Node C: $Q_C = 1 - \max\left(\left(\frac{1}{8}\right), \left(\frac{5}{8}\right), \left(\frac{1}{4}\right)\right) = 1 - \left(\frac{5}{8}\right) = \frac{3}{8}$

Overall MR at Depth 1 :
proportion of data samples in Bx Q_B + proportion x Q_C

$Q_1 = \left(\frac{10}{18}\right)\left(\frac{2}{5}\right) + \left(\frac{8}{18}\right)\left(\frac{3}{8}\right) = \frac{7}{18} = 0.3889$

Tallies with misclassified samples in node B (4 red) and C (1 red+ 2 blue)

Question2

Calculate the overall MSE for the following data at depth 1 of a regression tree assuming a decision threshold is taken at x=5.0. How does it compare with the MSE at the root?

{x, y}: {1, 2}, {0.8, 3}, {2, 2.5}, {2.5, 1}, {3, 2.3}, {4, 2.8}, {4.2, 1.5}, {6, 2.6}, {6.3, 3.5}, {7, 4}, {8, 3.5}, {8.2, 5}, {9, 4.5}

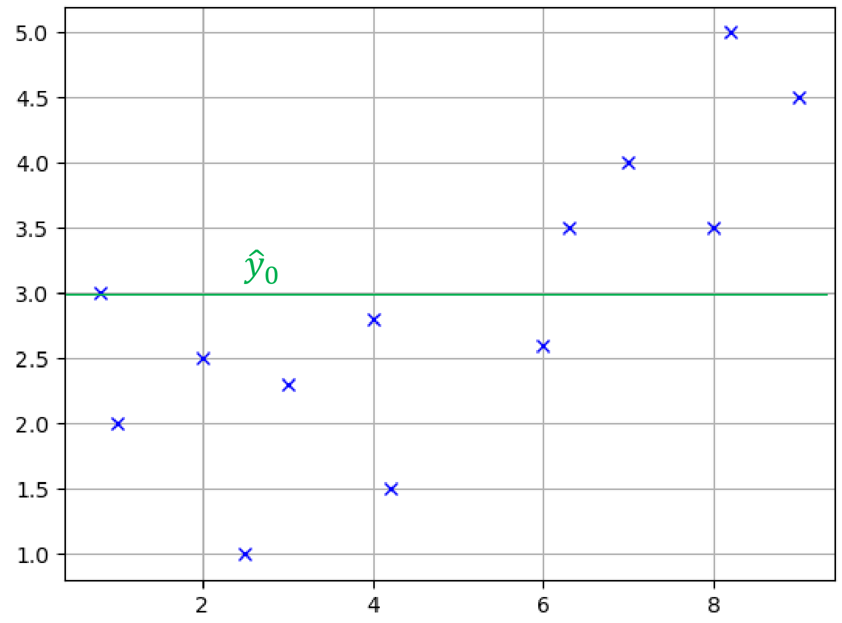
SOLUTION : (a) MSE AT ROOT

$\hat{y}_m = \frac{1}{J_m}\sum_{j=1}^{J_m} y_j$ = average of all the y values = $(2 + 3 + 2.5 + 1 + 2.3 + 2.8 + 1.5 + 2.6 + 3.5 + 4 + 3.5 + 5 + 4.5)/13$ = 2.9385

MSE = $\frac{1}{13}((2.6-\bar{y})^2 + (3.5-\bar{y})^2 + (4-\bar{y})^2 + (3.5-\bar{y})^2 + (5-\bar{y})^2 + (4.5-\bar{y})^2 + (2-\bar{y})^2 + (3-\bar{y})^2 + (2.5-\bar{y})^2 + (1-\bar{y})^2 + (2.3-\bar{y})^2 + (2.8-\bar{y})^2)$

= 1.2224

Root Node 13 Samples



Question2

Calculate the overall MSE for the following data at depth 1 of a regression tree assuming a decision threshold is taken at x=5.0. How does it compare with the MSE at the root?

{x, y}: {1, 2}, {0.8, 3}, {2, 2.5}, {2.5, 1}, {3, 2.3}, {4, 2.8}, {4.2, 1.5}, {6, 2.6}, {6.3, 3.5}, {7, 4}, {8, 3.5}, {8.2, 5}, {9, 4.5}

SOLUTION : (b) MSE AT Depth 1

When x>5, $\hat{y}_B$ = average of all the y values = $(2.6 + 3.5 + 4 + 3.5 + 5 + 4.5)/6$ = 3.85

When x>5, node B MSE = 0.5958

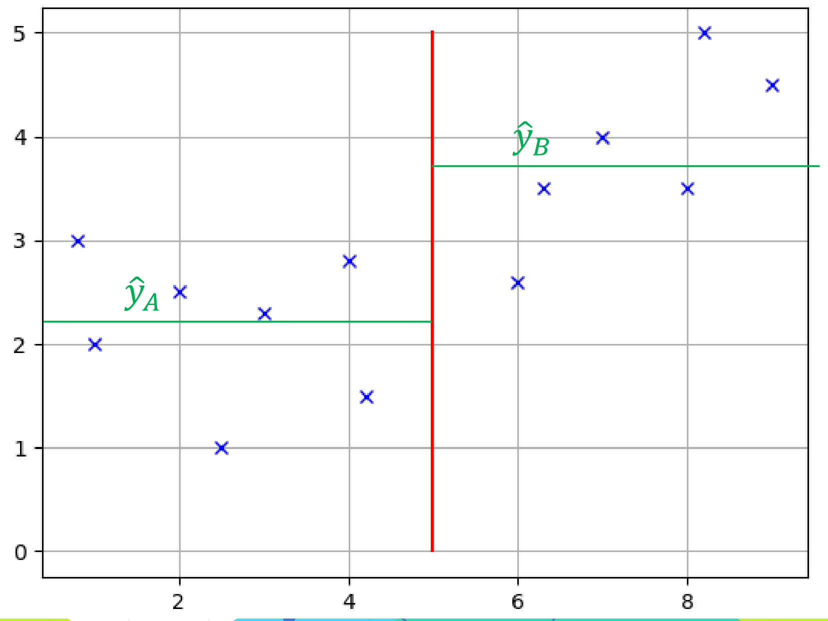
When x<=5, $\hat{y}_A$ = average of all the y values = $(2 + 3 + 2.5 + 1 + 2.3 + 2.8 + 1.5 + 2.6)/7$ = 2.1571

When x<=5, node A MSE = 0.4367

Overall MSE at depth 1 = $\underbrace{\frac{6}{13} \times 0.5958}_{B} + \underbrace{\frac{7}{13} \times 0.4367}_{A} =$ 0.5102

MSE has decreased from 1.2224 to 0.5102.

Nodes at Depth 1 13 Samples



Question 3

Import the Boston Housing dataset "from sklearn import datasets" and "boston = datasets.load_boston()". This data set contains 13 features and 1 target variable listed below. Use "LSTAT" as the input feature and "MEDV" as the target output. Fit a regression tree to depth 2 and compare your results with results generated by "from sklearn.tree import DecisionTreeRegressor" using the "mean square error" criterion.

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html

`sklearn.datasets.load_boston(*, return_X_y=False)` [source]

Load and return the boston house-prices dataset (regression).

Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5. - 50.

Returns:

data : Bunch
Dictionary-like object, with the following attributes.

data : *ndarray of shape (506, 13)*
The data matrix.

target : *ndarray of shape (506,)*
The regression target.

filename : *str*
The physical location of boston csv dataset.

New in version 0.20.

DESCR : *str*
The full description of the dataset.

feature_names : *ndarray*
The names of features

(data, target) : *tuple if* `return_X_y` *is True*
New in version 0.18.

```
# Boston House prices dataset
# Feature attributes:
# CRIM per capita crime rate by town
# ZN proportion of residential land zoned for lots over 25,000 sq.ft.
# INDUS proportion of non-retail business acres per town
# CHAS Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
# NOX nitric oxides concentration (parts per 10 million)
# RM average number of rooms per dwelling
# AGE proportion of owner-occupied units built prior to 1940
# DIS weighted distances to five Boston employment centres
# RAD index of accessibility to radial highways
# TAX full-value property-tax rate per $10,000
# PTRATIO pupil-teacher ratio by town
# B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
# LSTAT % lower status of the population
# MEDV Median value of owner-occupied homes in $1000's
```

x

y

Question 3

Import the Boston Housing dataset "from sklearn import datasets" and "boston = datasets.load_boston()". This data set contains 13 features and 1 target variable listed below. Use "LSTAT" as the input feature and "MEDV" as the target output. Fit a regression tree to depth 2 and compare your results with results generated by "from sklearn.tree import DecisionTreeRegressor" using the "mean square error" criterion.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor.fit
https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html#sphx-glr-auto-examples-tree-plot-tree-regression-py

sklearn.tree.DecisionTreeRegressor

class sklearn.tree.**DecisionTreeRegressor**(**, criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0*) [source]

Methods

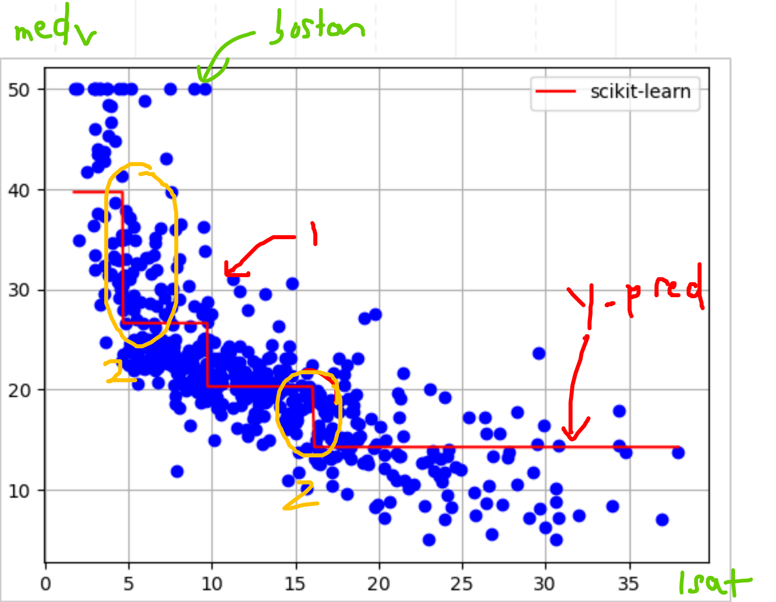
apply(X[, check_input])	Return the index of the leaf that each sample is predicted as.
cost_complexity_pruning_path(X, y[, ...])	Compute the pruning path during Minimal Cost-Compl
decision_path(X[, check_input])	Return the decision path in the tree.
fit(X, y[, sample_weight, check_input, ...])	Build a decision tree regressor from the training set (X, y
get_depth()	Return the depth of the decision tree.
get_n_leaves()	Return the number of leaves of the decision tree.
get_params([deep])	Get parameters for this estimator.
predict(X[, check_input])	Predict class or regression value for X.
score(X, y[, sample_weight])	Return the coefficient of determination R^2 of the pred
set_params(**params)	Set the parameters of this estimator.

```
#Using decision tree in sklearn
from sklearn.tree import DecisionTreeRegressor

#Creating an object called tree and applying it to training data
tree = DecisionTreeRegressor(criterion='mse', max_depth=2)
tree.fit(xtrain, ytrain)

#Calculating predicted y values for xtrain values
y_pred = tree.predict(xtrain)
```

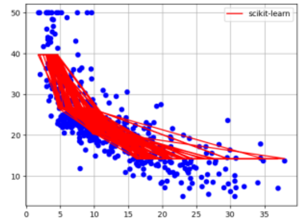
Question 3

Import the Boston Housing dataset “from sklearn import datasets” and “boston = datasets.load_boston()”. This data set contains 13 features and 1 target variable listed below. Use “LSTAT” as the input feature and “MEDV” as the target output. Fit a regression tree to depth 2 and compare your results with results generated by “from sklearn.tree import DecisionTreeRegressor” using the “mean square error” criterion.

```
#Using decision tree in sklearn
from sklearn.tree import DecisionTreeRegressor

#Creating an object called tree and applying it to training data
tree = DecisionTreeRegressor(criterion='mse', max_depth=2)
tree.fit(xtrain, ytrain)

#Calculating predicted y values for xtrain values
#y_pred = tree.predict(xtrain)
y_pred = tree.predict(xtrain)
print(y_pred)

plt.figure(1)
plt.plot(xtrain,ytrain,'bo')
plt.plot(xtrain,y_pred,'r.',label="scikit-learn")
plt.grid()
plt.show()
```



If you want to plot a line like above, use np.sort to sort xtrain…



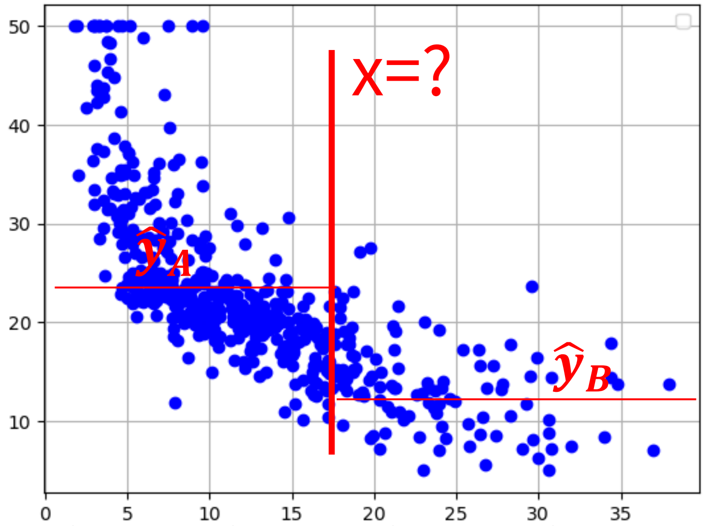
Question 3

Fit a regression tree to depth 2.

STEPS to implement BEST THRESHOLD ::

Find best x value such that MSE of node A and MSE of node B is minimum

- Given set of data x and y
- For x = 0 to max x value
  - Find mean y in node A ➔ calculate sum of squared errors in node A. (also can do mse)
  - Find mean y in node B ➔ calculate sum of squared errors in node B. (also can do mse)
  - Sum the two errors above and store the value
- Identify x value with minimum sum of errors and select that as threshold at current depth



Algorithm: Regression Tree Learning

Input: parameter *max_depth* & training set
Output: Tree

root ← all training samples
for $d \leftarrow 1$ ***to*** *max_depth* **do**
  for *each leaf node m at depth $d-1$* **do**
    Find best feature & best threshold, so splitting node m into two reduces MSE the most
    Use decision rule to distribute training samples from node m across two new leaf nodes
return *tree*

Question 3

Fit a regression tree to depth 2.

STEPS to implement BEST THRESHOLD ::

Find best x value such that MSE of node A and MSE of node B is minimum

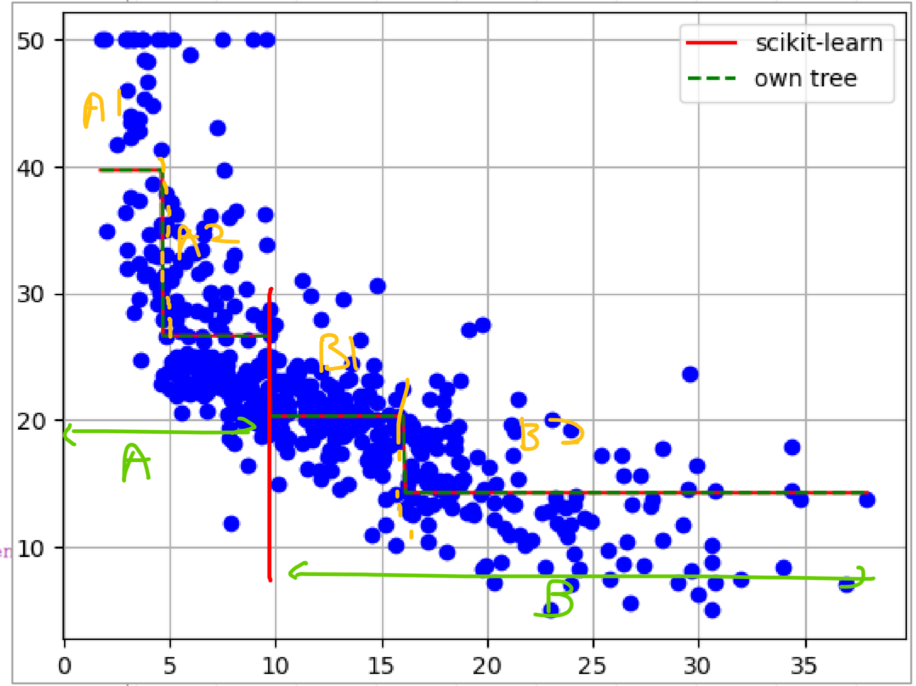
- Given set of data x and y
- For x = 0 to max x value
  - Find mean y in node A ➔ calculate sum of squared errors in node A. (also can do mse)
  - Find mean y in node B ➔ calculate sum of squared errors in node B. (also can do mse)
  - Sum the two errors above and store the value
- Identify x value with minimum sum of errors and select that as threshold at current depth

```python
# This function finds the best threshold in given dataset y according to minimum sum of
def find_best_split(y):

    # index represents last element in the below threshold node
    sq_err_vec = np.zeros(len(y)-1)
    for index in range(0, len(y)-1):

        # split the data
        data_below_threshold = y[:index+1]   # ==> from 0 to index+1
        data_above_threshold = y[index+1:]   # ==> index+1 to the end of array

        # Compute estimate
        mean_below_threshold = np.mean(data_below_threshold)
        mean_above_threshold = np.mean(data_above_threshold)

        # Compute total square error
        # Note that MSE = total square error divided by number of data points
        below_sq_err = np.sum(np.square(data_below_threshold - mean_below_threshold))
        above_sq_err = np.sum(np.square(data_above_threshold - mean_above_threshold))
        sq_err_vec[index] = below_sq_err + above_sq_err

    #np.argmin returns the index of the minimum value in this array
    best_index = np.argmin(sq_err_vec)

    yL = y[:best_index+1]    #y values in node A / left side
    yR = y[best_index+1:]    #y values in node B  / right side

    return yL, yR
```

node A

node B

```
#sorting data according to x values
#argsort is a numpy function that returns the indices that would sort this array xtrain
xtrain = boston.data[:,12]
sorted_index = xtrain.argsort()
xtrain = xtrain[sorted_index]
print(xtrain)

#Remember to sort y accordingly too
ytrain=boston.target
ytrain = ytrain[sorted_index]


#Splitting data at first level (depth = 1 )
yA, yB = find_best_split(ytrain)

#Splitting data at second level (depth = 2)
yA1, yA2 = find_best_split(yA)
yB1, yB2 = find_best_split(yB)

#Calculating the regression tree y values (average of y values in this node)
yA1_pred = np.mean(yA1)
yA2_pred = np.mean(yA2)
yB1_pred = np.mean(yB1)
yB2_pred = np.mean(yB2)

#generating the y values for plotting
y_pred_plotting = np.concatenate([yA1_pred*np.ones(len(yA1)),yA2_pred*np.ones(len

plt.plot(xtrain,y_pred_plotting,'g', linestyle='dashed',label="own tree")
plt.legend()
plt.grid()
plt.show()
```



Question 4

Get the data set "from sklearn.datasets import load_iris". Perform the following tasks.

(a) Split the database into two sets: 80% of samples for training, and 20% of samples for testing using random_state=0.

(b) Train a decision tree classifier (i.e., "tree.DecisionTreeClassifier" from sklearn) using the training set with a maximum depth of 4 based on the "entropy" criterion.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0)* ¶

Methods

apply(X[, check_input])	Return the index of the leaf that each sample is predicted
cost_complexity_pruning_path(X, y[, …])	Compute the pruning path during Minimal Cost-Complex
decision_path(X[, check_input])	Return the decision path in the tree.
fit(X, y[, sample_weight, check_input, …])	Build a decision tree classifier from the training set (X, y).
get_depth()	Return the depth of the decision tree.
get_n_leaves()	Return the number of leaves of the decision tree.
get_params([deep])	Get parameters for this estimator.
predict(X[, check_input])	Predict class or regression value for X.
predict_log_proba(X)	Predict class log-probabilities of the input samples X.
predict_proba(X[, check_input])	Predict class probabilities of the input samples X.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.

```
## (a) split data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
iris_dataset['data'], iris_dataset['target'], test_size=0.2, random_state=0)
```

```
#Using decision tree in sklearn
from sklearn.tree import DecisionTreeClassifier

#Creating an object called tree and applying it to training data
tree = DecisionTreeClassifier(criterion='entropy', max_depth=4)
tree.fit(X_train, y_train)

#Calculating predicted y values for xtrain values
y_pred = tree.predict(X_train)
```

Question 4

(c) Compute the training and test accuracies. You can use accuracy_score from sklearn.metrics for accuracy computation

sklearn.metrics. **accuracy_score**(*y_true*, *y_pred*, ***, *normalize=True*, *sample_weight=None*) ¶ [source]

Accuracy classification score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true.

```
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```

```
#Calculating predicted y values for xtrain values
y_pred_train = tree.predict(X_train)
y_pred_test =  tree.predict(X_test)

score_train = accuracy_score(y_train, y_pred_train)
score_test = accuracy_score(y_test, y_pred_test)
```

```
Training accuracy score : 0.9916666666666667
Testing accuracy score : 1.0
```

Question 4

(d) Plot the tree using “tree.plot_tree”.

sklearn.tree.plot_tree

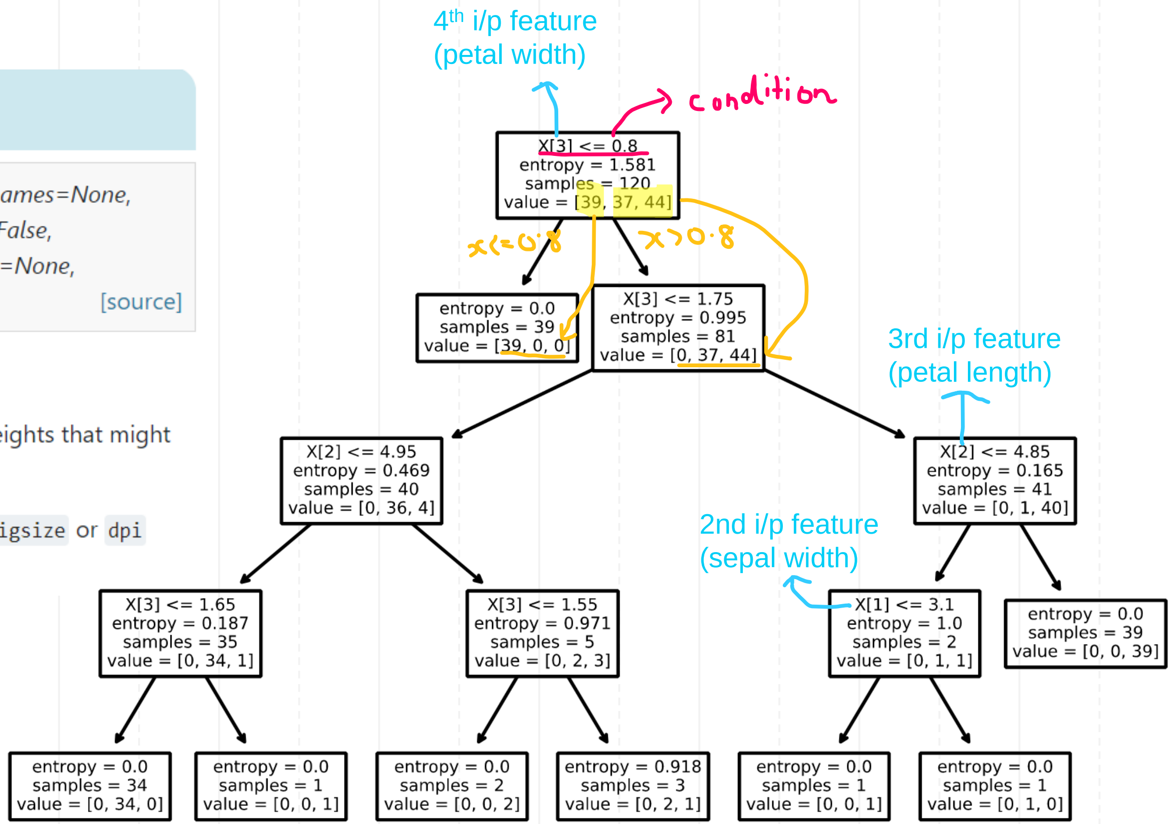
`sklearn.tree.plot_tree`(*decision_tree, *, max_depth=None, feature_names=None, class_names=None, label='all', filled=False, impurity=True, node_ids=False, proportion=False, rotate='deprecated', rounded=False, precision=3, ax=None, fontsize=None*) ¶ [source]

Plot a decision tree.

The sample counts that are shown are weighted with any sample_weights that might be present.

The visualization is fit automatically to the size of the axis. Use the `figsize` or `dpi` arguments of `plt.figure` to control the size of the rendering.

```
from sklearn.tree import plot_tree

plot_tree(tree)
plt.show()
```



https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html#sphx-glr-auto-examples-tree-plot-iris-dtc-py

4th i/p feature
(petal width)

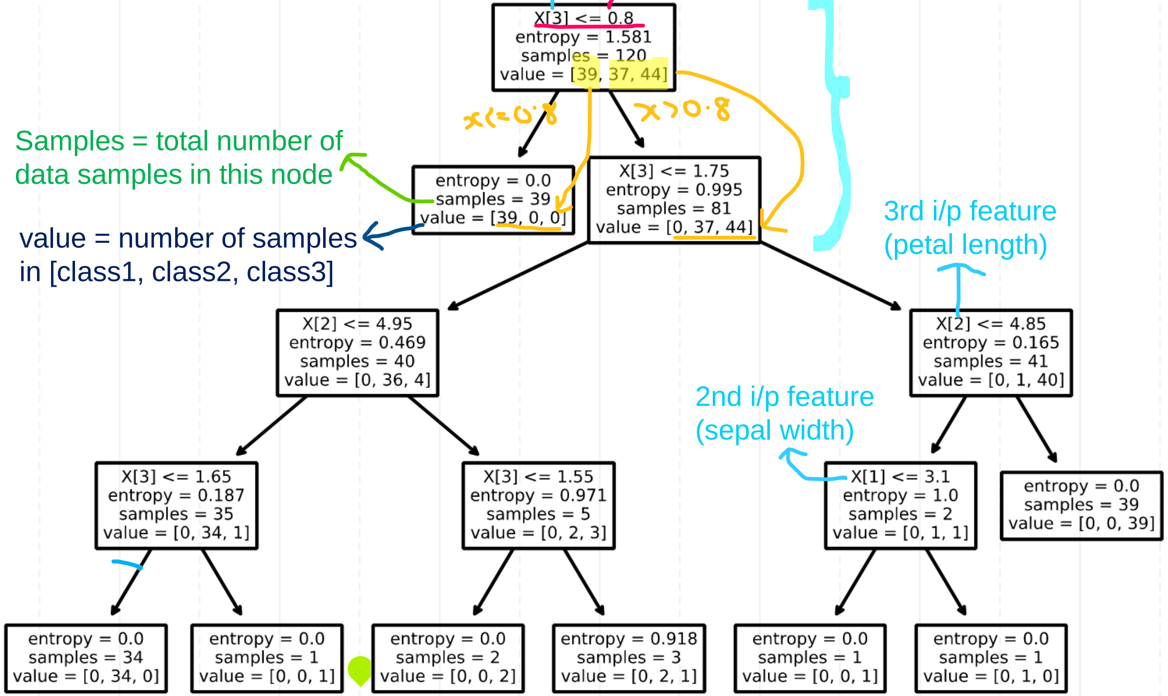
condition

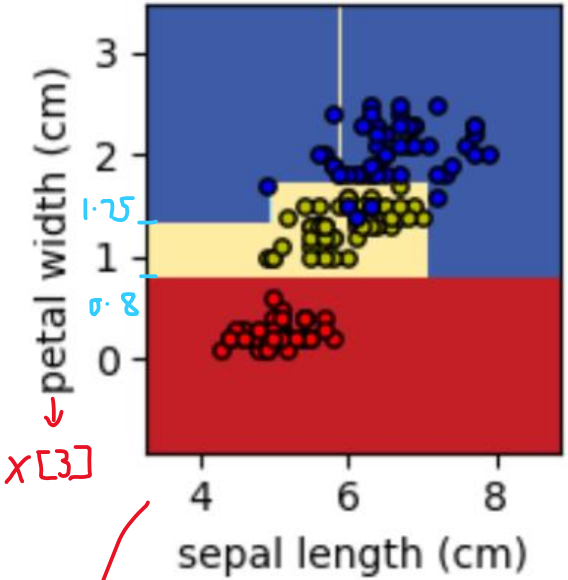
Samples = total number of data samples in this node

value = number of samples in [class1, class2, class3]

3rd i/p feature
(petal length)

2nd i/p feature
(sepal width)





This plot is taken from this link!

https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html#sphx-glr-auto-examples-tree-plot-iris-dtc-py