Artificial Intelligence

Decision trees,
for classification

Decision Tree (DT) ?

- Supervised learning algorithm
  - For classification
  - And for regression
- It can be represented as a logical tree, with a root node, decision nodes, leaf nodes, connected by "decisions"
- A DT models decisions and their consequences
- For both tasks, scikit-learn uses the "CART" algorithm,
  - CART = "ClAssification and Regression Trees"
    - The one supported in scikit-learn
  - It produces "binary trees"
    - Each node has exactly 2 children
  - Classification with
    - Gini-index
    - Entropy
  - Regression with
    - MSE (Mean Squared Error)

Decision Tree (DT) - Concept

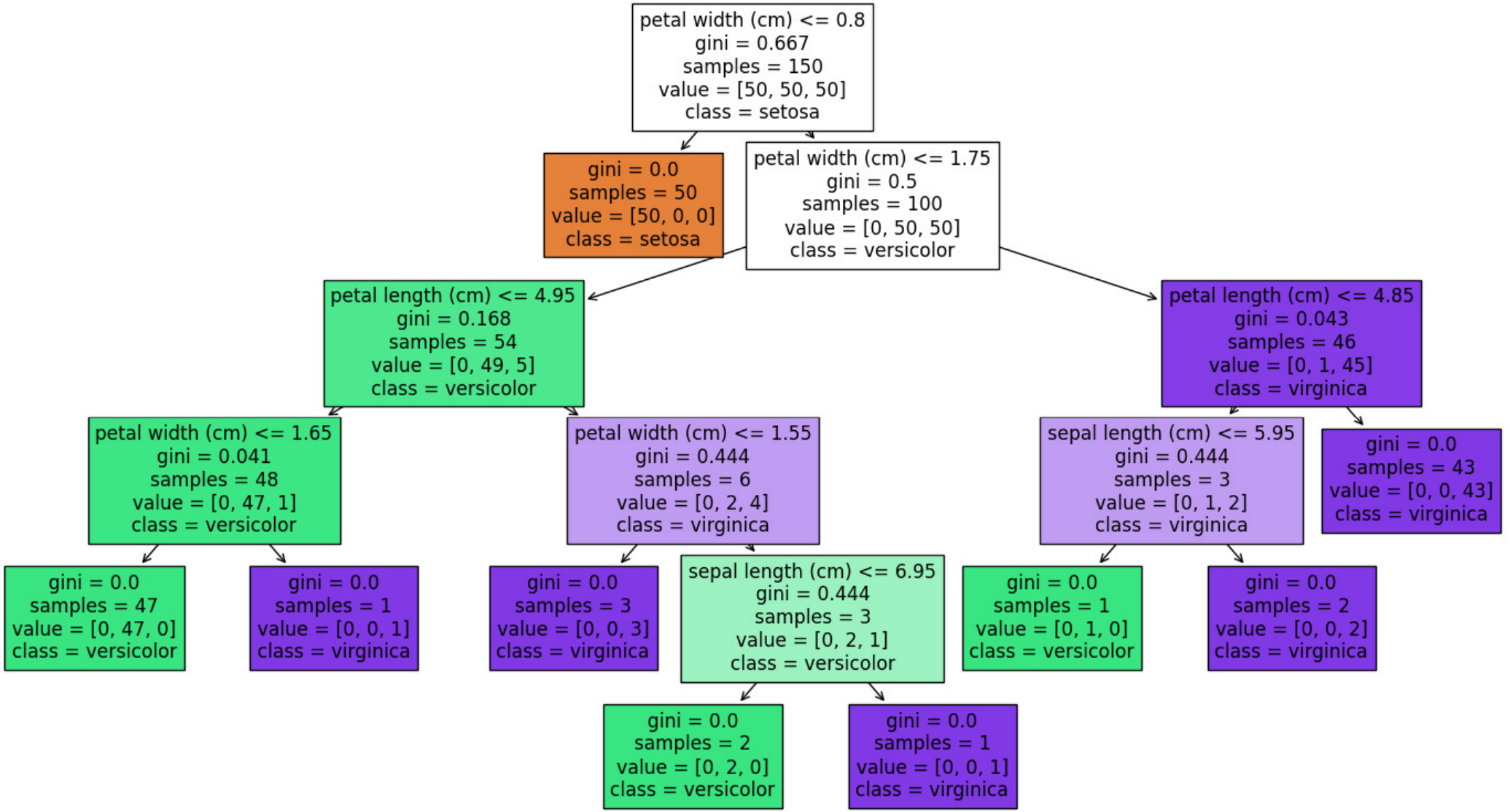
- Each node should represent a binary split of the samples being classified
  - By what criterion/attribute/feature?
    - The one that "best" separates the data into [2 different] classes
    - Gini-index is a metric to find that criterion
      - Also known as "Gini impurity"
      - Spoiler: pick the feature with the lowest Gini-index
- Start by considering the entire labelled dataset
  - Compute the "best" split criterion
  - When applied, that criterion splits the tree in 2 different branches
    - Conducting to 2 different sub-trees
      - representing sub-sets of the original dataset
- Repeat for each sub-set
  - So, it is a recursive approach
  - Until a stop condition is matched

Decision Tree (DT) - When to stop?

- The CART hyper-parameters (Leo Breiman, 1984)
  - max_depth (lower values, simpler, less prone to overfit)
  - min_samples_split (default 2)
  - min_samples_leaf (default 1)
- Limit by metric
  - Gini of 0 (zero) means "no impurity"
    - All data belongs to a single class
- Limit the tree's depth
  - Control the recursive exploration
- Limit by number of samples in [leaf] nodes
- Take a look at some trees, as plotted by:
  - 1_dt_classification_iris.py
  - 1_dt_classification_wine.py
  - 1_dt_regression_boston.py # fail - ethics
  - 1_dt_regression_california.py

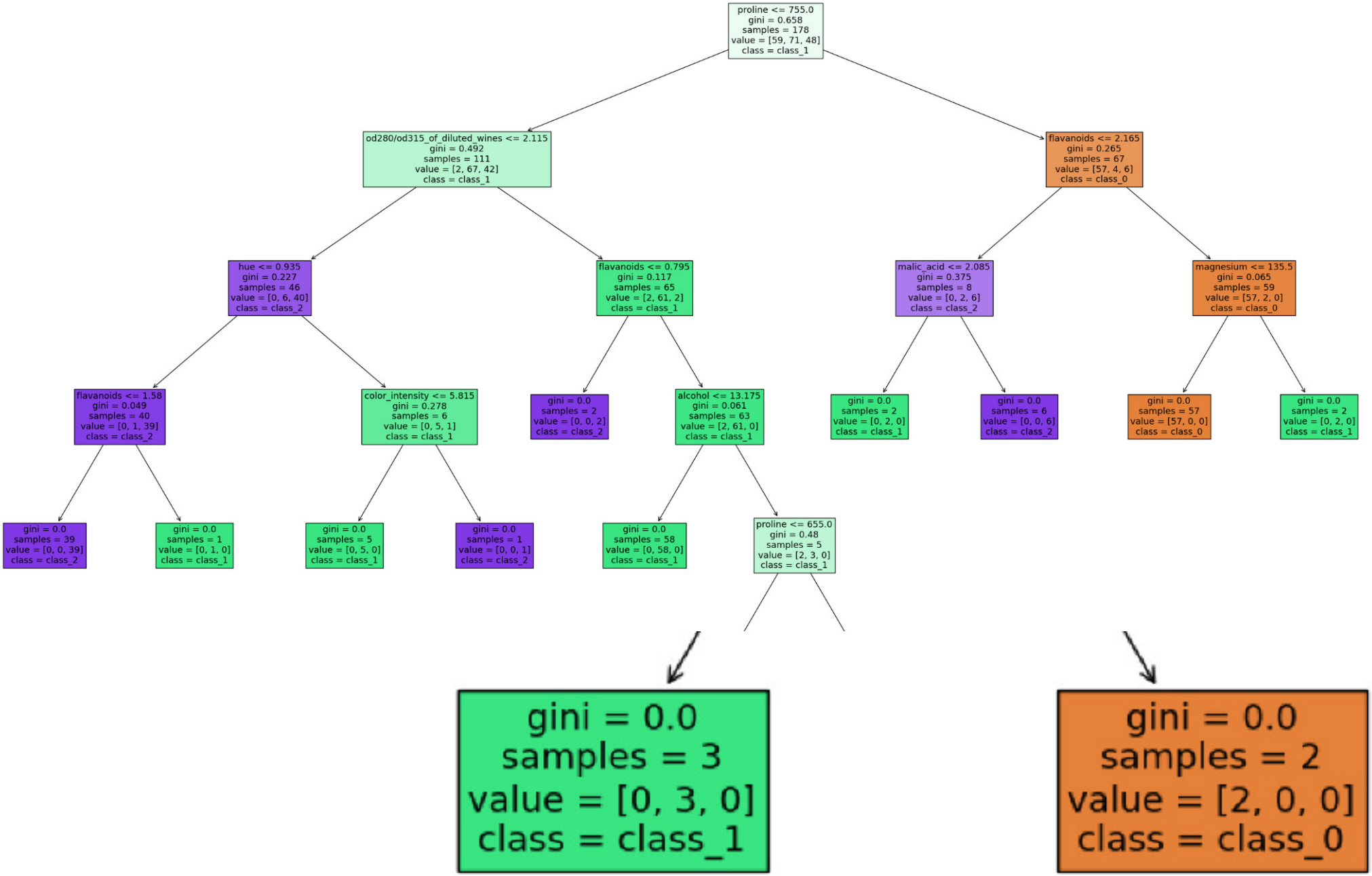
DT for classification using the "Iris" DS

- Each node: criterion ; gini ; samples in it ; samples distribution ; majority class

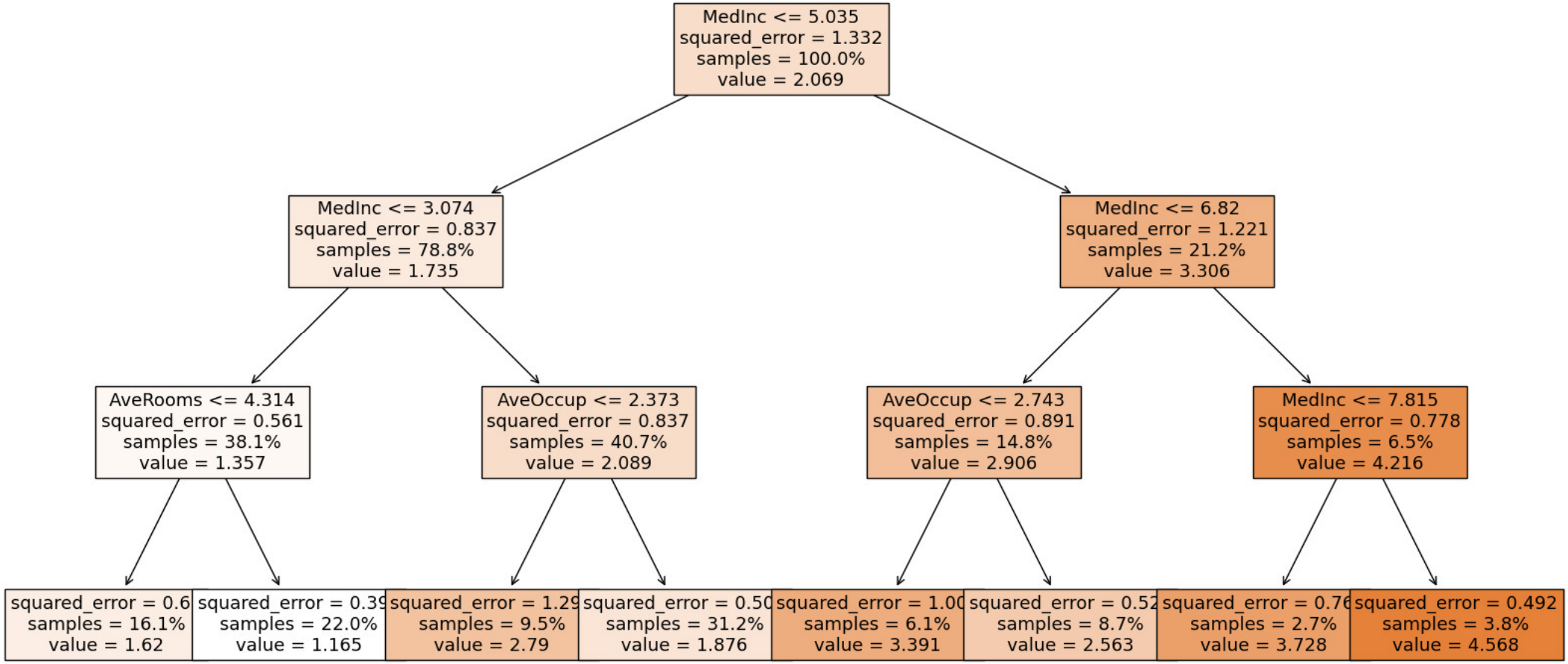


DT for classification using the "Wine" DS

- Notice the gini = 0 on leaf nodes



DT for regression using the "California housing" DS



$$G = 1 - \sum_{i=1}^{c} (p_i)^2$$

Computing the Gini-index

- Gini-index or Gini impurity (or just G in this doc)
- Computed per feature and in parts
  - G = 1 - sum(proportion of class i “in the node”**2)
  - One part per feature value
  - For example, if the possible values for a feature F are
    - Yes or No, then compute
    - G(Yes)
    - G(No)
    - G = weight(Yes) * G(Yes) + weight(No) * G(No)
      - Weight(Yes) = number of samples with F=Yes / total number of samples
      - Weight(No) = number of samples with F=No / total number of samples

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Computing Gini algorithm by example 1/5

- Consider the supervised dataset above
  - The target column is named “preferred”
- Start with any feature,
  - for example, with feature F = “Account age > 1”
- Step #1 - build as many sub-sets as necessary to hold the different values for F
  - Each set will contain the samples with a specific F value
- 2 sets, in this case, one set for “No”, other for “Yes”
- set_no = [2, 6, 7] ; set_yes = [0, 1, 3, 4, 5]

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Computing Gini algorithm by example 2/5

- set_no = [2, 6, 7] ; set_yes = [0, 1, 3, 4, 5]
- Step #2 - build the same sub-sets, but representing the dataset's different values, for the target column
  - group_no = [0, 1, 2, 4, 7] ; group_yes = [3, 5, 6]
- Next, find the intersections between the sets formed for the feature and for the target/labels
  - set_no.intersection(group_no) = [2, 7]
  - set_no. intersection(group_yes) = [6]
  - *set_yes*. Intersection(group_no) = [0, 1, 3]
  - *set_yes*. Intersection(group_yes) = [3, 5]

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Computing Gini algorithm by example 3/5

- Remember
  - set_no = [2, 6, 7] ; set_yes = [0, 1, 3, 4, 5]
  - set_no.intercept[*group_no*] = [2, 7] ; set_no.intercept[group_yes] = [6]
  - set_yes.intercept[*group_no*] = [0, 1, 4] ; set_yes.intercept[group_yes] = [3, 5]
- Step #3 – Compute the relation between the #samples in the intersection groups and the #samples in the sets with a specific feature value (set_<value>)
- set_no has 3 elements
  - 2 of set_no are in the intersection with *group_no* (1)
  - 1 of set_no is in the intersection with group_yes (2)
  - Pno1 = 2/3 ; Pno2 = 1/3
- set_yes has 5 elements
  - 3 of set_yes are in the intersection with *group_no* (1)
  - 2 of set_yes are in the intersection with group_yes (2)
  - Pyes1 = 3/5 ; Pyes2 = 2/5

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Computing Gini algorithm by example 4/5

- Remember
  - set_no = [2, 6, 7] ; set_yes = [0, 1, 3, 4, 5]
  - set_no.intercept[*group_no*] = [2, 7] ; set_no.intercept[group_yes] = [6]
  - set_yes.intercept[*group_no*] = [0, 1, 4] ; set_yes.intercept[group_yes] = [3, 5]
  - Pno1 = 2/3 ; Pno2 = 1/3 ; Pyes1 = 3/5 ; Pyes2 = 2/5
- Step #4 – Compute the partials
  - G(No) = 1 - (2/3**2 + 1/3**2) = 0.44
  - G(Yes) = 1 – (3/5**2 + 2/5**2) = 0.48
- Next, compute the weights for the partials
  - set_no has 3 elements ; set_yes has 5 elements
  - The dataset has 8 elements
  - Weight(No) = 3/8
  - Weight(Yes) = 5/8

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Computing Gini algorithm by example 5/5

- Remember
  - set_no = [2, 6, 7] ; set_yes = [0, 1, 3, 4, 5]
  - set_no.intercept[*group_no*] = [2, 7] ; set_no.intercept[group_yes] = [6]
  - set_yes.intercept[*group_no*] = [0, 1, 4] ; set_yes.intercept[group_yes] = [3, 5]
  - Pno1 = 2/3 ; Pno2 = 1/3 ; Pyes1 = 3/5 ; Pyes2 = 2/5
  - G(No) = 0.44 ; G(Yes) = 0.48 ; W(No) = 3/8 ; W(Yes) = 5/8
- Step #5 – Compute the Gini-index for the feature
  - G = W(No)*G(No) + W(Yes)*G(Yes)
  - G = 3/8 * 0.44 + 5/8 * 0.48 = 0.465 ~ 0.47
- Interpretation?
  - Do this for ALL features
  - The lowest G value represents the feature to be used for splitting the dataset in the decision tree

$$G = 1 - \sum_{i=1}^{c} (p_i)^2$$

Using the Gini-index

- Exercise: compute the Gini-index for the dataset, for the other features "Violations > 0" and "Balance > 50.000"
- Help:

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Feature	F-V	P1	P2	P1**2	P**2	sum	G	
Account Age > 1	No	2/3	1/3	4/9	1/9	0.56	0.44	GNo
Account Age > 1	Yes	3/5	2/5	1/3	1/6	0.52	0.48	GYes
Violations > 0	No	1/2	1/2	1/4	1/4	0.50	0.50	GNo
Violations > 0	Yes	1	0	1	0	1.00	0.00	GYes
Balance > 50.000	No	1	0	1	0	1.00	0.00	GNo
Balance > 50.000	Yes	4/7	3/7	1/3	1/5	0.51	0.49	GYes

$$G = 1 - \sum_{i=1}^{c} (p_i)^2$$

Using the Gini-index

- G(Violations>0) = 6/8 * 0.5 + 2/8 * 0 = 0.375
- G(Balance > 50.000) = 1/8*0 + 7/8 * 0.49 = 0.42875
- Help:

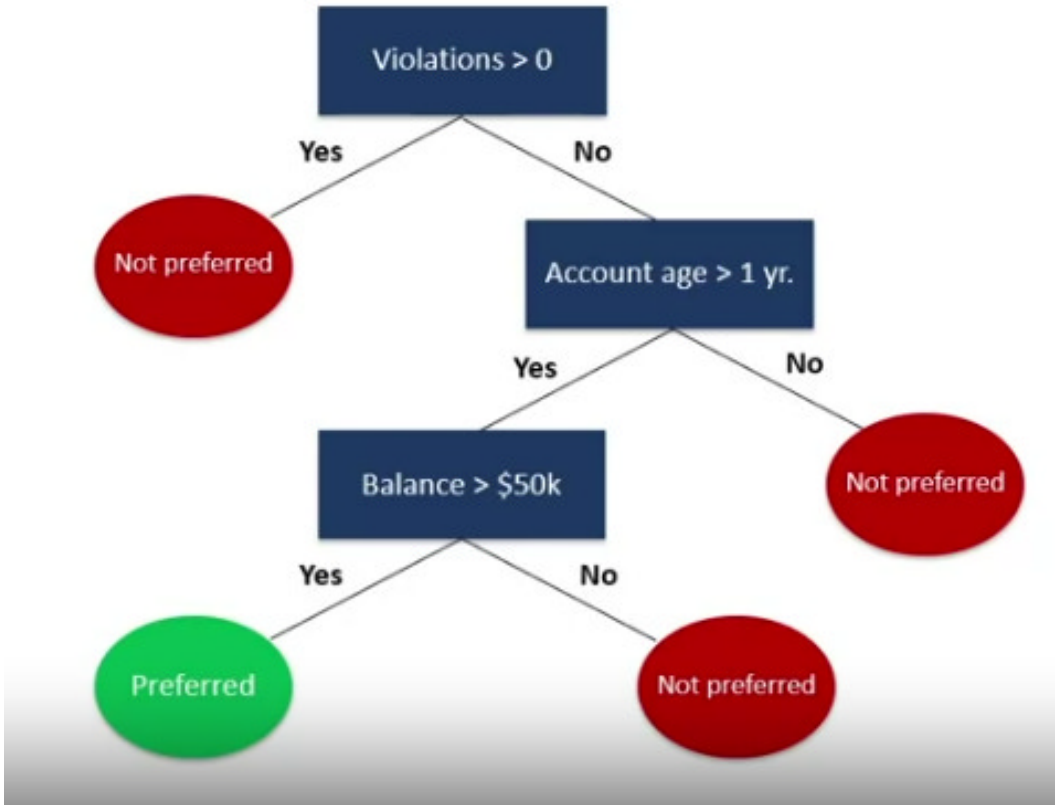
Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Feature	F-V	P1	P2	P1**2	P**2	sum	G	
Account Age > 1	No	2/3	1/3	4/9	1/9	0.56	0.44	GNo
Account Age > 1	Yes	3/5	2/5	1/3	1/6	0.52	0.48	GYes
Violations > 0	No	1/2	1/2	1/4	1/4	0.50	0.50	GNo
Violations > 0	Yes	1	0	1	0	1.00	0.00	GYes
Balance > 50.000	No	1	0	1	0	1.00	0.00	GNo
Balance > 50.000	Yes	4/7	3/7	1/3	1/5	0.51	0.49	GYes

$$G = 1 - \sum_{i=1}^{c} (p_i)^2$$

Using the Gini-index

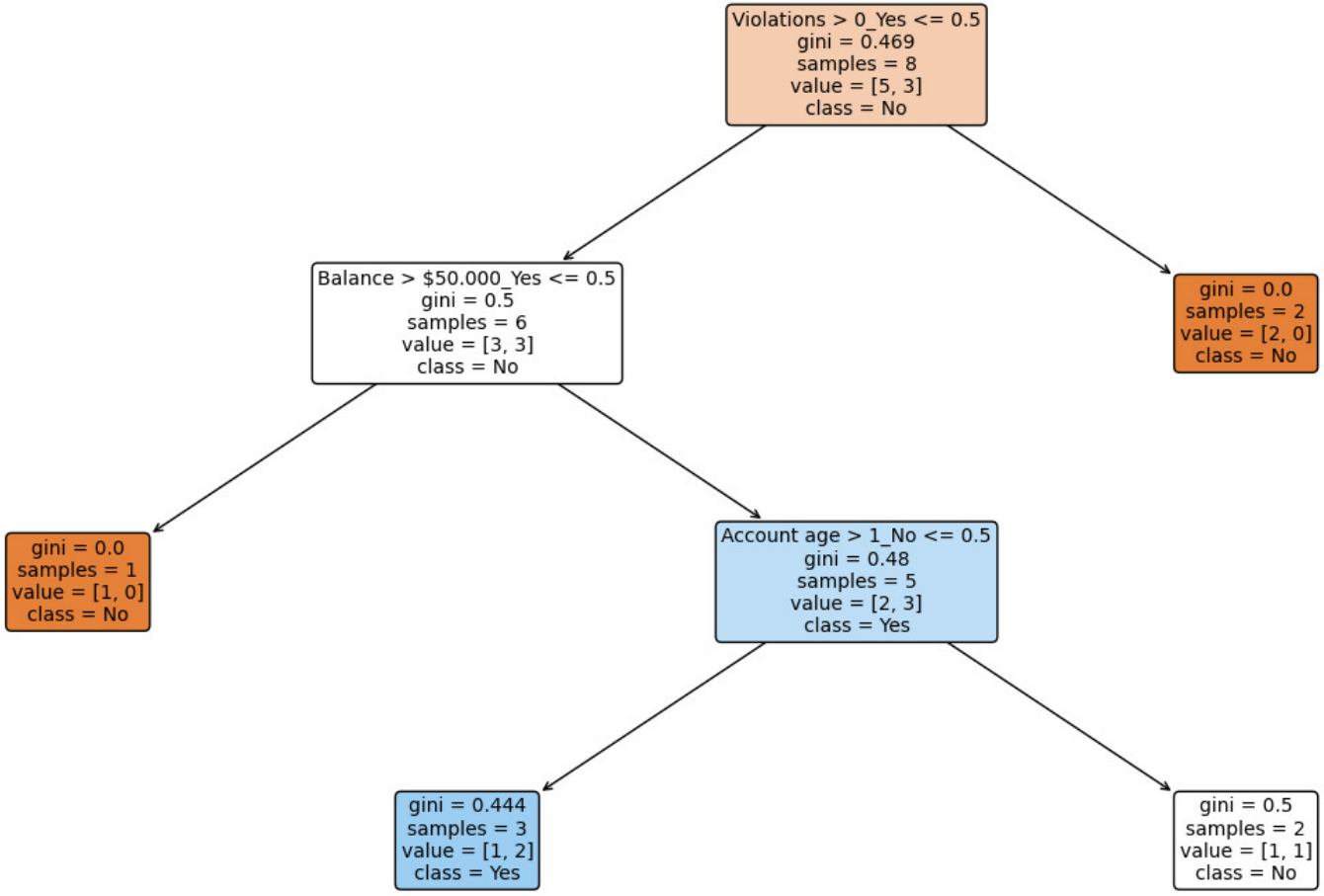
- G(Account Age>1) = 0.465 ~ 0.47
- G(Violations>0) = 0.375 ~ 0.38 lowest impurity
- G(Balance > 50.000) = 0.42875 ~ 0.43
- Continue, recursively



$$G = 1 - \sum_{i=1}^{c} (p_i)^2$$

Using the Gini-index

- But scikit-learn's tree shows gini = 0.469, why?
- Because it is the value computed before considering any split, considering only the labels. It is the "Gini impurity" of the data points at that node
- G = 1 - (5/8 No ** 2 + 3/8 Yes ** 2) = 0.46875 ~ 0.469



References

- https://scikit-learn.org/stable/modules/tree.html
- https://github.com/amsm/am_decision_trees