**Decision Trees**

* Decision Trees are very versatile ML algorithm than can performs:
  + Classification Task
  + Regression Task
  + Multi-output Task
  + Also Used in Random Forest

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

iris = load\_iris()

iris.keys()

X = iris [:,2:]***# considering only the petal length and petal width as the features***

y = iris.target

tree\_clf = DecisionTreeClassifier(max\_depth=2)

tree\_clf.fit(X,y)

from sklearn.tree import export\_graphviz

export\_graphviz(

tree\_clf,

out\_file = ‘iris\_tree.dot’,

feature\_name = iris.feature\_names[2:]

class\_name = iris.target\_names,

rounded = True,

filled = True

)

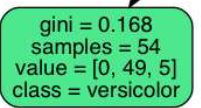
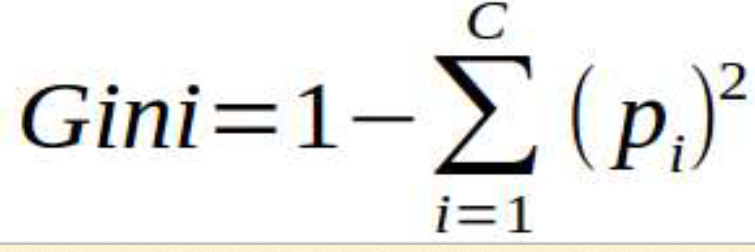
%ls **#Check the file in current directory**

!dot -Tpng iris\_tree.dot -o iris\_tree.png **#Convert file to png/pdf format**

**from** **IPython.display** **import** Image

Image(filename='iris\_tree.png')

* **Key Points from Tree :**
  + **Value Attributes** will tell how many training instances of each class being applied to the current node.
  + **Root Node** will divide based on most important feature.
  + **Gini Attribute** will measure the impurities. If **gini = 0**, then all the training instances applies to current node belongs to the same class.

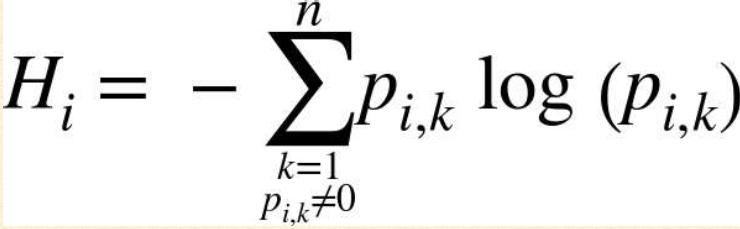
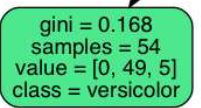
 **1- (0/54)2-(49/54)2-(5/54)2 = 0.168**

* Decision tree is a white box model, where decision is easy to interpret.
* For estimating the classification, it finds the probability of each class & return the class having maximum probability.

tree\_clf.predict\_proba([[5, 1.5]])

# CART Algorithm

* Scikit learn uses **Classification and Regression Tree (CART)** algorithm to train Decision Tree. The Algorithm splits the training sets into two subsets using a single feature ‘k’ and a threshold ‘tk’.
* It keeps on splitting the subsets until it reaches maximum depth(defined by the max\_depth Hypermeter) or it cannot find a split that will reduce the impurity.
* Complexity of prediction = O(log2(m)) . Hence complexity of prediction is independent from no of features, prediction is very fast.
* Complexity of Training = O(n\*m(log m)), n= no of features
* For smaller training sets (few thousand), Scikit learn can speed up by pre-sorting the data(set presort= True). But it is very slow for large Dataset
* **Measurement types :**
  + Gini
  + **Entropy : Degree of randomness.**

   = 0.31

* Most of the time both lead to similar trees. **Gini impurity is faster to compute**. However, Gini impurity isolate the most frequent class in its own branch of tree whereas **entropy trends to produce slightly more balanced tree**.
* To avoid overfitting, restriction is required over decision tree freedom , which is called as Regularization. Normally in case of Decision Tree, restriction is done on maximum depth.
* Model like Decision Tree are often called Non-Parametric Model, because the number of parameter is not defined prior to the model, so model structure is free to stick closely to the data. In contrast, Parametric model like linear model has a predetermined number of parameters, so its degree of freedom is limited, which reduce the risk of overfitting but increase the risk of underfitting.
* ***Regularization Parameter of DecisionTreeClassifier Class :***
* **max\_depth:** restricts the maximum depth of the tree.
* **min\_samples\_split :**  minimum no. of samples a node must have before it can be split.
* **min\_samples\_split :** minimum no. of samples a leaf node must have.
* **min\_weight\_fraction\_leaf :**  same as min\_samples\_leaf but expressed as a fraction of the total number of weighted instances
* **max\_leaf\_nodes :** maximum number of leaf nodes
* **max\_features :** maximum no of features that are evaluated for splitting at each node.
* Increasing min\_\* or Reducing max\_\* hyperparameters will regularize the model.

# Regression with Decision Trees

Decision Tree are also capable of performing Regression Task. The main difference is that instead of predicting the class, it predicts a value. It tries to minimize the MSE instead of impurity. Without regularization, the model may overfit.

Predicted value of each region is the average target value of the instances in that region.

from sklearn.trees import DecisionTreeRegressor

tree\_reg1 = DecisionTreeRegressor(random\_state=42,max\_depth=2)

tree\_reg2 = DecisionTreeRegressor(random\_state=42,max\_depth=3)

tree\_reg1.fit(X,y)

tree\_reg2.fit(X,y)

**def** plot\_regression\_predictions(tree\_reg, X, y, axes=[0, 1, -0.2, 1], ylabel="$y$"):

x1 = np.linspace(axes[0], axes[1], 500).reshape(-1, 1)

y\_pred = tree\_reg.predict(x1)

plt.axis(axes)

plt.xlabel("$x\_1$", fontsize=18)

**if** ylabel:

plt.ylabel(ylabel, fontsize=18, rotation=0)

plt.plot(X, y, "b.")

plt.plot(x1, y\_pred, "r.-", linewidth=2, label=r"$\hat**{y}**$")

plt.figure(figsize=(11, 4))

plt.subplot(121)

plot\_regression\_predictions(tree\_reg1, X, y)

**for** split, style **in** ((0.1973, "k-"), (0.0917, "k--"), (0.7718, "k--")):

plt.plot([split, split], [-0.2, 1], style, linewidth=2)

plt.text(0.21, 0.65, "Depth=0", fontsize=15)

plt.text(0.01, 0.2, "Depth=1", fontsize=13)

plt.text(0.65, 0.8, "Depth=1", fontsize=13)

plt.legend(loc="upper center", fontsize=18)

plt.title("max\_depth=2", fontsize=14)

plt.subplot(122)

plot\_regression\_predictions(tree\_reg2, X, y, ylabel=**None**)

**for** split, style **in** ((0.1973, "k-"), (0.0917, "k--"), (0.7718, "k--")):

plt.plot([split, split], [-0.2, 1], style, linewidth=2)

**for** split **in** (0.0458, 0.1298, 0.2873, 0.9040):

plt.plot([split, split], [-0.2, 1], "k:", linewidth=1)

plt.text(0.3, 0.5, "Depth=2", fontsize=13)

plt.title("max\_depth=3", fontsize=14)

plt.show()

export\_graphviz(

tree\_reg2,

out\_file='tree\_reg2.dot',

rounded=**True**,

filled=**True**)

!dot -Tpng tree\_reg2.dot -o tree\_reg2.png

* **Demerits of Decision Trees** 
  + Decision Tress love orthogonal decision boundaries, which makes them sensitive to training set rotation. They are very sensitive to small variations in the training data.