Ensemble Learning: Decision Trees

AAA-Python Edition

Plan

- 1- Decision Trees
- 2- Impurity
- 3- Mechanisms of binary decision trees
- 4- Decision Tree classifier example
- 5- Decision Tree regressor example
- 6- Visualization

1- Decision Trees

Concept

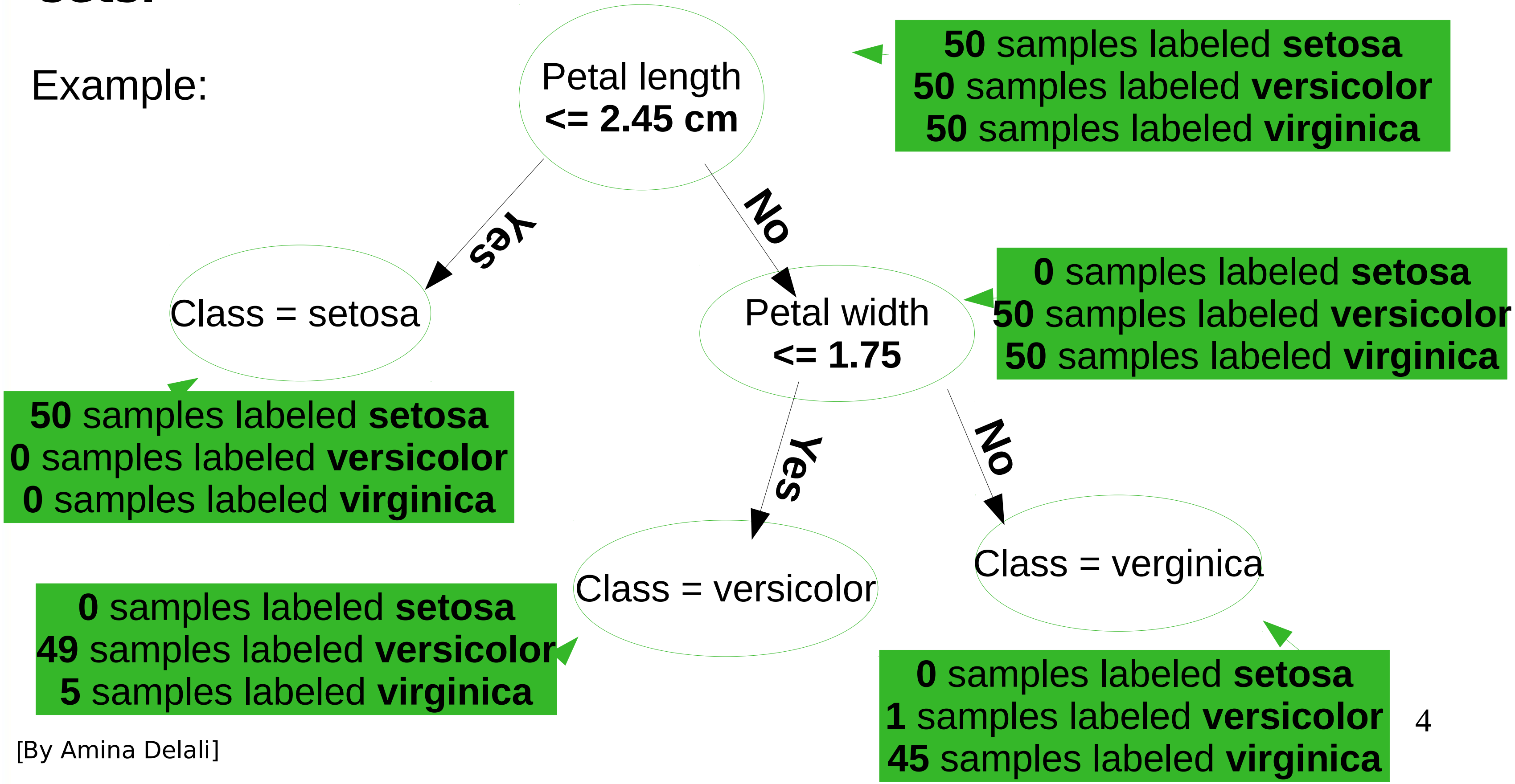
- **Decision Trees** are an other type of **learning models. Trained** on labeled data.
- They are used for both **classification** and **regression.**
- The model is trained, by **spliting** the dataset in order to reduce their **impurity** using a **tree structure.**
- The non-leaf nodes are the **decision nodes:** they perform a **test** or **apply a rule** on **feature values**. The result of the test will split the data into other sets ==> other nodes
- The leaf nodes: are **non splittable** nodes. They represent the final result of the classification or the regression.

[By Amina Delali]

Binary Decision Trees

- In a **binary decision trees:** each node will split the data into **2 sets.**

Example:



[By Amina Delali]

1- Decision Trees

Decision Trees in scikit-learn

- Scikit-learn implement an optimized version of the **Classification And Regression Tree (CART)** algorithm.

- It is a **Binary** tree that uses **one feature** to be tested regarding **one threshold** to split the data.

- The chosen feature and threshold are those which **minimize** the **impurity** value after each split <==> **information gain**

- The impurity caluclation formula, can be considered as the **cost function.**

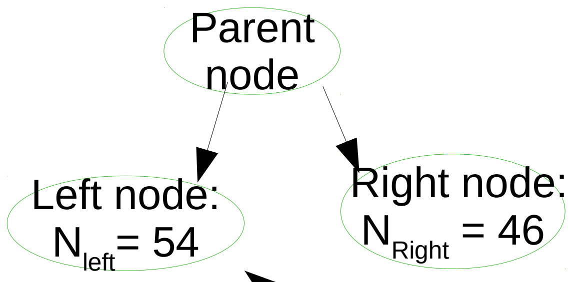
- For classification, scikit-learn implements the following impurity measures: **gini**, **entropy** and **miclassification** measures**.**

- For regression, it implements these measures: the **mean squared** and **mean absolute errors**.

[By Amina Delali]

Entropy

$$p_{mk} = 1/N_m * \sum_{x_i \in R_m} I(y_i = k)$$



- Entropy value for a node:

$$H(X_m) = -\sum_k p_{mk} \log_2(p_{mk})$$

From the Previous example

P_{0left}= 0/ (0+49+5)= 0/54 = 0
P_{1left}= 49 /(0+49+5)= 0.907
P_{2left} = 5 /(0+49+5) = 0.093

$H(X_{left})$= - 0 – 0.907*$\log_2$(0.907) – 0.093*$\log_2$(0.093)
= 0.446

- Entropy value for the partition

$$J = \frac{N_{left}}{(N_{left}+N_{righ})} * H(X_{left.node}) + \frac{N_{right}}{(N_{left}+N_{righ})} * H(X_{right.node})$$

J= 54/(54+46) * 0.446 + 46/(54+46)* $H(X_{right})$=
= 0.24 +0.46*$H(X_{right})$

[By Amina Delali]

With the convention: $0 \cdot \log_2(0) = 0$

2- Impurity

Gini

$$p_{mk} = 1/N_m * \sum_{x_i \in R_m} I(y_i = k)$$

- Gini measure for a node:

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}) = 1 - \sum_k p_{mk}^2$$

From the previous example

P_{0left}= 0/ (0+49+5)= 0/54 = 0
P_{1left}= 49 /(0+49+5)= 0.907
P_{2left} = 5 /(0+49+5) = 0.093

$H(X_{left})$= 1- 0^2 - $(0.907)^2$ - $(0.093)^2$ = 0.169

- Gini value for the partition

$$J = \frac{N_{left}}{(N_{left}+N_{righ})} * H(X_{left.node}) + \frac{N_{right}}{(N_{left}+N_{righ})} * H(X_{right.node})$$

J= 54/(54+46) * 0.169+ 46/(54+46)* $H(X_{right})$=
= 0.09 +0.46*$H(X_{right})$

2- Impurity

[By Amina Delali]

Mean and absolute Square Error

2- Impurity

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

- Mean squared error for a node:

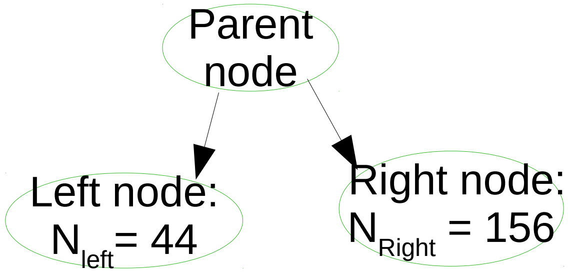
$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

- Mean absolute error for a node:

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y}_m|$$

- Error at the partition

$$J = \frac{N_{left}}{(N_{left}+N_{righ})} * H(X_{left.node}) + \frac{N_{right}}{(N_{left}+N_{righ})} * H(X_{right.node})$$



We suppose that:

$$\sum_{i=1}^{44} y_i = 0.69$$

So: $\bar{y}_{\text{left}}$ = (1/44)*0.69=0.157

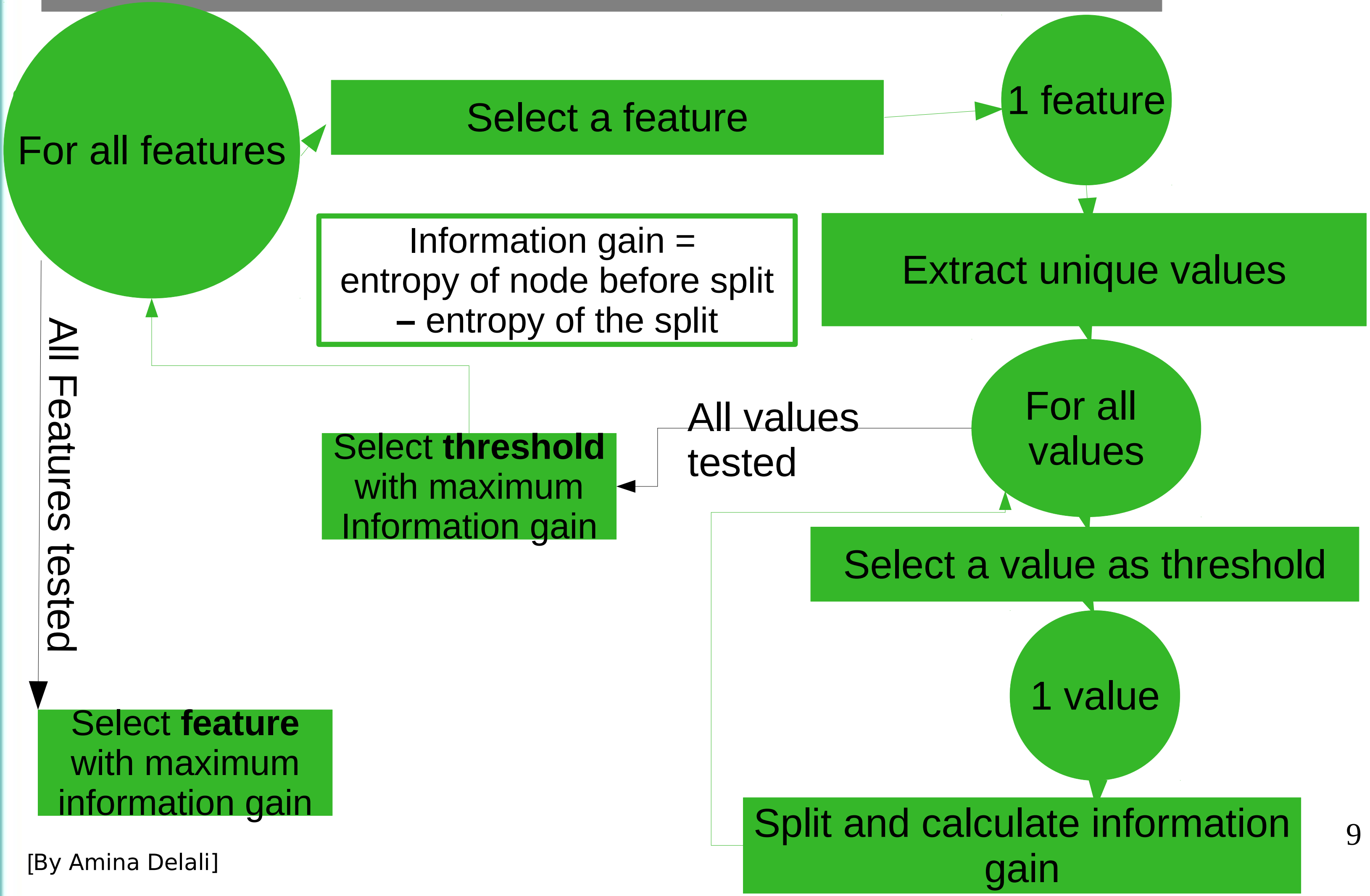
And : $H(X_{left})$= 1/44*

$$\sum_{i=1}^{44} (y_i - 0.157)^2$$

[By Amina Delali]

3- Mechanisms of binary decision trees

Thresholds and features selection: using entropy



[By Amina Delali]

Training

- After the selection of the (feature, threshold) pair, the root node is split into **2** other nodes using that pair.

- The process is repeated for the new nodes until:

  - Max depth of the tree is reached (the number of levels) or

  - Min samples by node is reached or

  - Other parameters or

  - No other split that reduces impurity is possible.

- A value is attributed to each final node (the leaf):
  - For a classification: it is a class number that represents the most represented class in that node.
  - For a regression: it is the average values of the y labels corresponding to all the samples belonging to that node.

[By Amina Delali]

Predicting

- You start from the root: you compare the values of the features selected in that node with the threshold value of the noce.
- The result : if it is True you go to the left, else you go to the right node.
- You repeat this process, until you rich a leaf.

- Classification
  - If you are doing classification, the predicted class for your sample, will be the class corresponding to that leaf.
  - A probability of belonging to that class can be computed: count of samples belonging to that class (in that node) / count of all samples (in that node)
- Regression
  - If your are doing a regression, the predicted value will be the value related to that leaf.

4-Decision Tree classifier example

Training

```
#import iris plant datasets tools
from sklearn.datasets import load_iris

# import the decision tree classifier
from sklearn.tree import DecisionTreeClassifier

# import numpy
import numpy as np

#import train_test_split from model_selection
from sklearn.model_selection import train_test_split

myIris = load_iris()
X = myIris.data
y = myIris.target
x_train,x_test,y_train,y_test= train_test_split(X, y, test_size=0.25)
myModel = DecisionTreeClassifier(max_depth=5)
myModel.fit(X, y)
```

Iris plant database:

- 150 samples, equally distributed in 3 classes:
  - Iris-Setosa,
  - Iris-Versicolor,
  - Iris-Virginica
- Each sample is described by 4 features:
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width

- max_depth: the root node’s depth ==0 (level 0).
- The first split’s depth ==1 (level 1), and so on.
- So, max_depth==5 means that we can not generate a tree with more than 5 levels

Other default parameters:

- criterion='gini'
- min_samples_leaf=1
- min_samples_split=2

[By Amina Delali]

Predicting

4-Decision Tree classifier example

```
y_pred = myModel.predict(x_test)
y_pred
```

```
array([0, 1, 2, 2, 2, 2, 0, 1, 1, 2, 1, 1, 0, 1, 0, 1, 0, 1, 2, 2, 2, 0,
       1, 0, 2, 1, 1, 0, 2, 0, 1, 1, 0, 1, 0, 0, 0, 1])
```

```
print(x_test[0])
```

```
[5.  3.5 1.6 0.6]
```

```
y_pred_prob= myModel.predict_proba(x_test)
y_pred_prob
```

```
array([[1.        , 0.        , 0.        ],
       [0.        , 0.96969697, 0.03030303],
       [0.        , 0.        , 1.        ],
       [0.        , 0.        , 1.        ],
       [0.        , 0.        , 1.        ],
       [0.        , 0.25      , 0.75      ],
```

```
print(myModel.decision_path(x_test))
```

```
  (0, 0)	1
  (0, 1)	1
  (1, 0)	1
  (1, 2)	1
  (1, 3)	1
  (1, 4)	1
  (2, 0)	1
  (2, 2)	1
  (2, 6)	1
```

- The sample **0** went to node **0**, then to node **1.**
- The node contained only “setosa” iris plant classes.

[By Amina Delali]

4-Decision Tree classifier example

Analyze the results

```python
# import the confusion matrix
from sklearn.metrics import confusion_matrix
myConfMat = confusion_matrix(y_test,y_pred)
print( myConfMat)
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```

```python
# import the classfication report
from sklearn.metrics import classification_report

myClassReport = classification_report (y_test,y_pred,target_names = myIris.target_names)
print(myClassReport)
```

```
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
 False  True  True  True  True
  True  True]
```

```
             precision    recall  f1-score   support

     setosa       1.00      1.00      1.00        13
 versicolor       1.00      0.94      0.97        16
  virginica       0.90      1.00      0.95         9

avg / total       0.98      0.97      0.97        38
```

```
[0.   0.25 0.75]
```

Even if all the 9 samples of virginca classes were detected, the precision was 0.9, and this is because of predicting wrongly one versicolor samples being a virginica iris plant

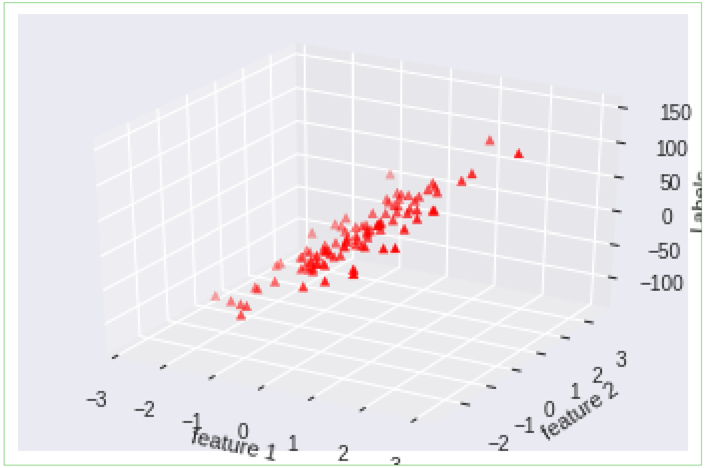
[By Amina Delali]

5- Decision Tree Regressor example

Training

```
from sklearn.datasets import make_regression
from mpl_toolkits.mplot3d import Axes3D

%matplotlib inline

# generate a random regression problem
xr,yr= make_regression( n_features=2)


fig=plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(xr[:,0], xr[:,1], yr,marker="^",c="r")
ax.set_xlabel('feature 1')
ax.set_ylabel('feature 2')
ax.set_zlabel("Labels")

plt.show()
```

Ploting in 3D (2 features,and y labels)



Generate random points that can be modeled by a regressor

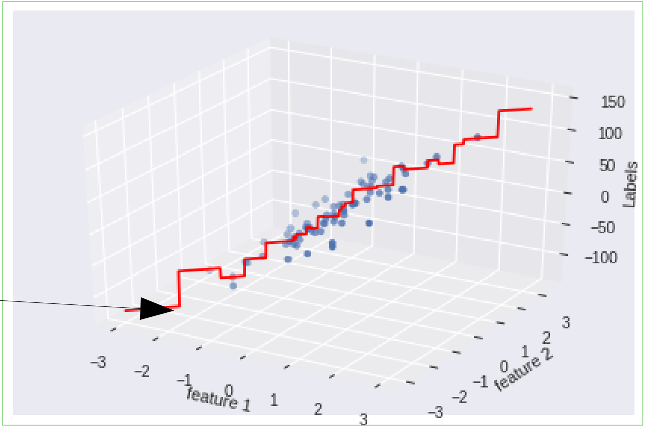
```
from sklearn.tree import DecisionTreeRegressor

xr_train,xr_test,yr_train,yr_test= train_test_split(xr, yr, test_size=0.25)

myModelR = DecisionTreeRegressor(max_depth=6)
myModelR.fit(xr_train, yr_train)
```

The decision tree regressor

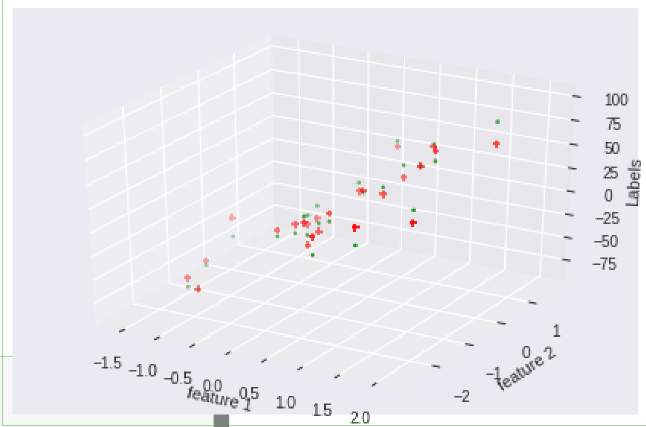
Predicting

```
xr_new_f1= np.linspace(-3,3,1000)
xr_new_f2= np.linspace(-3,3,1000)
xr_new1 = xr_new_f1.reshape(-1,1)
xr_new2 = xr_new_f2.reshape(-1,1)
xr_new = np.append(xr_new1,xr_new2,axis=1)


fig=plt.figure()

yrPred1 = myModelR.predict(xr_new)

ax2 = fig.add_subplot(111, projection='3d')
ax2.scatter(xr_train[:,0],xr_train[:,1],yr_train)
ax2.set_xlabel('feature 1')
ax2.set_ylabel('feature 2')
ax2.set_zlabel("Labels")


ax3 = fig.gca(projection='3d')

ax3.plot(xr_new_f1,xr_new_f2, yrPred1,c="r")

plt.tight_layout()

plt.show()
```

The model:different from a regular regression model




```
yrPred2 = myModelR.predict(xr_test)
fig=plt.figure()
ax4 = fig.add_subplot(111, projection='3d')
ax4.scatter(xr_test[:,0],xr_test[:,1],yr_test,c="g",marker=".")
ax4.set_xlabel('feature 1')
ax4.set_ylabel('feature 2')
ax4.set_zlabel("Labels")
ax4.scatter(xr_test[:,0],xr_test[:,1],yrPred2,c="r",marker="+")
plt.tight_layout()
plt.show()
```



5- Decision Tree Regressor example

[By Amina Delali]

5- Decision Tree Regressor example

Analyze the results

```
# the r2 coefficient determination
myModelR.score(xr_test,yr_test)
```

```
0.9108786174105965
```

- The score is calculated as follow:
  - $$R^2 = 1 - \left(\frac{u}{v}\right)$$

$$u = \sum (y_{true} - y_{pred})^2$$

$$v = \sum (y_{true} - \hat{y}_{true})^2$$

y_{true} : The true labels

y_{pred} : The predicted labels

$\hat{y}_{true}$: The mean of the true labels

- If the score was near 0, we would say, that the model is not different than calculating the mean (which is a bad thing)
- If it was negative, we would say, that the model is worse than calculating the mean value
- Since its value is approaching 1 (which is the best score), we can say that the model is good.

6- Visualization

Graphviz

Graphviz is a graph vizualization software. Scikitlearn enables you to export the trees into graphivz formats like .dot format.

```
!apt-get install graphviz
```

You have first to install graphviz package (system install)

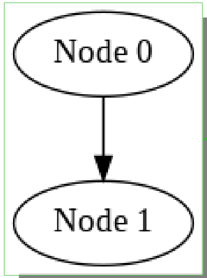
```
!pip install graphviz
```

Then you have to install the corresponding python graphviz library

```
import graphviz
```

Finally, import the library

```
from IPython.display import Image, display
display(Image("testing.dot.png"))
```




```
import graphviz
from graphviz import Digraph

dot = Digraph(comment='Testing Graphviz')
dot.node('A', 'Node 0')
dot.node('B', 'Node 1')
dot.edge('A', 'B')
print(dot.source)
```

```
// Testing Graphviz
digraph {
	A [label="Node 0"]
	B [label="Node 1"]
	A -> B
}
```

```
dot.render('testing.dot',format = "png")
```

```
'testing.dot.png'
```

[By Amina Delali]

Visualize a decision tree classifier

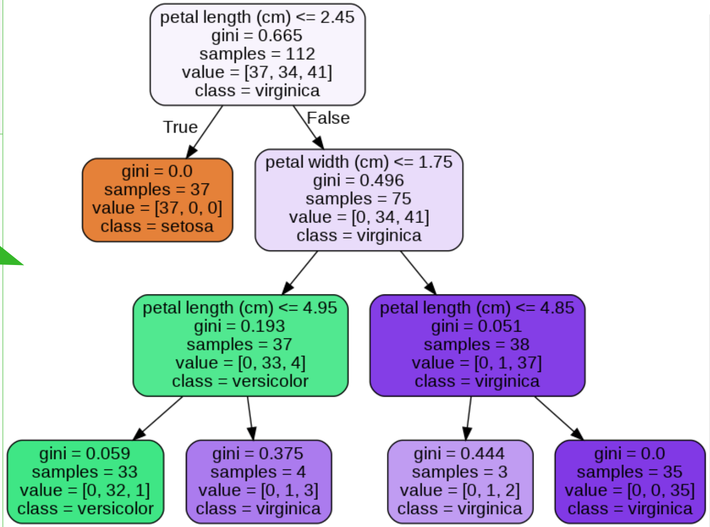
6-Visualization

```python
#import export_graphviz to export the tree as a graphviz (graph) file
from sklearn.tree import export_graphviz

# export to .dot file
export_graphviz(myModel,out_file="iris_tree.dot",feature_names=myIris.feature_names[:],
                class_names=myIris.target_names,rounded=True,filled=True)

from IPython.display import Image, display
# read the dot file
with open("iris_tree.dot") as f:
    dot_graph = f.read()

# create a graph from the dot file: Verbatim DOT source code string(dot_graph)
#to be rendered by Graphviz.
g = graphviz.Source(dot_graph, format="png")
# save the source and render with graphivs engine

display(Image(g.render()))
```

Export to a dot file

```
[5.  3.5 1.6 0.6]
```

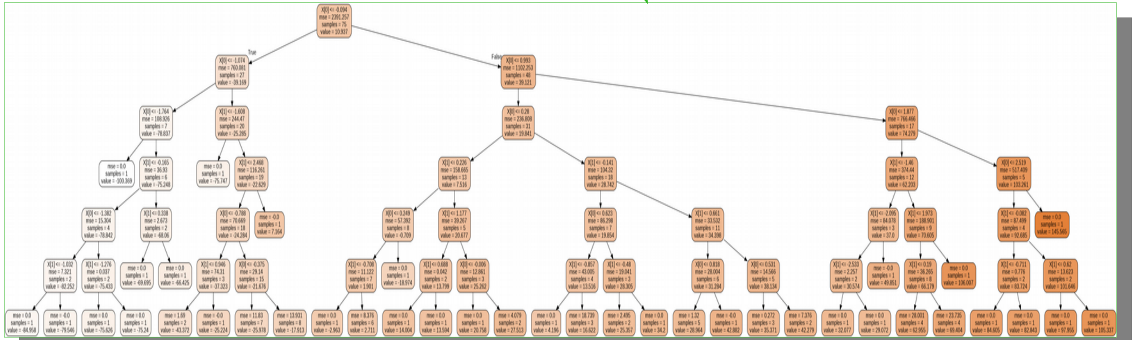
The petal length==1.6 <=2.45 ==> goes to the left node==> classed setosa.



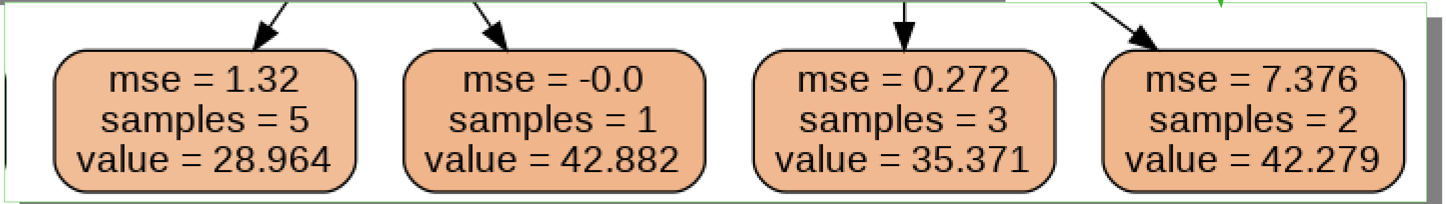
[By Amina Delali]

Visualize a decision tree regressor

```python
# export to .dot file
export_graphviz(myModelR,out_file="iris_treeR.dot",rounded=True,filled=True)

# read the dot file
with open("iris_treeR.dot") as f:
    dot_graphR = f.read()

# create a graph from the dot file: Verbatim DOT source code string(dot_graph)
#to be rendered by Graphviz.
g2 = graphviz.Source(dot_graphR, format="png")
# save the source and render with graphivs engine

display(Image(g2.render()))
```



In the leafs we have float values instead of classes



[By Amina Delali]

6- Visualization

References

- Python Software Foundation. Simple python interface for graphviz. On-line at https://pypi.org/project/graphviz/, 2018. Accessed on 01-12-2018.
- Aurélien Géron. Hands-on machine learning with Scikit-Learn and Tensor-Flow: concepts, tools, and techniques to build intelligent systems. O’Reilly Media, Inc, 2017.
- Joshi Prateek. Artificial intelligence with Python. Packt Publishing, 2017.
- Scikit-learn.org. scikit-learn, machine learning in python. On-line at https://scikit-learn.org/stable/. Accessed on 01-12-2018.
- Jake VanderPlas. Python data science handbook: essential tools for working with data. O’Reilly Media, Inc, 2017.

Thank you!

FOR ALL YOUR TIME