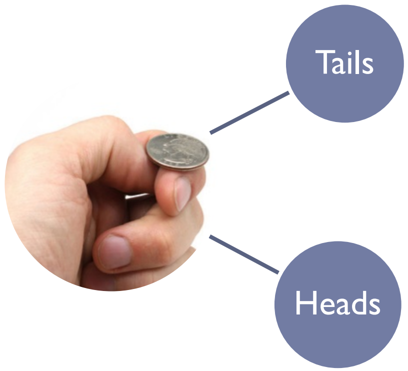
Decision Trees

Prof.Dr. Bahadır AKTUĞ
Machine Learning with Python

**Compiled from sources given in the references.*

Decision Trees

- There are a few variants of decision trees which are altogether grouped as «tree-based methods»
  - Decision Trees
  - Random Forests
  - Boosted Trees
- In the most fundamental form, a decision tree can be thought of a flowchart mapping out an outcome.



Historical Evolution

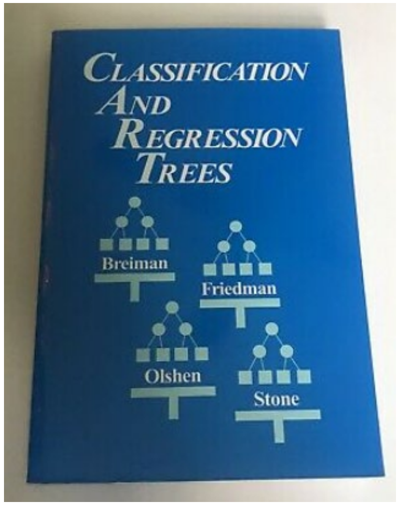
- Decision trees are based on split nodes where the data can be split for thresholds
- Piece-wise Constant Model (Morgan and Sonquist, 1963)
  - Splits are based on node impurity which is basically an error metric defined by the squared difference of the observation and prediction

$$\phi(t) = \sum_{i \in t} (y_i - \bar{y})^2$$

- Modified Split Condition (Messenger and Mandell, 1972)
  - THAID (Theta Automatic Interaction Detection)

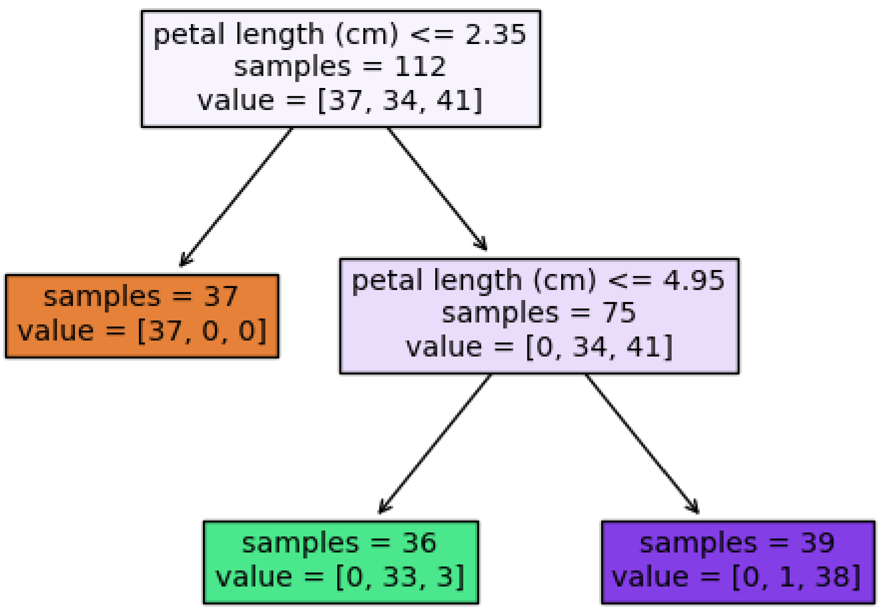
Historical Evolution

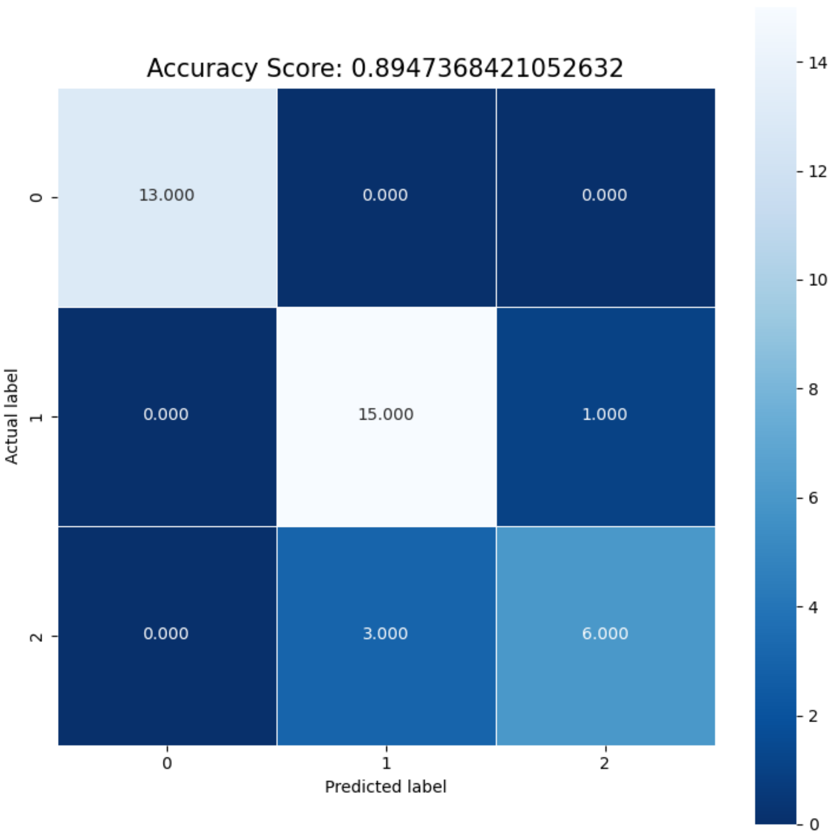
- CART Algorithms (Breiman and Stone, 1970)
  - CART (Classification and Regression Tree)
- CHAID Decision Tree (Kass, 1980)
  - CHAID (Chi-squared Automatic Interaction Detection)
- The CART Book (Breiman et al., 1984)
  - A universal standard now and introduces many concepts:
    - Cross validation of Trees
    - Pruning Trees
    - Surrogate Splits
    - Variable Importance
    - Search for Linear Splits



1. Example: Classifying IRIS Dataset

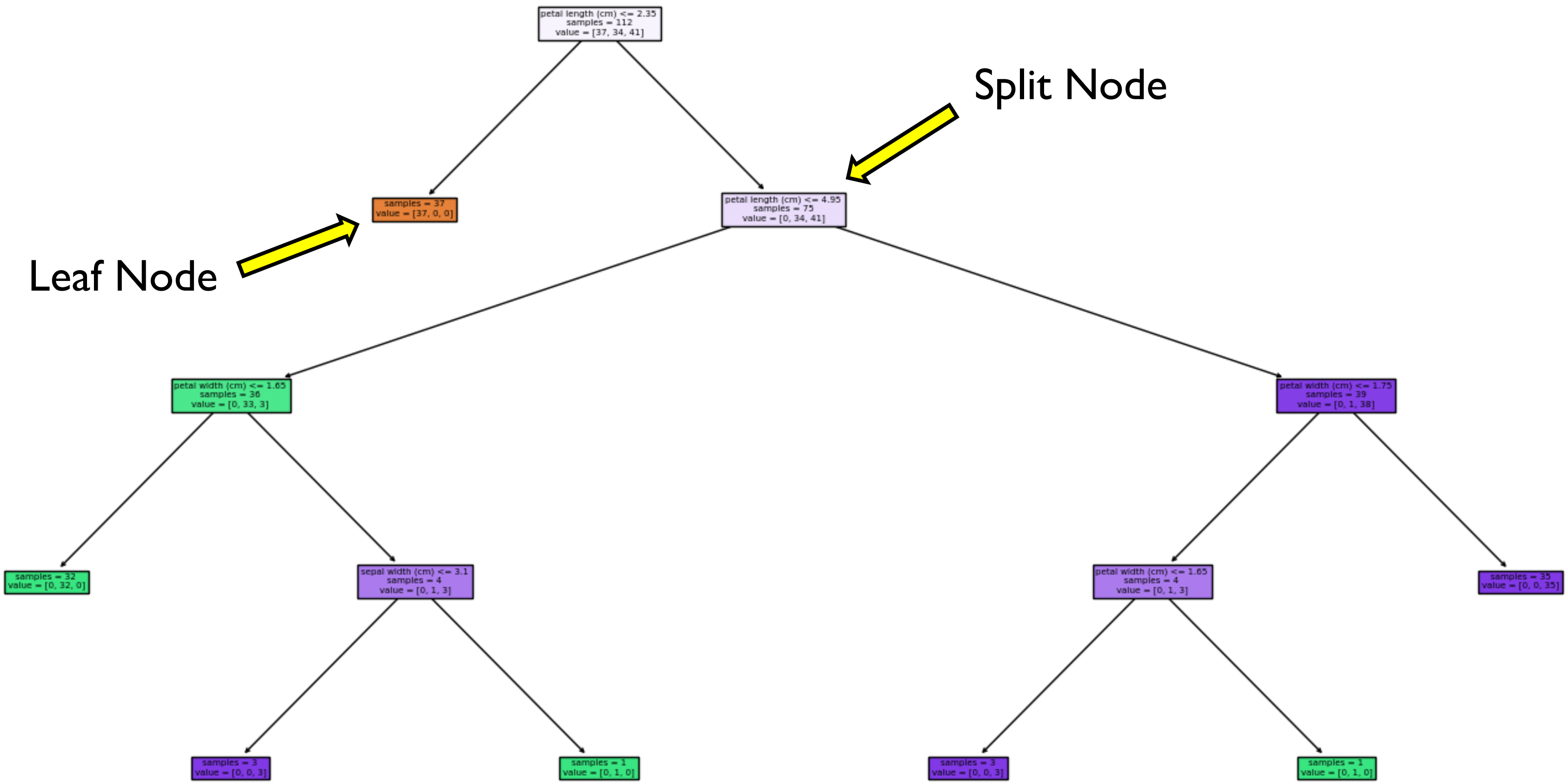
```
from sklearn.tree import DecisionTreeRegressor
```



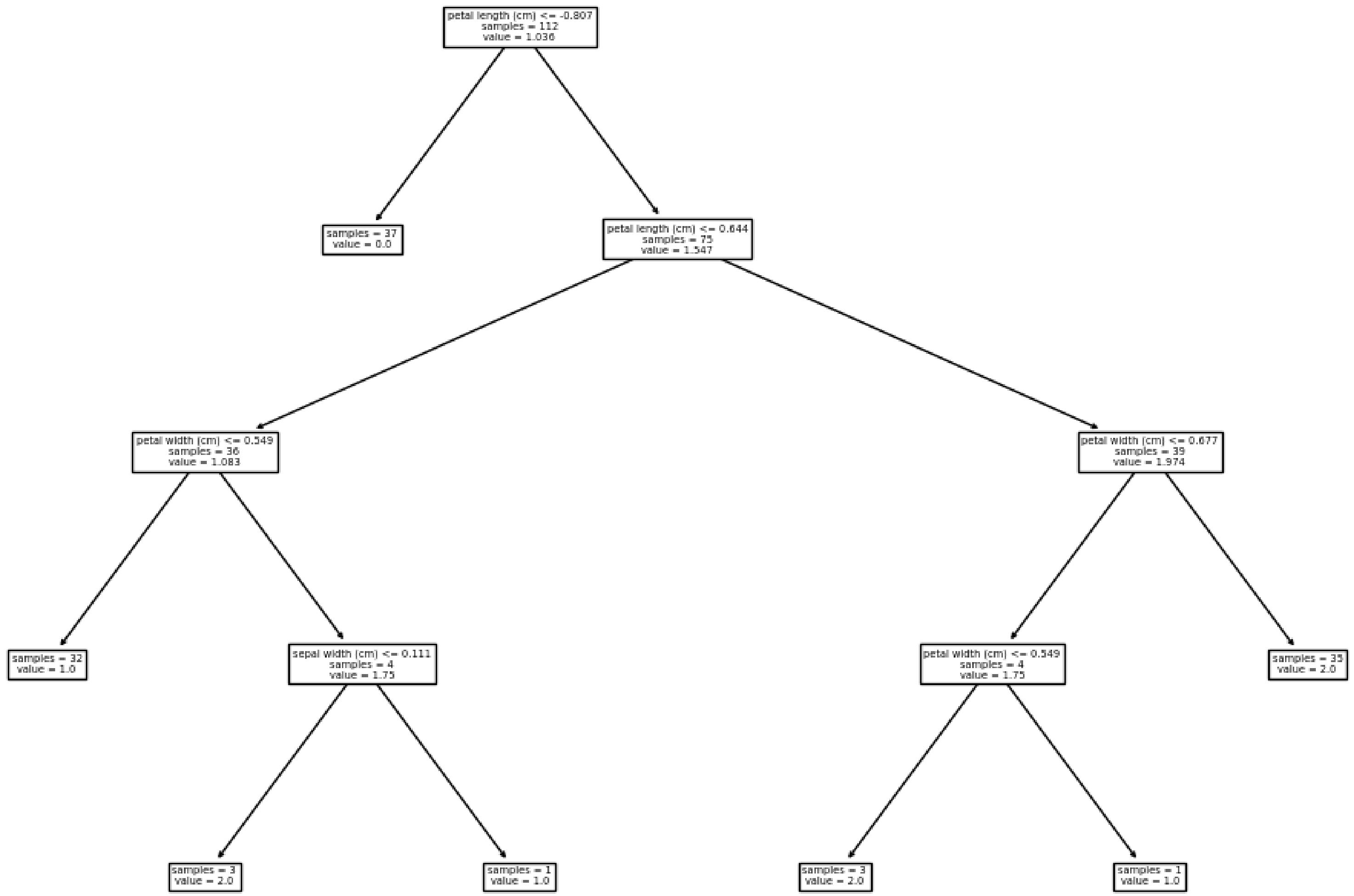


1. Example: Classifying IRIS Dataset

```
dt = DecisionTreeClassifier(max_depth=4)
```



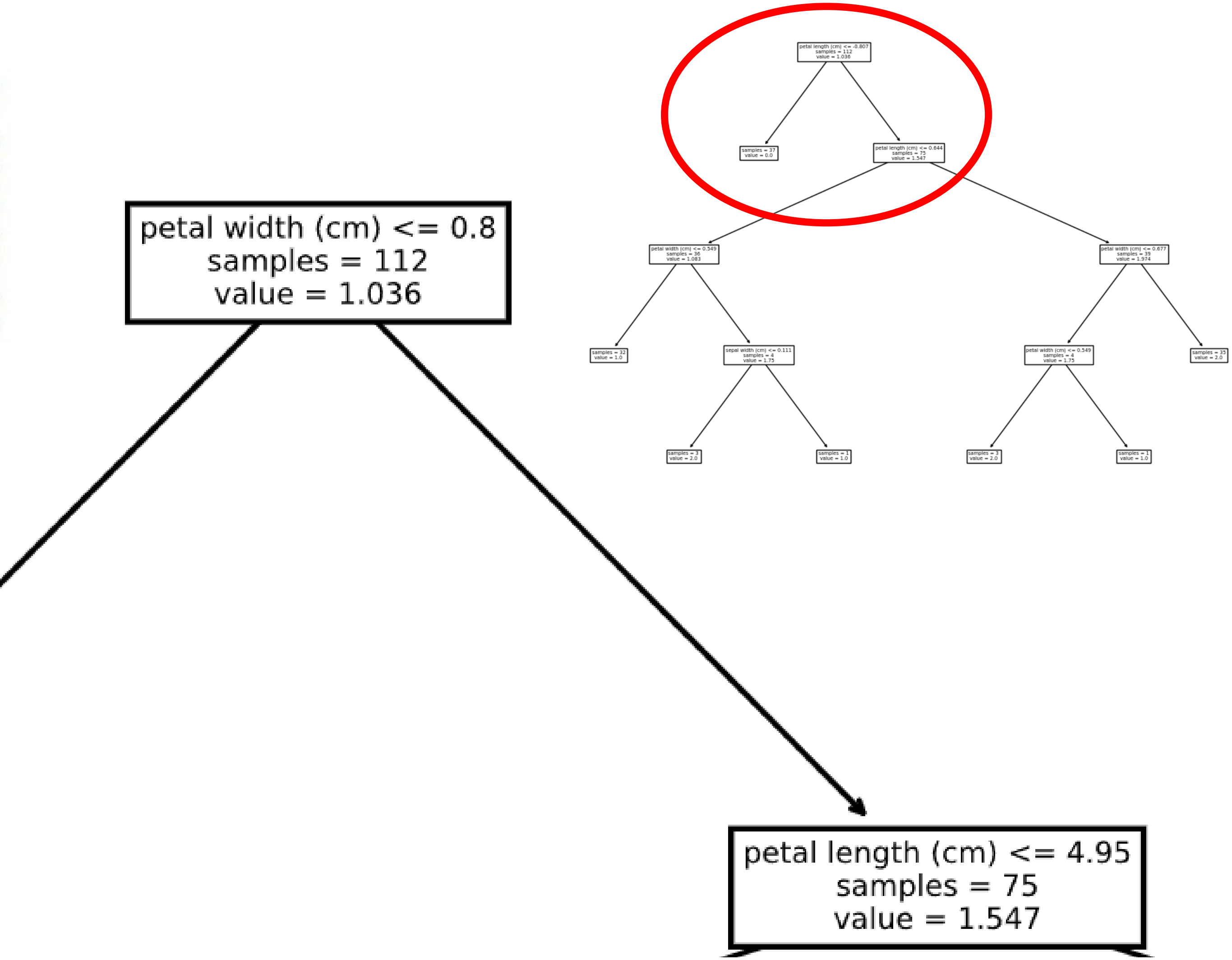
1. Example: Classifying IRIS Dataset



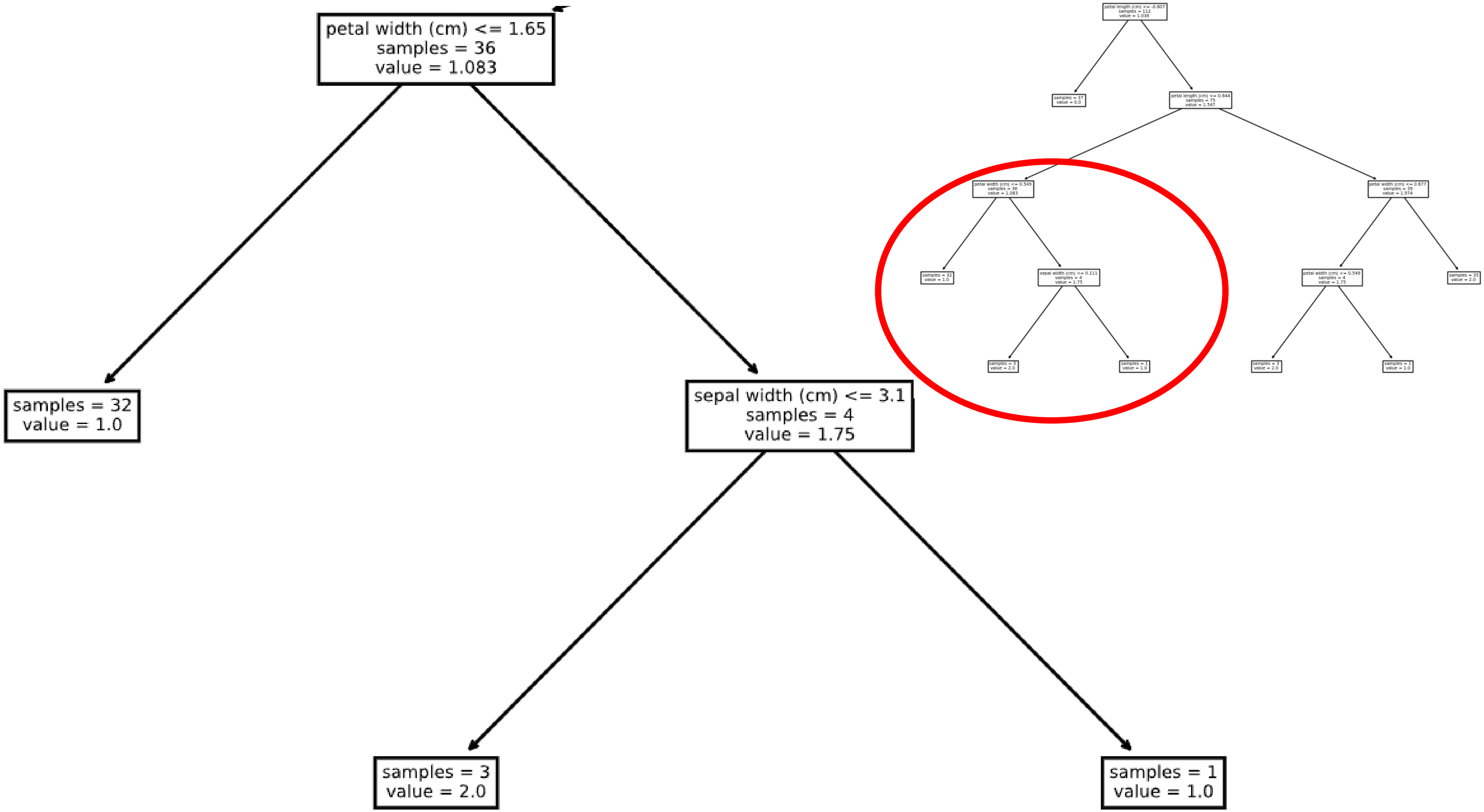
1. Example: Classifying IRIS Dataset

Training / test data

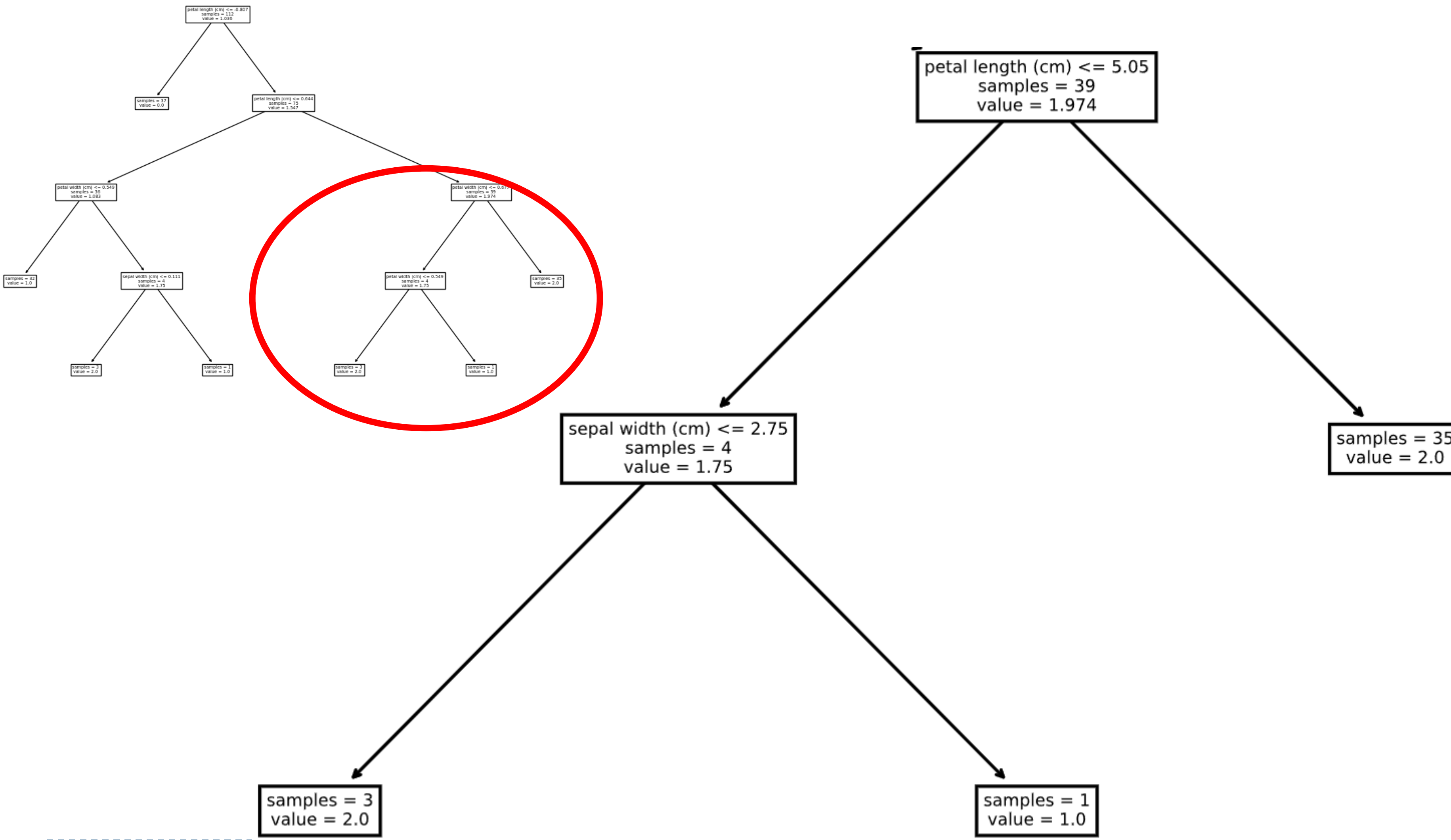
Features				Labels
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica



1. Example: Classifying IRIS Dataset



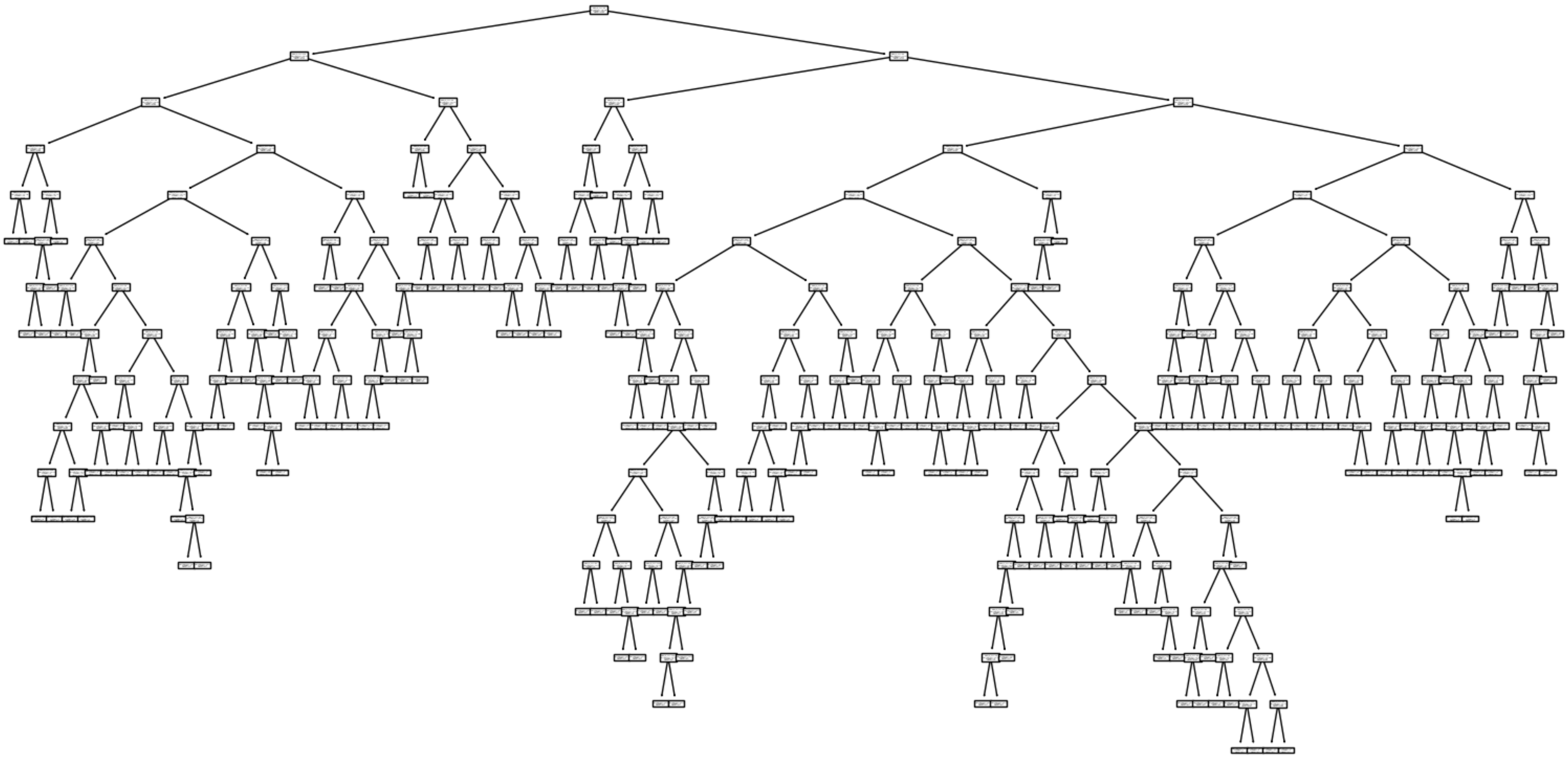
1. Example: Classifying IRIS Dataset



1. Example: Classifying IRIS Dataset

```
Algorithm: DecisionTreeRegressor
Score : 0.9536585365853658

Confusion Matrix
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      0.94      0.97        16
           2       0.90      1.00      0.95         9

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

2. Example: Classifying Handwritten Digits



2. Example: Classifying Handwritten Digits

```
dataLoaded = load_digits()
# dataName = dataLoaded.filename[:-4]
print("Data Shape" , dataLoaded.data.shape)
print("Label Shape", dataLoaded.target.shape)

x = dataLoaded.data
# x = dataLoaded.data[:,:2]
y = dataLoaded.target

x_train,x_test,y_train,y_test=train_test_split(x,y,
                                  test_size=0.25,random_state=0)
```

```
# Classification
classifier = DecisionTreeRegressor()
methodName = type(classifier).__name__
classifier.fit(x_train, y_train)
predictions = classifier.predict(x_test)
```

2. Example: Classifying Handwritten Digits

```
Data Shape (1797, 64)
Label Shape (1797,)

Algorithm: DecisionTreeRegressor
Score : 0.6368961260924944

Confusion Matrix
[[35  0  1  0  0  0  1  0  0  0]
 [ 0 38  0  1  0  0  1  0  2  1]
 [ 1  2 33  2  0  0  1  0  3  2]
 [ 0  0  2 37  0  2  0  3  1  0]
 [ 1  0  0  0 32  0  1  2  1  1]
 [ 1  1  0  1  1 40  2  1  0  1]
 [ 1  0  0  0  0  0 50  1  0  0]
 [ 0  0  1  2  1  0  1 39  4  0]
 [ 0  0  1  1  0  2  1  4 35  4]
 [ 2  1  1  3  0  1  1  3  2 33]]
```

```
              precision    recall  f1-score   support

           0       0.85      0.95      0.90        37
           1       0.90      0.88      0.89        43
           2       0.85      0.75      0.80        44
           3       0.79      0.82      0.80        45
           4       0.94      0.84      0.89        38
           5       0.89      0.83      0.86        48
           6       0.85      0.96      0.90        52
           7       0.74      0.81      0.77        48
           8       0.73      0.73      0.73        48
           9       0.79      0.70      0.74        47

    accuracy                           0.83       450
   macro avg       0.83      0.83      0.83       450
weighted avg       0.83      0.83      0.83       450
```

3. Example: Classifying Diabetes Dataset

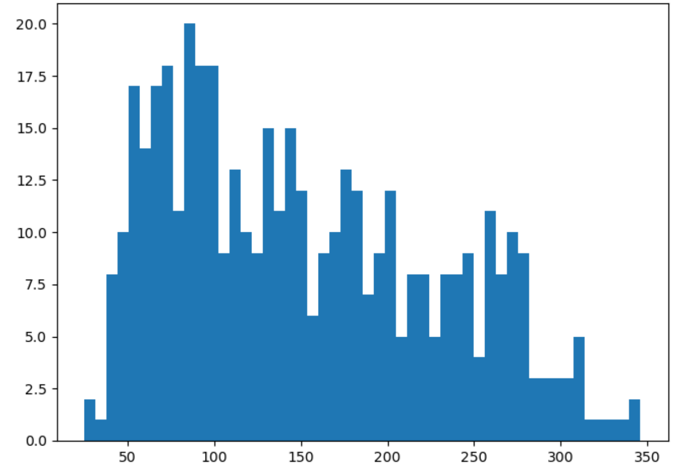
- We will use «Diabetes» dataset as example for this algorithm which is included in the Scikit Learn module.
- Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.
- Each of these 10 feature variables have been mean centered and scaled by the standard deviation times n_samples (i.e. the sum of squares of each column totals 1).
- URL: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

Data Set Characteristics

Number of Instances:	442
Number of Attributes:	First 10 columns are numeric predictive values
Target:	Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information:	• age age in years • sex • bmi body mass index • bp average blood pressure • s1 tc, T-Cells (a type of white blood cells) • s2 ldl, low-density lipoproteins • s3 hdl, high-density lipoproteins • s4 tch, thyroid stimulating hormone • s5 ltg, lamotrigine • s6 glu, blood sugar level

Histogram of Disease Progression

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_diabetes

diabetes = load_diabetes()

X = diabetes.data
y = diabetes.target

X_feature_names = ['age', 'gender', 'body mass index', 'average blood pressure',
                   'bl_0','bl_1','bl_2','bl_3','bl_4','bl_5']

pd.Series(y).hist(bins=50)
plt.grid(b=None)
plt.show()
```



Decision Trees (Regression)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.2,stratify=binned_y)
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
y_pred = dtr.predict(X_test)
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
print((np.abs(y_test - y_pred)/(y_test)).mean())
```

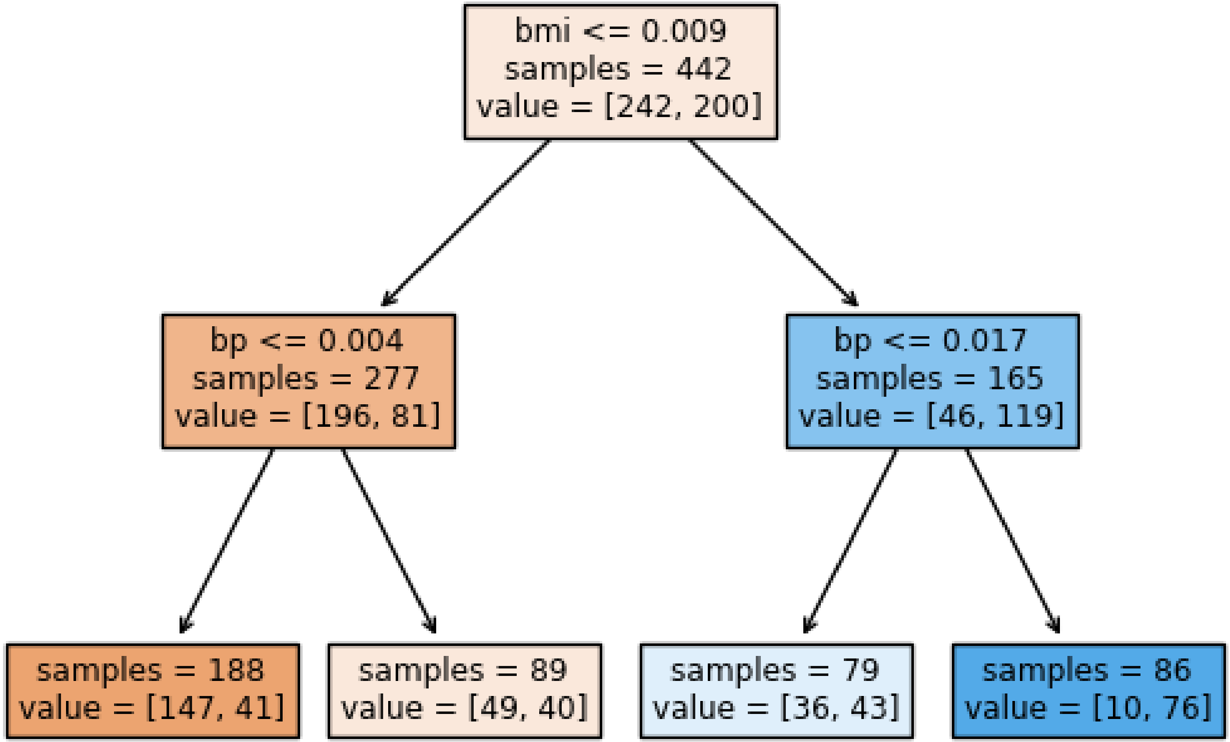
59.68539325842696

0.49967689870633586 → The answer

Since the test and train groups will be chosen randomly you will get a different but similar (hopefully!) result each time.

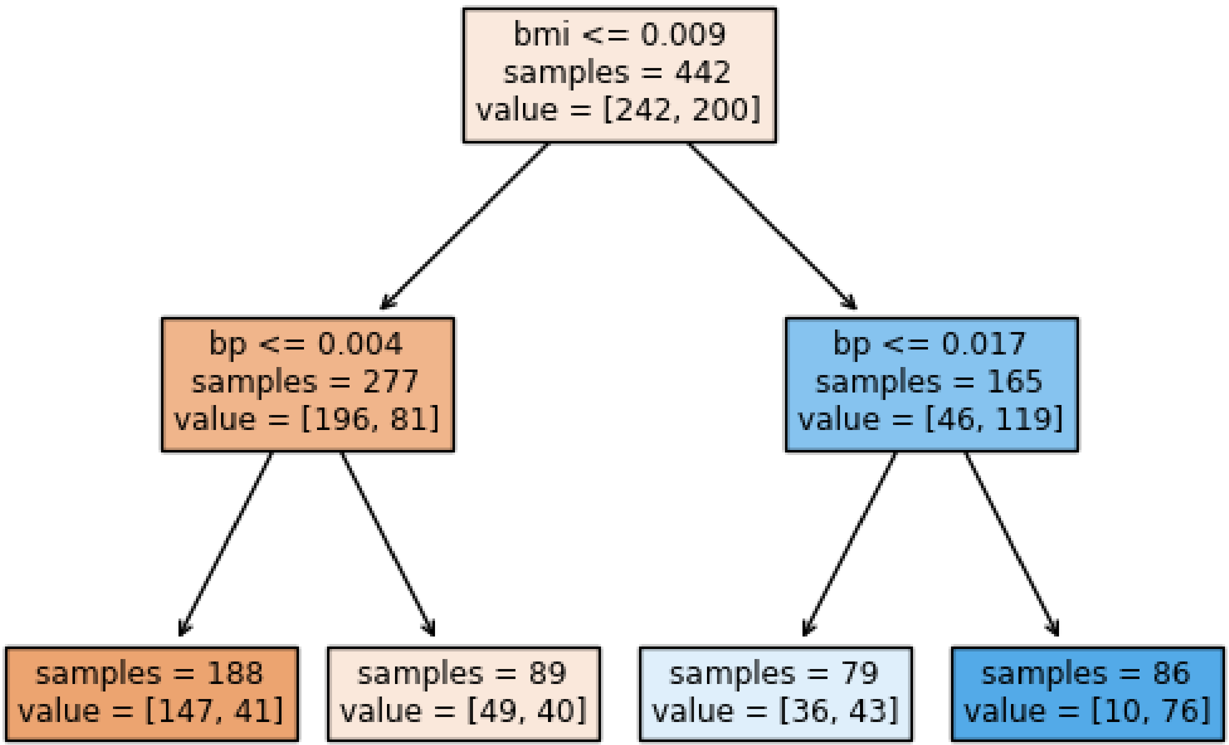
Decision Trees (Classification)

```
from matplotlib import pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
# create and fit the model
clf = DecisionTreeClassifier(max_depth=2)
clf.fit(diabetes_X.iloc[:,:4], diabetes_y_risk)
# visualize the model
plot_tree(clf, feature_names=diabetes_X.columns[:4], impurity=False)
```



Decision Trees

$$r(x) = \begin{cases} \text{true} & \text{if } x_{\text{bmi}} \leq 0.009 \\ \text{false} & \text{if } x_{\text{bmi}} > 0.009 \end{cases} \qquad R = \{x \in \mathcal{X} \mid x_{\text{bmi}} \leq 0.009 \text{ and } x_{\text{bp}} > 0.004\}$$



References

1 Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).
2 Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.
3 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
4 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
5 Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.
6 Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.
7 John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.
8 Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.
9 Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.
10 http://www.diveintopython.net/
11 https://docs.python.org/3/tutorial/
12 http://www.python-course.eu
13 https://developers.google.com/edu/python/
14 http://learnpythonthehardway.org/book/