Decision-Tree for Classification

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

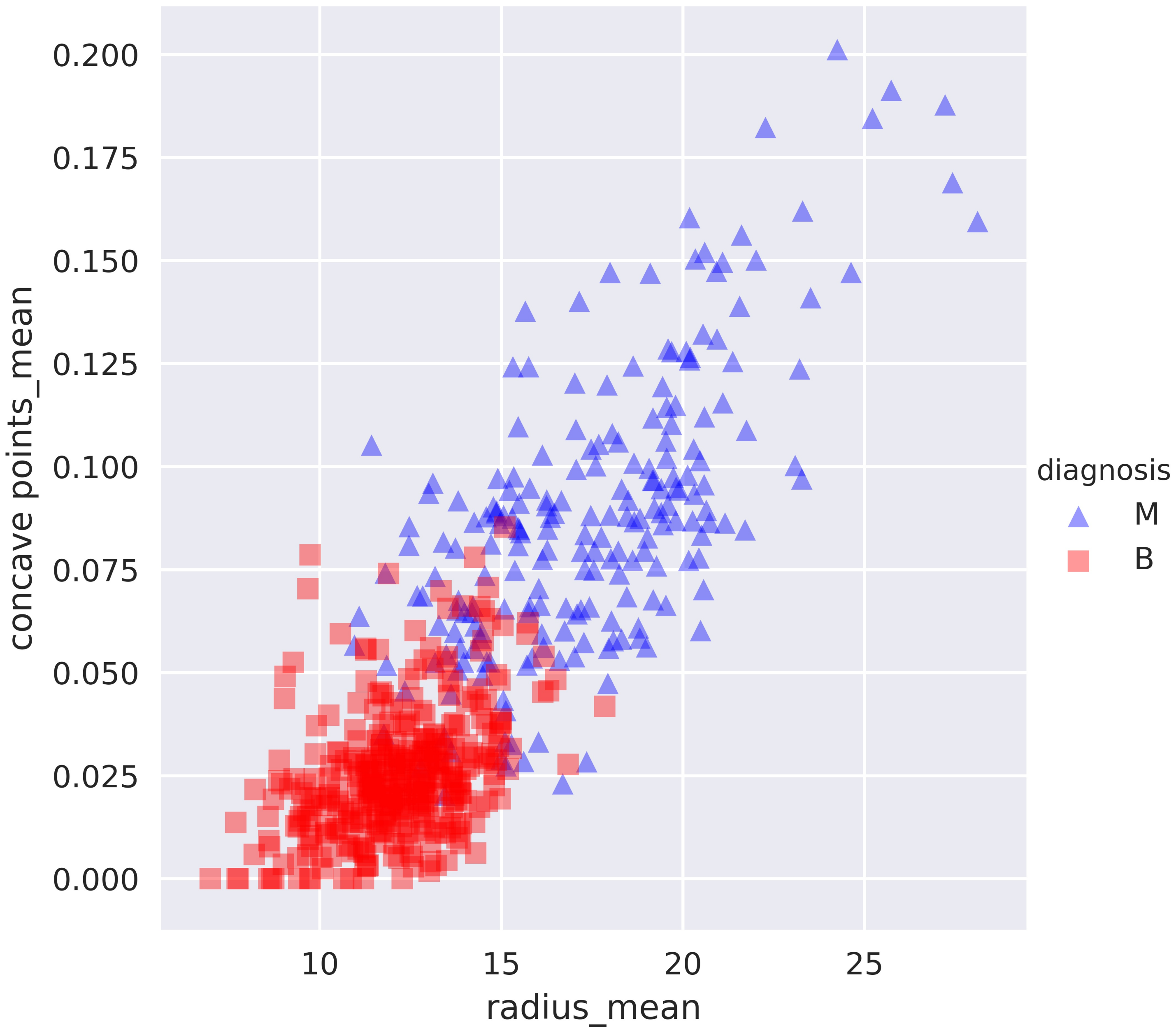
Course Overview

- **Chap 1:** Classification And Regression Tree (CART)
- **Chap 2:** The Bias-Variance Tradeoff
- **Chap 3:** Bagging and Random Forests
- **Chap 4:** Boosting
- **Chap 5:** Model Tuning

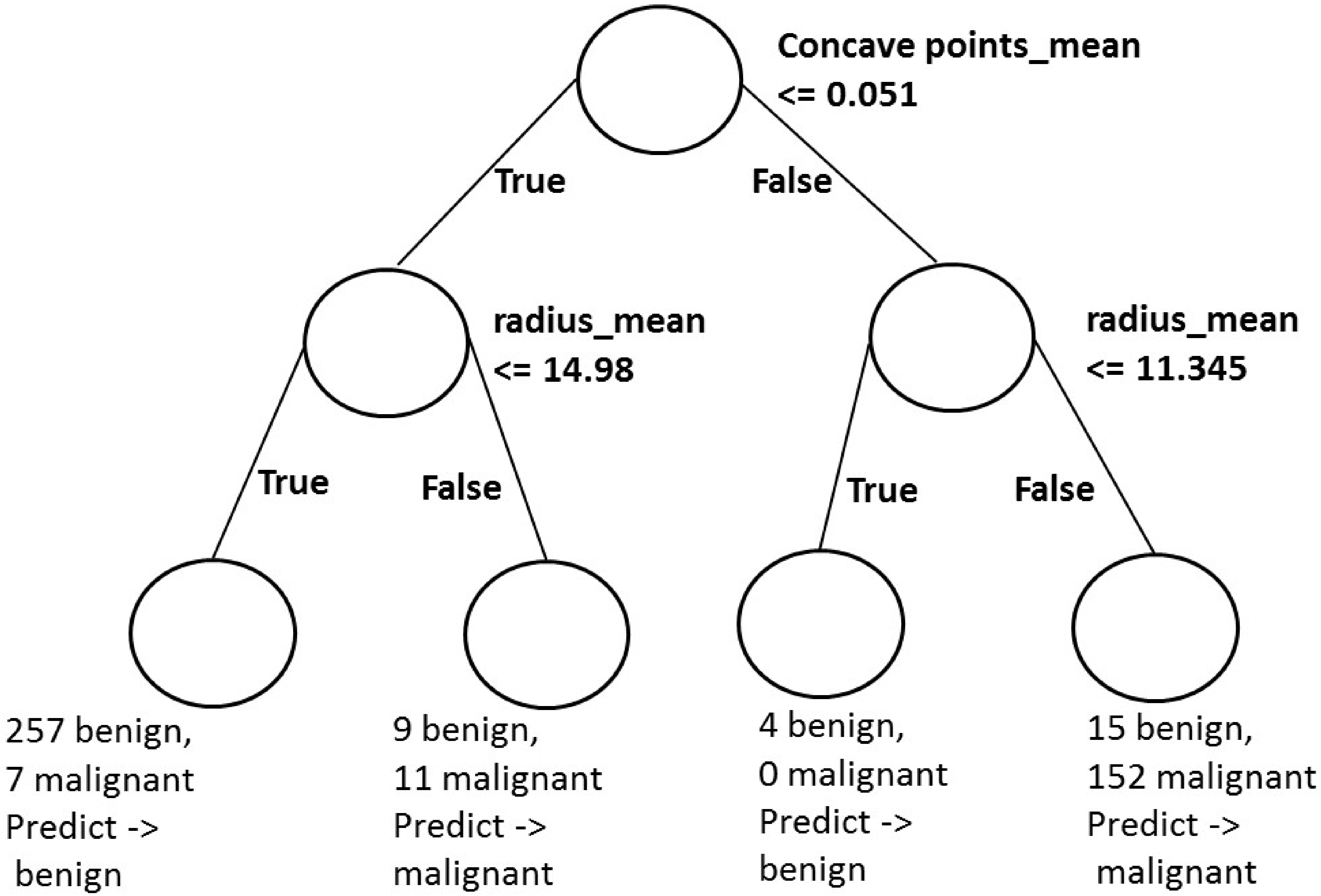
Classification-tree

- Sequence of if-else questions about individual features.
- **Objective**: infer class labels.
- Able to capture non-linear relationships between features and labels.
- Don't require feature scaling (ex: Standardization, ..)

Breast Cancer Dataset in 2D



Decision-tree Diagram



Classification-tree in scikit-learn

```
# Import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Split the dataset into 80% train, 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=1)

# Instantiate dt
dt = DecisionTreeClassifier(max_depth=2, random_state=1)
```

Classification-tree in scikit-learn

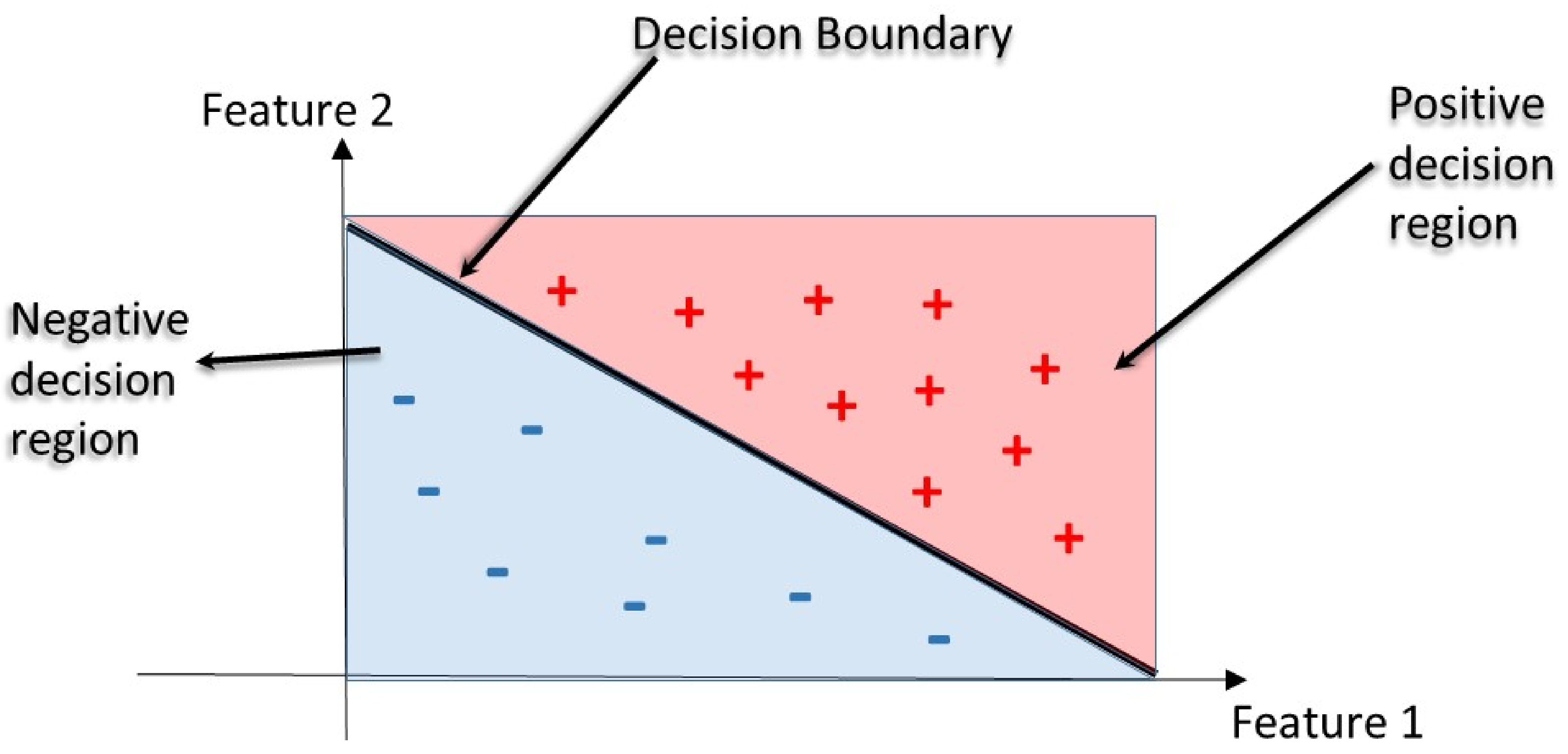
```
# Fit dt to the training set
dt.fit(X_train,y_train)

# Predict the test set labels
y_pred = dt.predict(X_test)
# Evaluate the test-set accuracy
accuracy_score(y_test, y_pred)
```

```
0.90350877192982459
```

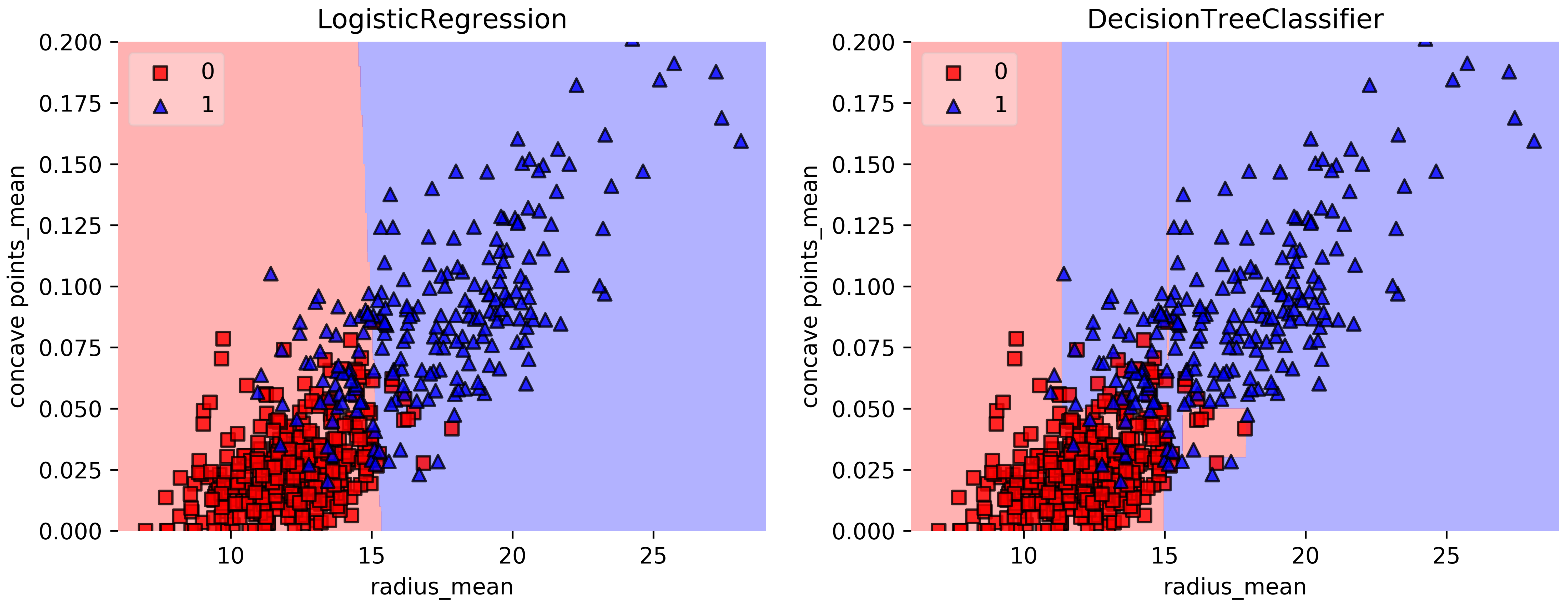
Decision Regions

Decision region: region in the feature space where all instances are assigned to one class label.

Decision Boundary: surface separating different decision regions.



Decision Regions: CART vs. Linear Model



Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Classification-Tree Learning

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

Building Blocks of a Decision-Tree

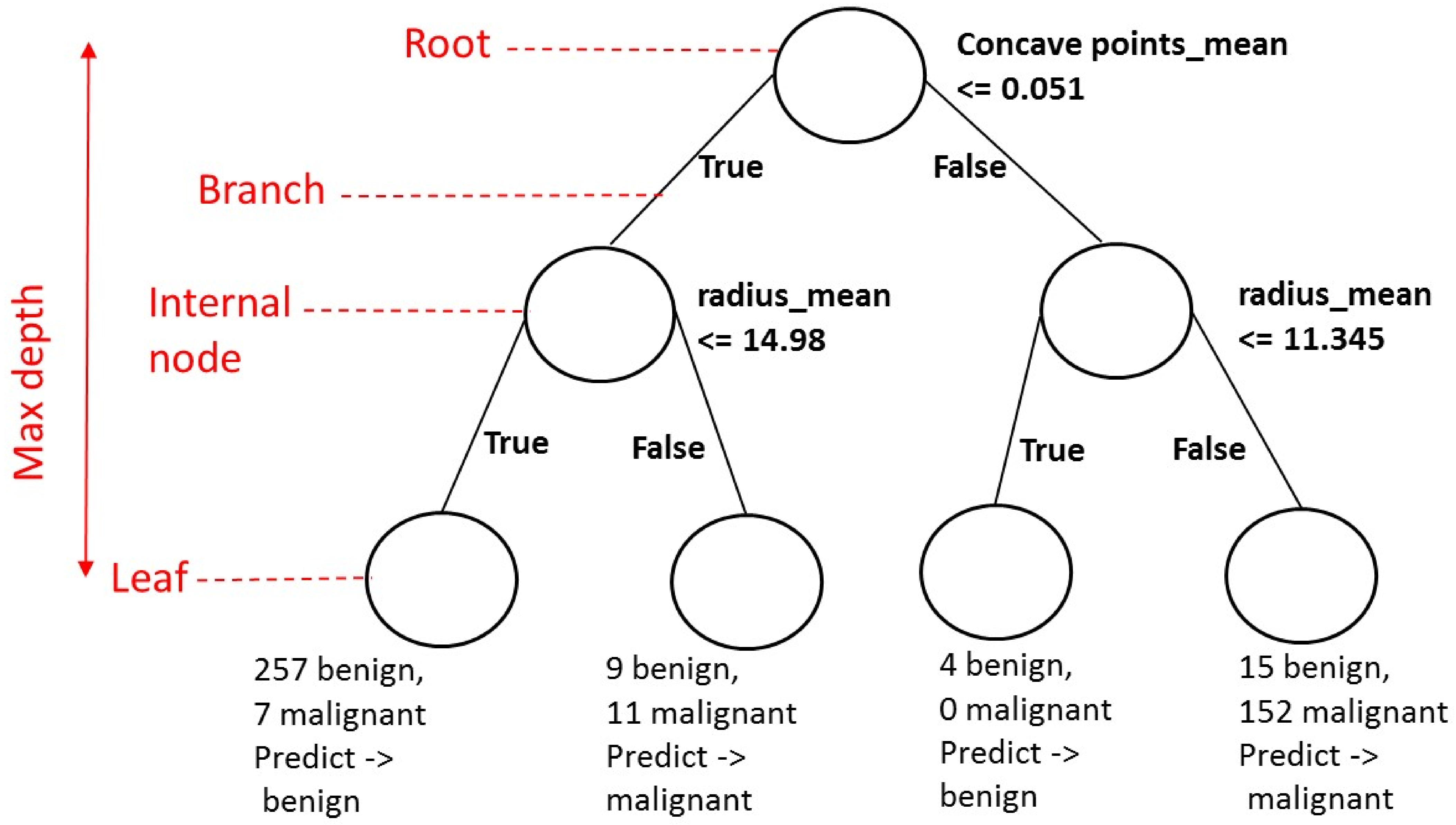
- **Decision-Tree:** data structure consisting of a hierarchy of nodes.
- **Node**: question or prediction.

Building Blocks of a Decision-Tree

Three kinds of nodes:

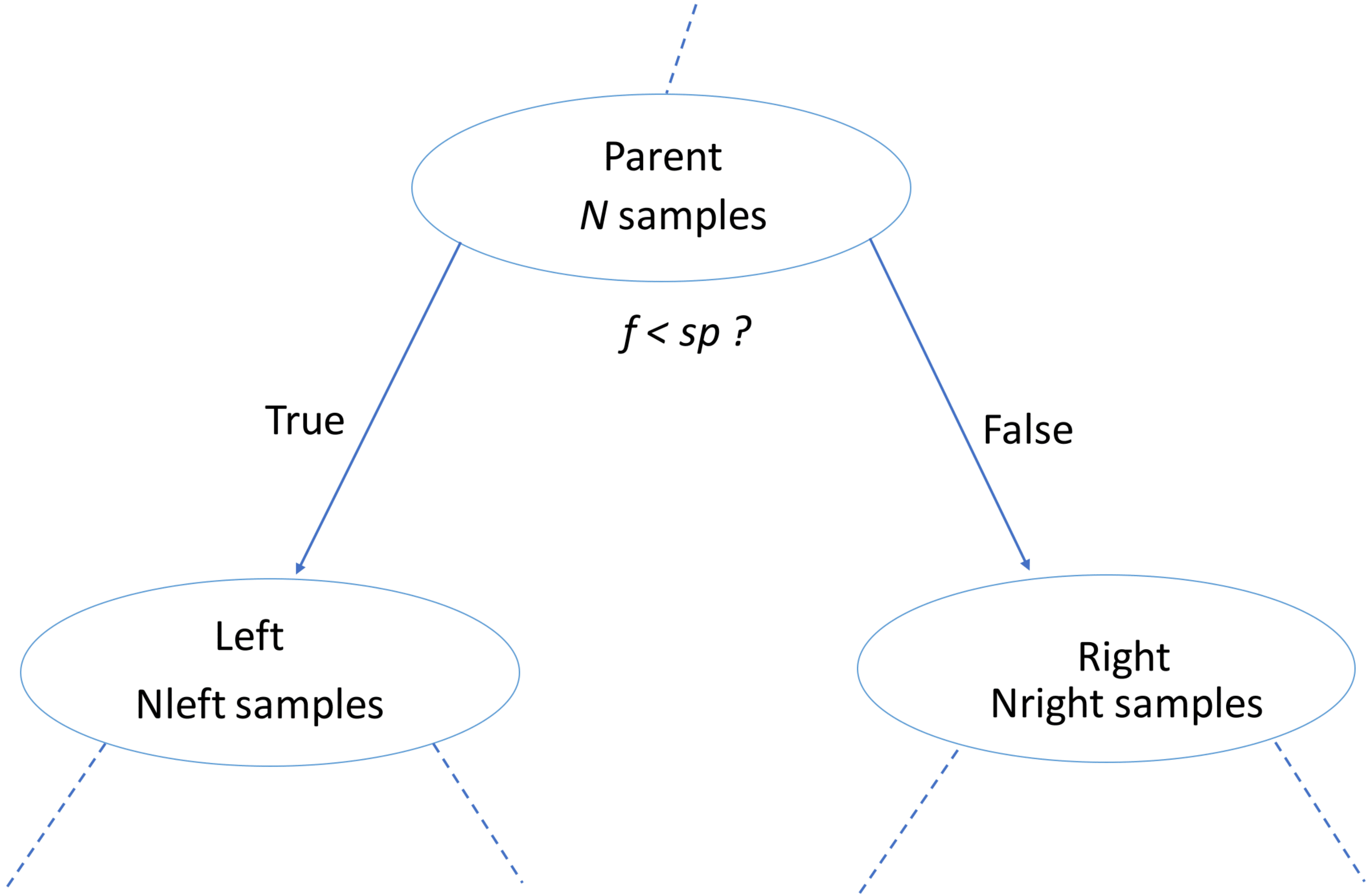
- **Root**: *no* parent node, question giving rise to *two* children nodes.
- **Internal node**: *one* parent node, question giving rise to *two* children nodes.
- **Leaf**: *one* parent node, *no* children nodes --> *prediction*.

Prediction



Root
Concave points_mean
<= 0.051
Branch
True
False
Internal
node
radius_mean
<= 14.98
radius_mean
<= 11.345
Max depth
True
False
True
False
Leaf
257 benign,
7 malignant
Predict ->
benign
9 benign,
11 malignant
Predict ->
malignant
4 benign,
0 malignant
Predict ->
benign
15 benign,
152 malignant
Predict ->
malignant

Information Gain (IG)



Information Gain (IG)

$$IG(\underbrace{f}_{feature}, \underbrace{sp}_{split-point}) = I(parent) - \left(\frac{N_{left}}{N} I(left) + \frac{N_{right}}{N} I(right)\right)$$

Criteria to measure the impurity of a node $I(node)$:

- gini index,
- entropy. ...

Classification-Tree Learning

- Nodes are grown recursively.
- At each node, split the data based on:
  - feature f and split-point sp to maximize $IG(\mathrm{node})$.
- If $IG(\mathrm{node})$= 0, declare the node a leaf. ...

```python
# Import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Split dataset into 80% train, 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                   test_size=0.2,
                                                   stratify=y,
                                                   random_state=1)
# Instantiate dt, set 'criterion' to 'gini'
dt = DecisionTreeClassifier(criterion='gini', random_state=1)
```

Information Criterion in scikit-learn

```
# Fit dt to the training set
dt.fit(X_train,y_train)

# Predict test-set labels
y_pred= dt.predict(X_test)

# Evaluate test-set accuracy
accuracy_score(y_test, y_pred)
```

```
0.92105263157894735
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Decision-Tree for Regression

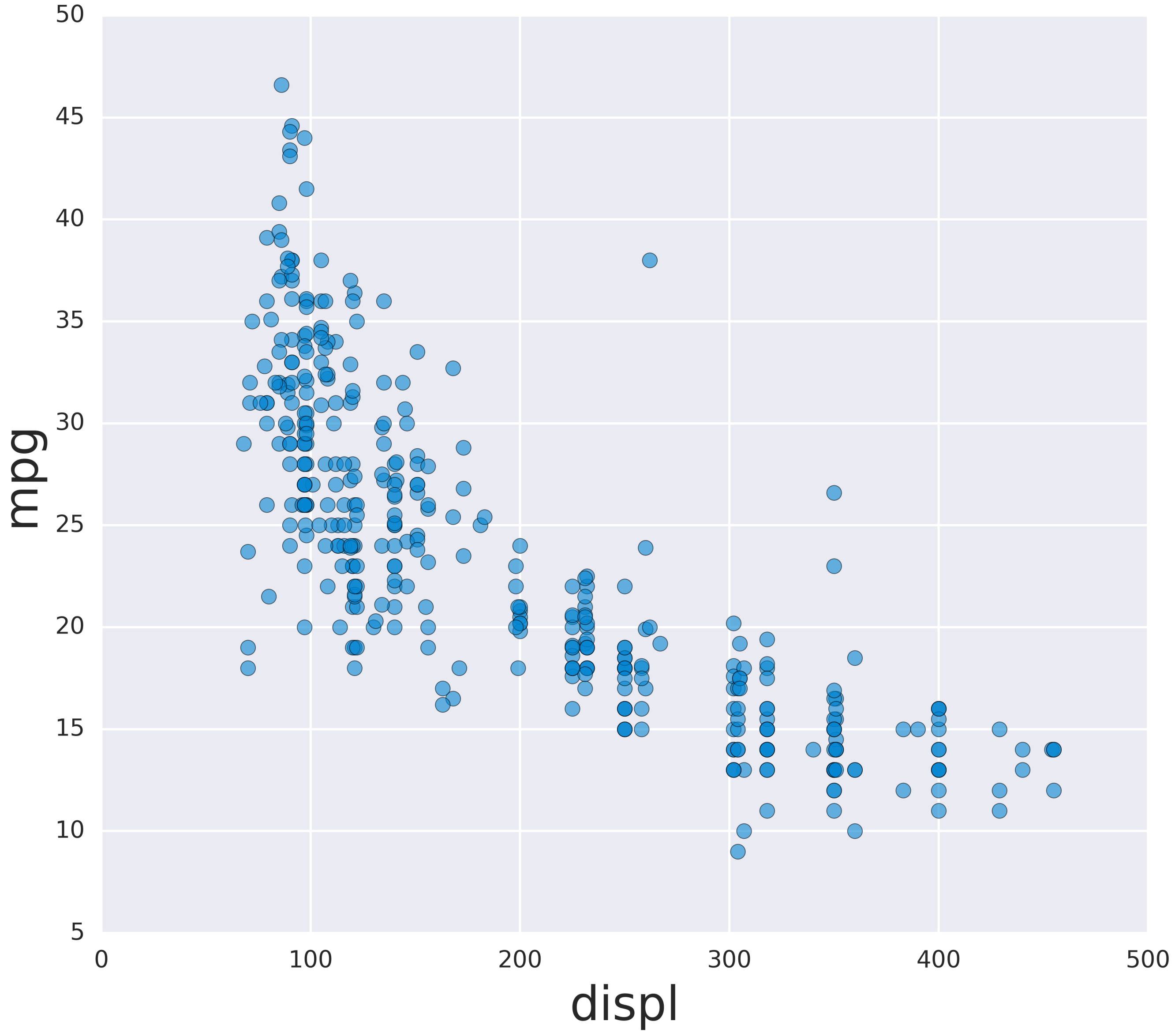
MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

Auto-mpg Dataset

	mpg	displ	hp	weight	accel	origin	size
0	18.0	250.0	88	3139	14.5	US	15.0
1	9.0	304.0	193	4732	18.5	US	20.0
2	36.1	91.0	60	1800	16.4	Asia	10.0
3	18.5	250.0	98	3525	19.0	US	15.0
4	34.3	97.0	78	2188	15.8	Europe	10.0
5	32.9	119.0	100	2615	14.8	Asia	10.0

Auto-mpg with one feature



Regression-Tree in scikit-learn

```
# Import DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import mean_squared_error as MSE
from sklearn.metrics import mean_squared_error as MSE
# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                   test_size=0.2,
                                                   random_state=3)

# Instantiate a DecisionTreeRegressor 'dt'
dt = DecisionTreeRegressor(max_depth=4,
                           min_samples_leaf=0.1,
                           random_state=3)
```

Regression-Tree in scikit-learn

```python
# Fit 'dt' to the training-set
dt.fit(X_train, y_train)
# Predict test-set labels
y_pred = dt.predict(X_test)
# Compute test-set MSE
mse_dt =  MSE(y_test, y_pred)
# Compute test-set RMSE
rmse_dt = mse_dt**(1/2)
# Print rmse_dt
print(rmse_dt)
```

```
5.1023068889
```

Information Criterion for Regression-Tree

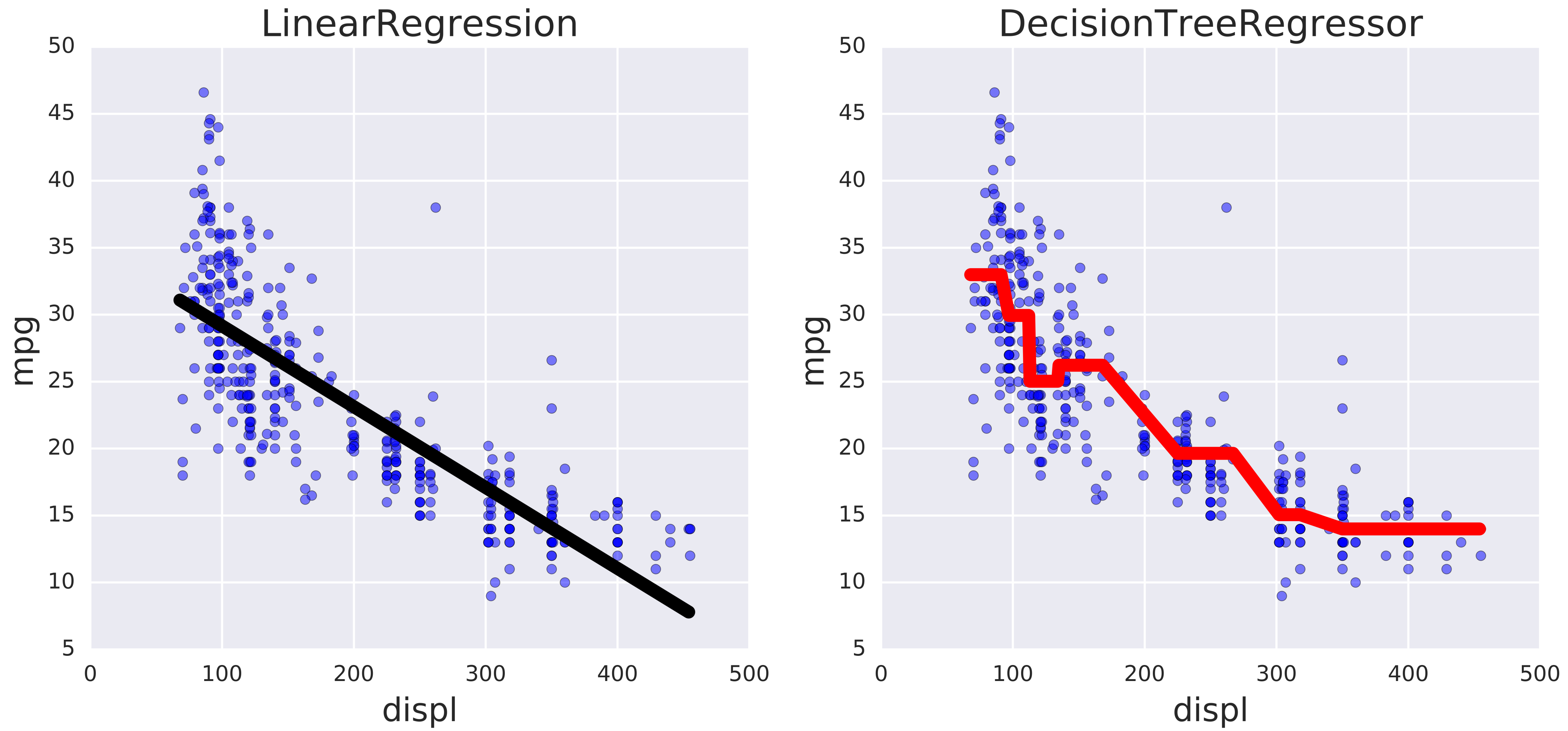
$$I(\text{node}) = \underbrace{\text{MSE}(\text{node})}_{mean-squared-error} = \frac{1}{N_{node}} \sum_{i \in node} \left(y^{(i)} - \hat{y}_{node}\right)^2$$

$$\underbrace{\hat{y}_{node}}_{mean-target-value} = \frac{1}{N_{node}} \sum_{i \in node} y^{(i)}$$

Prediction

$$\hat{y}_{pred}(\text{leaf}) = \frac{1}{N_{leaf}} \sum_{i \in leaf} y^{(i)}$$

Linear Regression vs. Regression-Tree



Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON