Decision-Tree for Classification

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Classification-tree

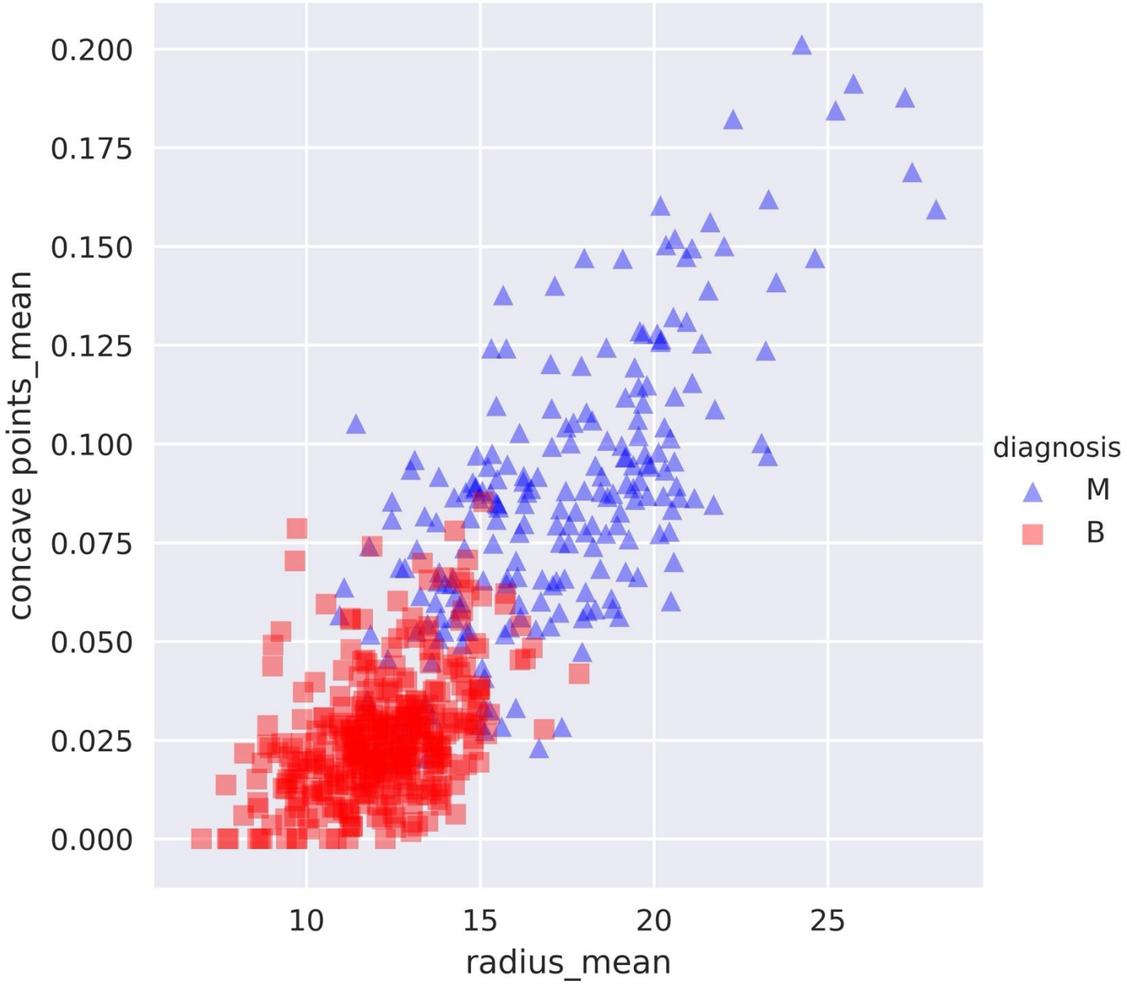
- Sequence of if-else questions about individual features.
- **Objective**: infer class labels.
- Able to capture non-linear relationships between features and labels.
- Don't require feature scaling (ex: Standardization, ..)

Classification tree methods (such as **Decision Trees**, **Random Forests**, and **Gradient Boosted Trees**) do not require feature scaling (like standardization or normalization) because they are inherently **scale-invariant (Invariant** means **unchanging** or **remaining constant** under specific transformations or conditions.)

- Decision trees partition the feature space by comparing feature values to thresholds. These comparisons depend only on the **relative order** of the feature values, not their absolute magnitudes or units.
- Many algorithms like k-Nearest Neighbors (k-NN), Support Vector Machines (SVMs), or Linear Regression rely on distance calculations or dot products, which are sensitive to the scale of the features. Decision trees, however, do not compute distances between data points. Instead, they evaluate individual feature thresholds, making them unaffected by the scale of the features.
- Classification trees use measures like **Gini Impurity** or **Entropy** to evaluate the quality of a split. These measures are based on the class distribution in a node and are independent of the feature scales.
- Decision trees do not assume linear relationships between features and the target variable. Features are treated independently, and splits are determined purely based on information gain (or similar criteria).

The classification tree method does not require feature scaling because:
It uses thresholds for splitting based on relative order; It evaluates splits using impurity measures that are unaffected by the scale.
It does not involve distance metrics or gradient-based optimization.
This property makes decision trees and tree-based ensemble methods highly practical for datasets with features on different scales or units.

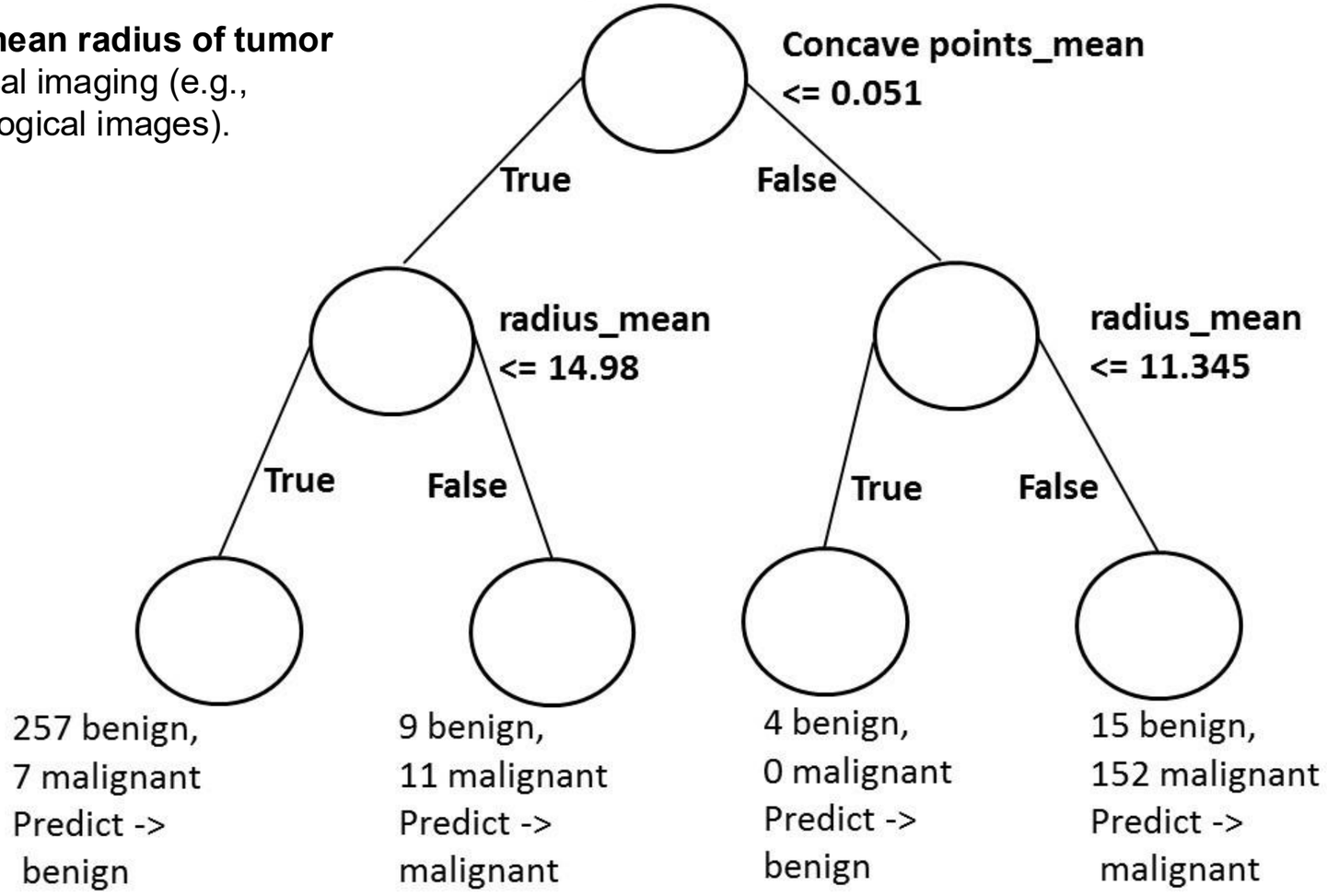
Breast Cancer Dataset in 2D



Decision-tree Diagram

Concave points in the context of breast cancer typically refer to **concave-shaped indentations** along the boundary of a tumor or lesion. These points are a geometric feature extracted from imaging data (e.g., mammograms or histopathological images) and are often used in computational analysis for distinguishing between benign and malignant tumors.

Radius mean refers to the **mean radius of tumor nuclei** computed from medical imaging (e.g., mammograms or histopathological images).



Classification-tree in scikit-learn

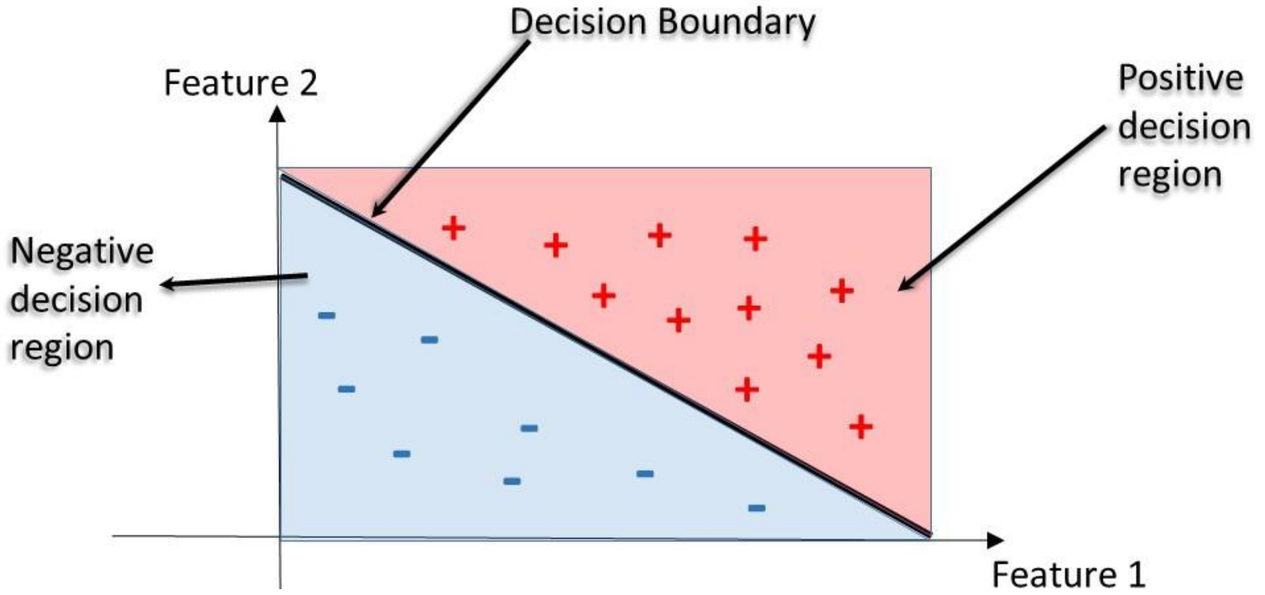
```
# Fit dt to the training set
dt.fit(X_train,y_train)


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

```
0.9035087719298245
```

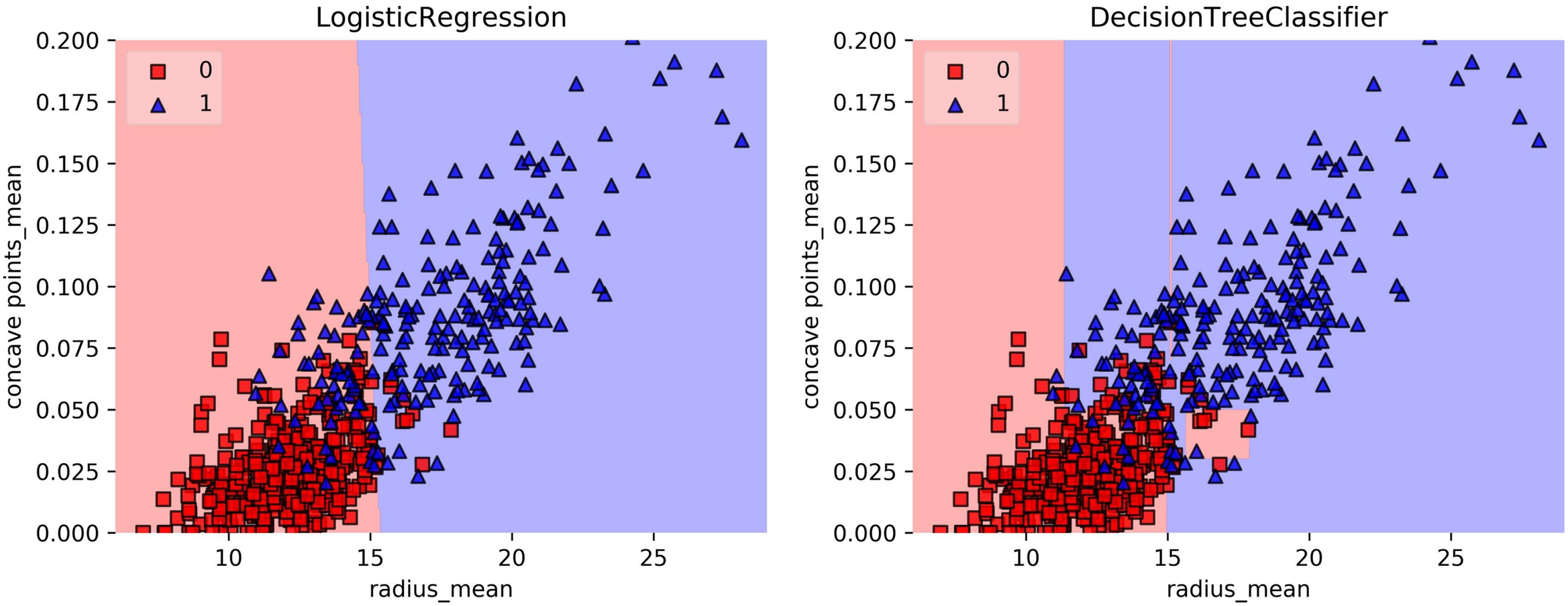
Decision Regions

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



CART stands for **Classification and Regression Trees**, a machine learning algorithm used for both classification and regression tasks.

Classification-Tree Learning

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Building Blocks of a Decision-Tree

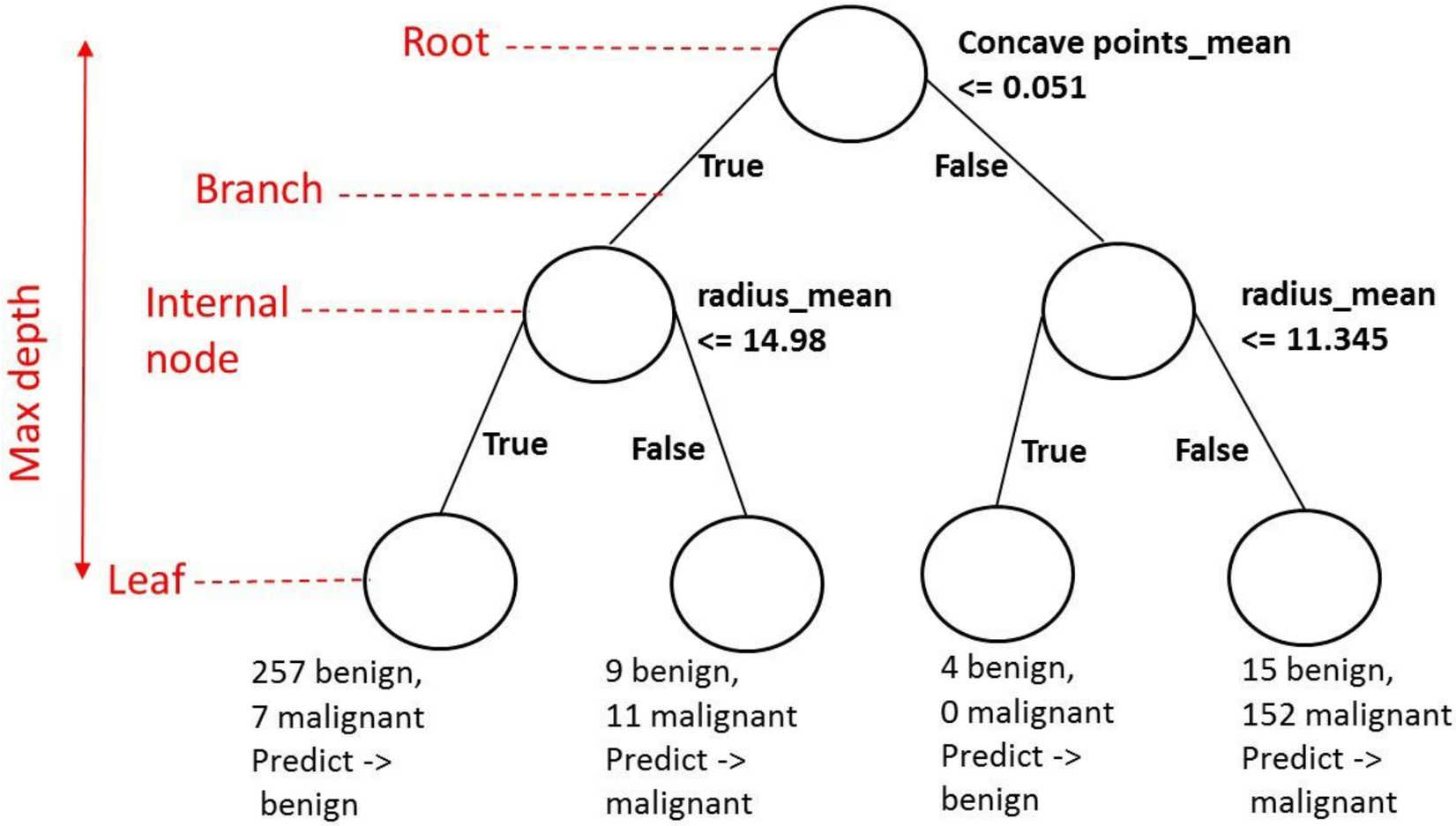
- **Decision-Tree**: data structure consisting of a hierarchy of nodes.
- **Node**: question or prediction.

Building Blocks of a Decision-Tree

Three kinds of nodes:

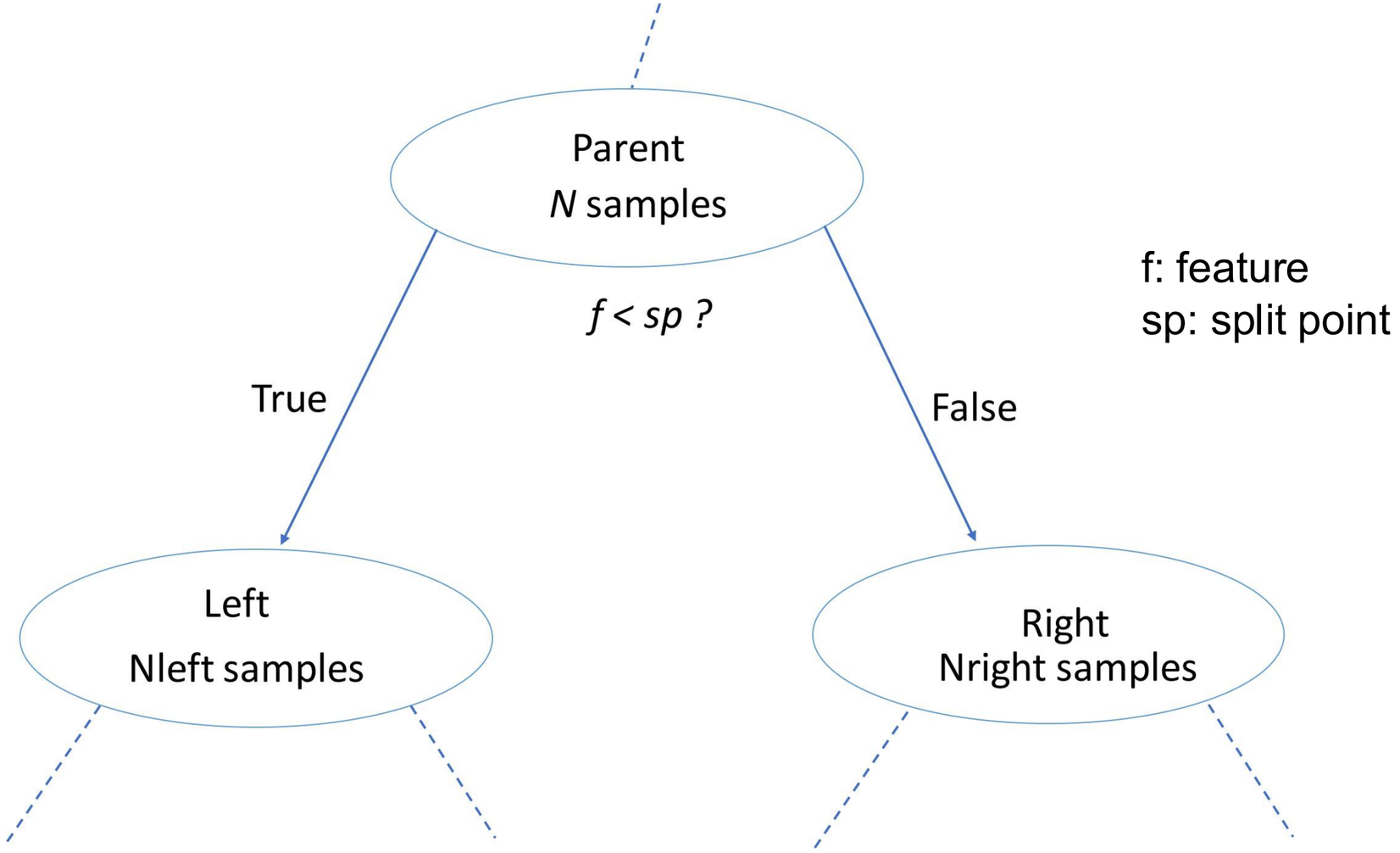
- **Root**: *no* parent node, question giving rise to *two* children nodes.
- **Internal node**: *one* parent node, question giving rise to *two* children nodes.
- **Leaf**: *one* parent node, *no* children nodes --> *prediction*.

Prediction


Root
Concave points_mean
<= 0.051
True
False
Branch
Internal
node
radius_mean
<= 14.98
radius_mean
<= 11.345
True
False
True
False
Max depth
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 is a key metric used in decision tree algorithms for selecting the best feature to split the data at each step. It measures the **reduction in impurity** achieved by a split, quantifying how well a particular split separates the data into homogeneous groups.



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. …

The **Gini Index**, also known as **Gini Impurity**, is a metric used in decision to evaluate the quality of a split. It measures the level of impurity or disorder in a dataset, with the goal of minimizing impurity in child nodes after a split.

Key Properties of the Gini Index

1.Range:

1. The Gini Index ranges from 0 to 1.
2. Gini = 0: Pure node (all samples belong to one class).
3. Gini = 1: Maximum impurity (all classes are equally represented).

2.Interpretation:

1. Lower Gini values indicate purer nodes.
2. Splits are chosen to minimize the Gini Index, resulting in more homogeneous child nodes.

Classification-Tree Learning

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

```
# 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
```

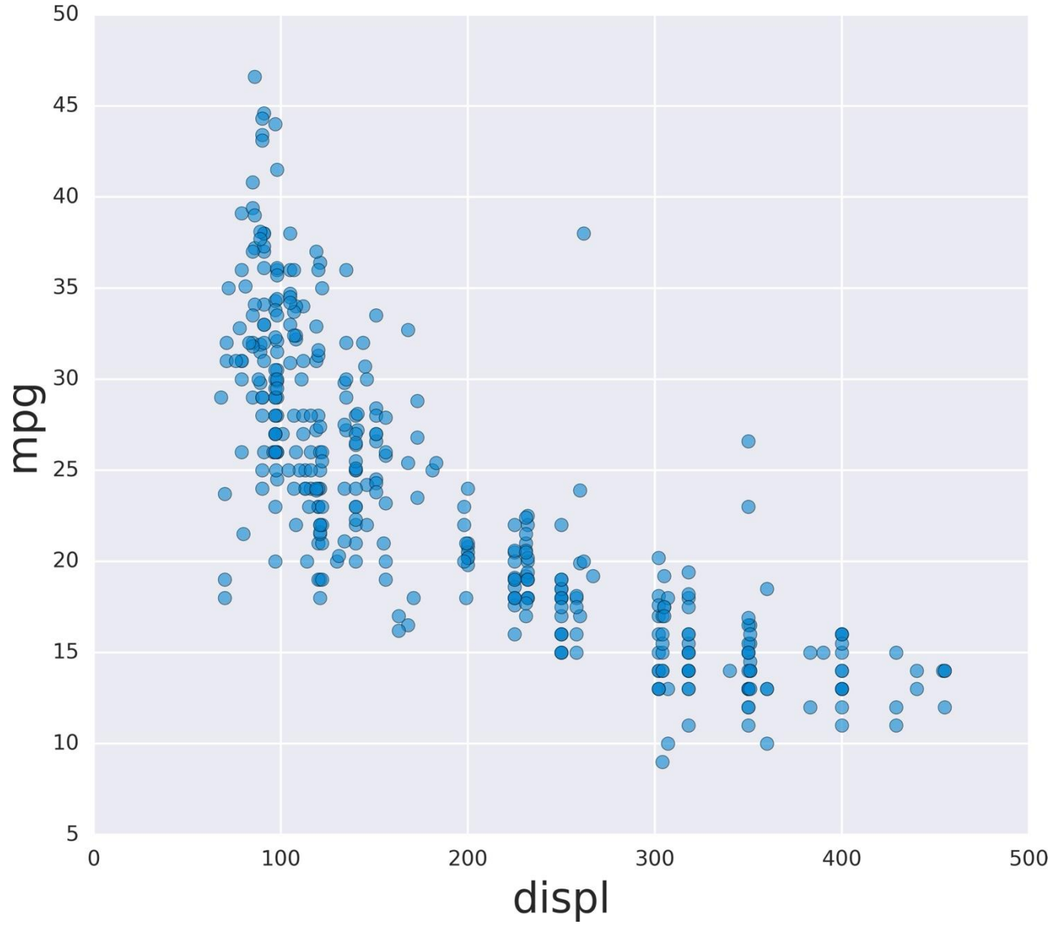
Decision-Tree for Regression

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

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

```
# 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
```

Root Mean Squared Error (RMSE) is a commonly used metric to evaluate the performance of regression models, including **Regression Trees**. It measures the average magnitude of prediction errors by taking the square root of the mean of squared differences between predicted and actual values.

- RMSE quantifies how far off the predicted values are from the actual values.
- Lower RMSE indicates better model performance.

Information Criterion for Regression-Tree

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

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

As the tree grows, the RSS decreases because the model captures more details of the data (fit improves).
However, the complexity penalty increases due to the addition of more parameters (splits or nodes).

Prediction

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

Linear Regression vs. Regression-Tree