DATA SCIENCE 2 - DATA & A.I.

3 V MACHINE LEARNING

4-7 ML TECHNIQUES

Karel de Grote Hogeschool

DATA SCIENCE 2
DATA & A.I. 3

I

PYTHON BASICS

Python for data science

II

WORKING WITH ARRAYS

Numpy

III

DATA ENGINEERING

pandas

IV

DATA VISUALISATION

Matplotlib

V

MACHINE LEARNING

Automatically find patterns

V

MACHINE LEARNING

scikit-learn

MACHINE LEARNING

01 **WHAT IS MACHINE LEARNING**
Automatically find patterns

02 **INTRODUCING SCIKIT-LEARN**
Machine learning with Python

03 **HYPERPARAMETERS AND CROSS VALIDATION**
Holdout samples and cross-validation

04 **REGRESSION**
Best fitting line

05 **DECISION TREES**
Best separating lines

06 **K-MEANS CLUSTERING**
Object grouping

07 **ASSOCIATION RULES**
Frequent itemsets

08 **ARTIFICIAL NEURAL NETWORK**
Imitate the human brain

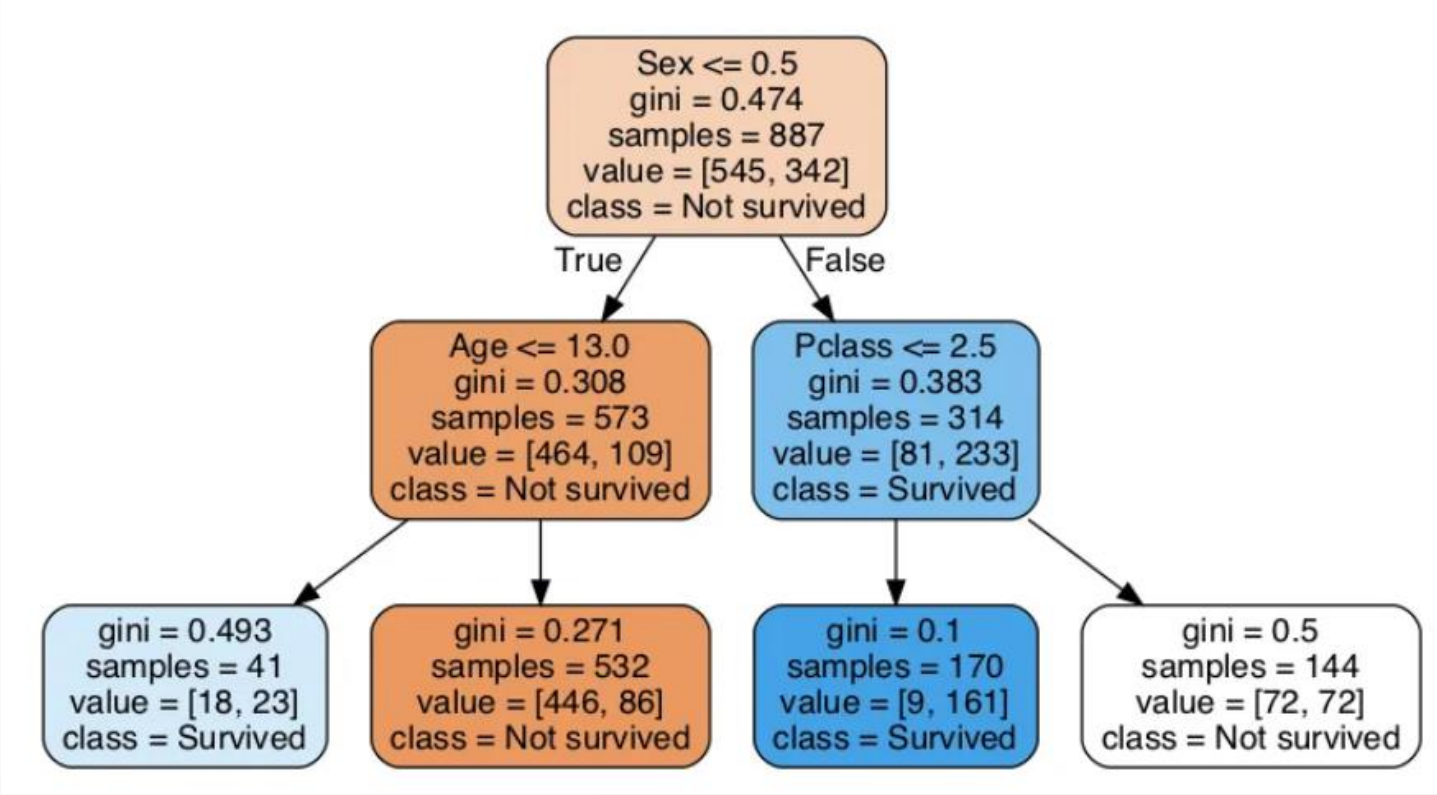
SUPERVISED LEARNING: DECISION TREES

Decision Trees and Random Forests

DECISION TREES

Decision trees:

A supervised learning algorithm used for **classification** and **regression** tasks creating a tree-like model of decisions based on input features



DECISION TREES WITH SCIKIT-LEARN

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='gini')
```

Hyperparameters:

criterion

- **Type**: string, default='gini'
- **Description**: This function measures the quality of a split. Supported criteria are 'gini' for the Gini impurity and 'entropy' for the information gain.
- **Usage**: Choose 'gini' for the Gini impurity (default) or 'entropy' for information gain when building the decision tree.

max_depth

- **Type**: int, default=None
- **Description**: The maximum depth of the tree. If None, nodes are expanded until all leaves are pure or contain fewer samples than the min_samples_split.
- **Usage**: Set this to limit the depth of the tree and avoid overfitting.

min_samples_split

- **Type**: int or float, default=2
- **Description**: The minimum number of samples required to split an internal node. If an integer, it's the minimum number. If a float, it's the fraction of the total number of samples.
- **Usage**: Increase this value to reduce overfitting by making the tree more generalized.

DECISION TREES WITH SCIKIT-LEARN

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='gini')
```

Hyperparameters:

min_samples_leaf

- **Type**: int or float, default=1
- **Description**: The minimum number of samples required to be at a leaf node. A smaller value allows smaller leaves and may lead to overfitting.
- **Usage**: A higher value can make the model more conservative and less prone to overfitting.

max_features

- **Type**: int, float, string or None, default=None
- **Description**: The number of features to consider when looking for the best split. If None, all features are considered.
- **Usage**: Restricting the number of features can lead to a more generalized model.

DECISION TREES WITH SCIKIT-LEARN

```
# DATA PREPARATION
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split

iris = sns.load_dataset('iris')

# Target feature to predict: Pandas Series
y = iris['species']
# Predictors: Pandas DataFrame
X = iris[['sepal_width','sepal_length','petal_width','petal_width']]

# SPLIT in TRAIN and TEST dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, stratify=y)
```

DECISION TREES WITH SCIKIT-LEARN

```
# MODEL SELECTION AND HYPERPARAMETER SELECTION (MODEL SPECIFIC)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth=1)

# List all selected hyperparameters
print(model.get_params(deep=True))

# DERIVE MODEL FROM LABELED DATA (TRAIN MODEL/FIT MODEL)
model.fit(X_train, y_train)

# DISPLAY MODEL (MODEL SPECIFIC)
print(f"Model classes: {model.classes_}")
from sklearn.tree import plot_tree
plot_tree(model)
```

DECISION TREES WITH SCIKIT-LEARN

```
# VALIDATE MODEL USING LABELED TEST DATA

# Score model, used metric is model dependent, for decision tress: Accuracy
r2 = model.score(X_test, y_test)
print(f'ACC: {r2:.3f}')

# Other metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
                            classification_report

# Predict target feature for the labeled test data
y_test_pred = model.predict(X_test)
acc = accuracy_score(y_true=y_test, y_pred=y_test_pred)
prec = precision_score(y_true=y_test, y_pred=y_test_pred, average='weighted')
rec = recall_score(y_true=y_test, y_pred=y_test_pred, average='weighted')
f1 = f1_score(y_true=y_test, y_pred=y_test_pred, average='weighted')
# Multiclass classification -> precision, recall and F1 are calculated by class and averaged.
print(f'ACC : {acc:.3f} - PREC : {prec:.3f} - REC : {rec:.3f} - F1 : {f1:.3f}')

classification_report(y_true=y_test, y_pred=y_test_pred)
```

DECISION TREES WITH SCIKIT-LEARN

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_test_pred, labels=model.classes_)
print(conf_matrix)

# Plot confusion matrix (nicer output in Jupyter)
ConfusionMatrixDisplay.from_predictions(y_true=y_test, y_pred=y_test_pred,
                                        display_labels=model.classes_)
```

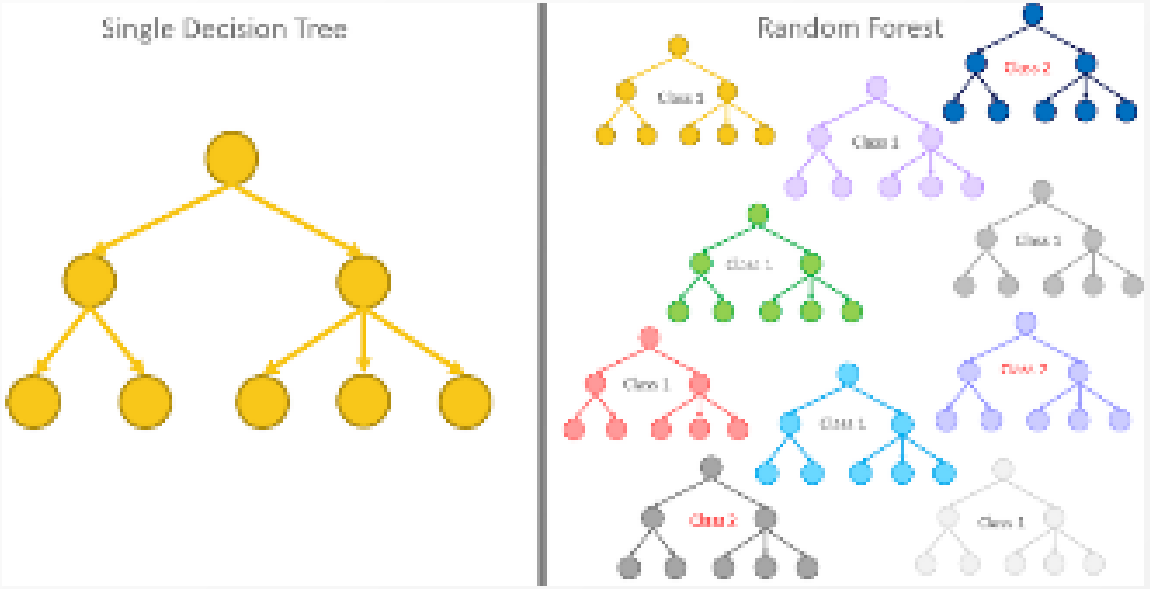
DECISION TREES WITH SCIKIT-LEARN

```
# APPLY MODEL ON NEW DATA
X_pred = …. (new feature data to predict the target feature for)
y_pred = model.predict(X_pred)
```

RANDOM FORESTS

Random forests:

An ensemble method that builds multiple decision trees and aggregates their results for improved accuracy in classification and regression tasks.



RANDOM FORESTS WITH SCIKIT-LEARN

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100)
```

Hyperparameters:

n_estimators

- **Type**: int, default=100
- **Description**: The number of trees in the forest. More trees usually lead to better performance but increase computation time.
- **Usage**: Use higher values (e.g., 500, 1000) for larger datasets to improve performance.

max_depth

- **Type**: int, default=None
- **Description**: The maximum depth of each tree in the forest. If None, nodes are expanded until all leaves are pure.
- **Usage**: Restrict this to prevent overfitting, especially when building deep trees.

max_features

- **Type**: int, float, string or None, default='auto'
- **Description**: The number of features to consider when looking for the best split. 'auto' uses the square root of the number of features.
- **Usage**: Try different values such as 'sqrt' (square root) or 'log2' to tune the model for better performance.

min_samples_split

- **Type**: int or float, default=2
- **Description**: The minimum number of samples required to split an internal node in each tree.
- **Usage**: Adjust this to balance between underfitting and overfitting.

RANDOM FORESTS WITH SCIKIT-LEARN

```
# Data preparation
iris = sns.load_dataset('iris')

# Target feature to predict: Pandas Series
y = iris['species']
# Predictors: Pandas DataFrame
X = iris[['sepal_width','sepal_length','petal_width','petal_width']]

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Initialize and train the Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
model.fit(X_train, y_train)
```

RANDOM FORESTS WITH SCIKIT-LEARN

```
# Make predictions
y_test_pred = model.predict(X_test)

# Evaluate the model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
accuracy = accuracy_score(y_test, y_test_pred)
conf_matrix = confusion_matrix(y_test, y_test_pred)
class_report = classification_report(y_test, y_test_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```