Data Analysis Lab

Ensemble Methods

Decision trees

Fátima Leal

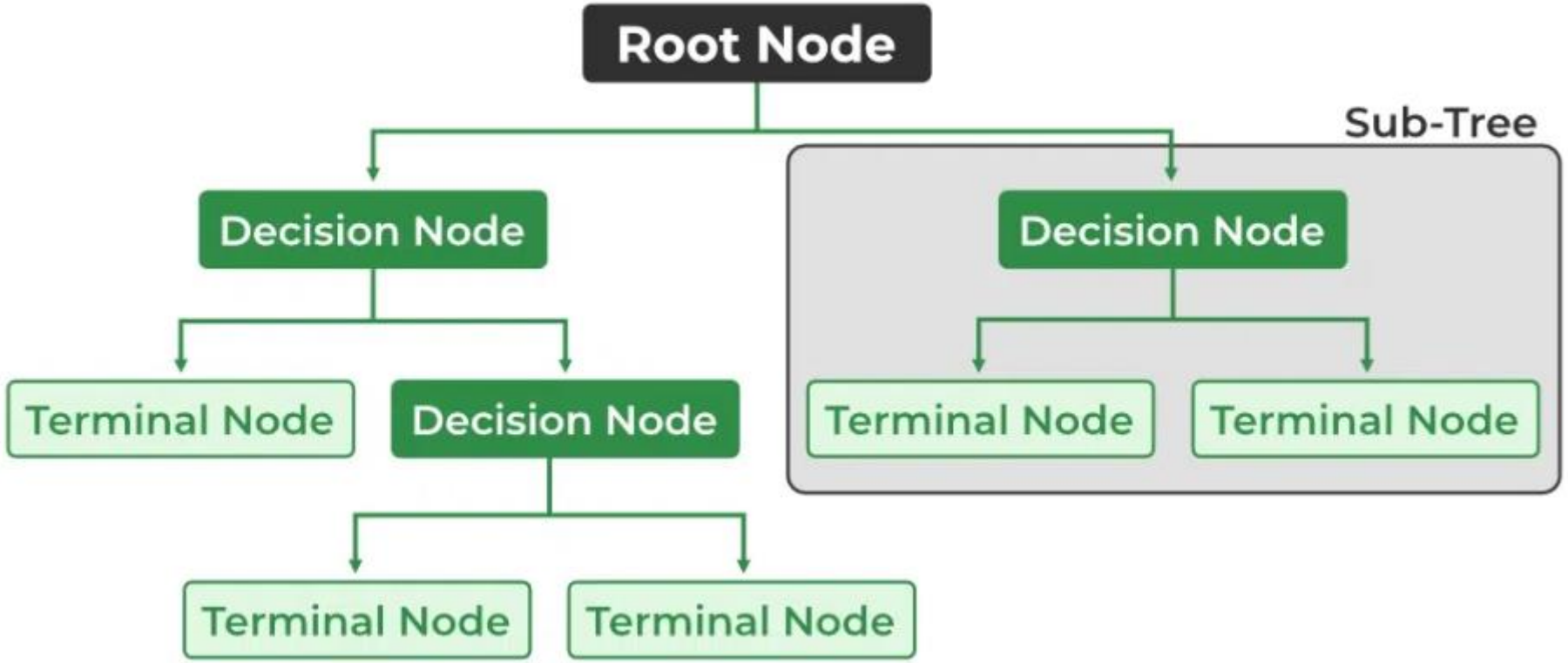
DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

Outline

- Ensemble Methods
- Decision tree classification
- Decision tree Regression
- Exercises

Binary Decision tree

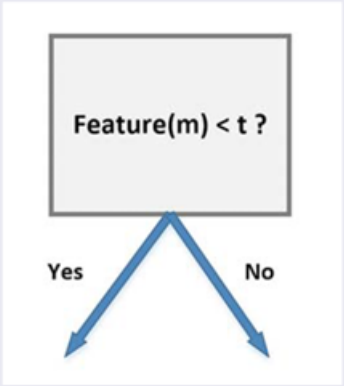


Binary Decision tree

- A binary decision tree is a structure based on a sequential decision process.
- Starting from the root, a feature is evaluated and one of the two branches is selected. This procedure is repeated until a final leaf is reached, which normally represents the classification target we're looking for.
- Decision trees can work efficiently with unnormalized datasets because their internal structure is not influenced by the values assumed by each feature.

Binary Decision Tree

- Let's consider an input dataset X: $X = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_n\}$ where $\bar{x}_i \in \mathrm{R}^m$
- Every vector is made up of m features, so each of them can be a good candidate to create a node based on the (feature, threshold) tuple:



- According to the feature and the threshold, the structure of the tree will change. Intuitively, we should pick the feature that best separates our data in other words, a perfect separating feature will be present only in a node and the two subsequent branches won't be based on it anymore.

Binary Decision Tree

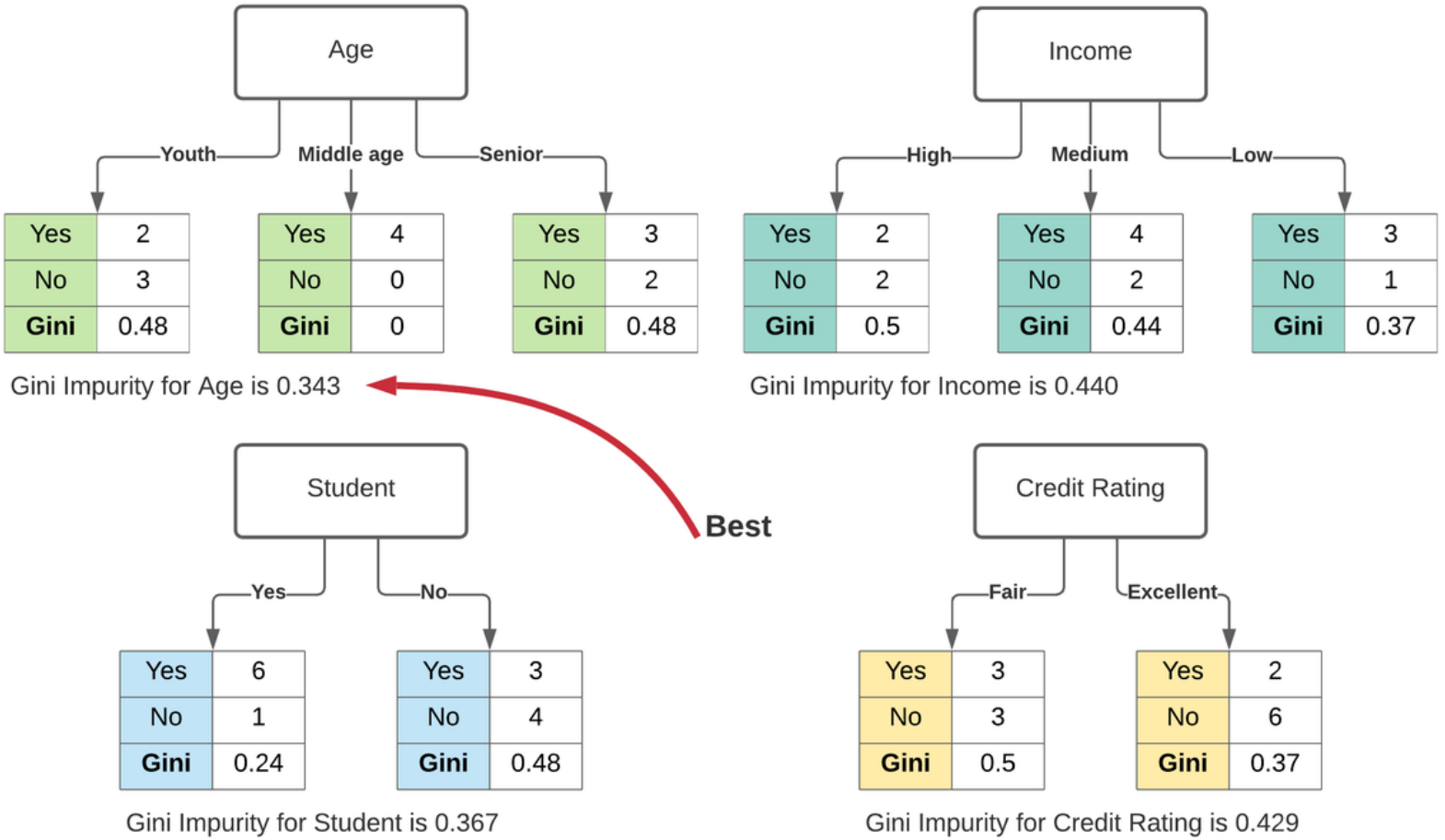
- The two resulting sets are now pure according to the feature, and this can be enough for our task.
- The choice of the best threshold is a fundamental element because it determines the structure of the tree and, therefore, its performance.
- The goal is to reduce the residual impurity in the least number of splits so as to have a very short decision path between the sample data and the classification result.
- We can also define a total impurity measure by considering the two branches: $I(D, \sigma) = \frac{N_{left}}{N_D} I(D_{left}) + \frac{N_{right}}{N_D} I(D_{right})$
- Here, D is the whole dataset at the selected node, D_{left} and D_{right} are the resulting subsets, and the I are impurity measures.

Impurity measures

- **Entropy** is the measure of the degree of randomness or uncertainty in the dataset. In the case of classifications, It measures the randomness based on the distribution of class labels in the dataset
  - 0 when the dataset is completely homogeneous
  - when the dataset is equally divided between multiple classes, the entropy is at its maximum value

- **Gini** is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset. In this case, we want to have a Gini index score as low as possible

Impurity measures



Feature importance

- Decision trees offer a different approach based on the impurity reduction determined by every single feature. In particular, considering a feature x_i, its importance can be determined as: $Importance(x_i) = \sum \frac{N_k}{N} \Delta I_{x_i}$
- The sum is extended to all nodes where x_i is used, and N_k is the number of samples reaching the node k. Therefore, the importance is a weighted sum of all impurity reductions computed considering only the nodes where the feature is used to split them. If the Gini impurity index is adopted, this measure is also called Gini importance

Decision Tree

- Decision Tree Classifier
- Decision Tree Regressor

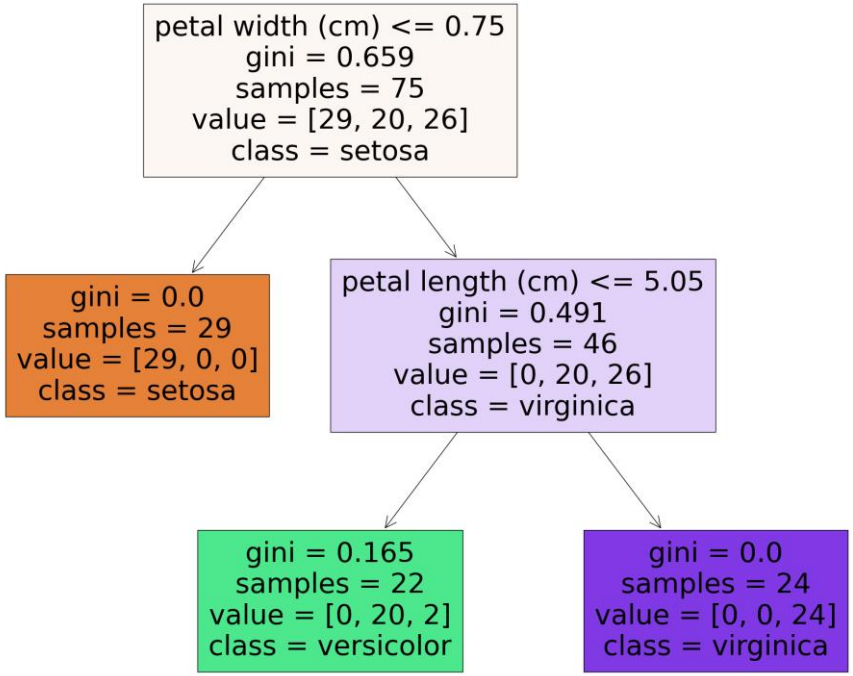
Classification and Regression Trees

```
from sklearn import datasets, tree
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

iris=datasets.load_iris()
X_train,X_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.5,random_st
ate=0)
clf=tree.DecisionTreeClassifier(criterion='gini',
                                max_depth=2)
clf=clf.fit(X_train,y_train)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
                   feature_names=iris.feature_names,
                   class_names=iris.target_names,
                   filled=True)
fig.savefig("decistion_tree.png")
plt.show()
```

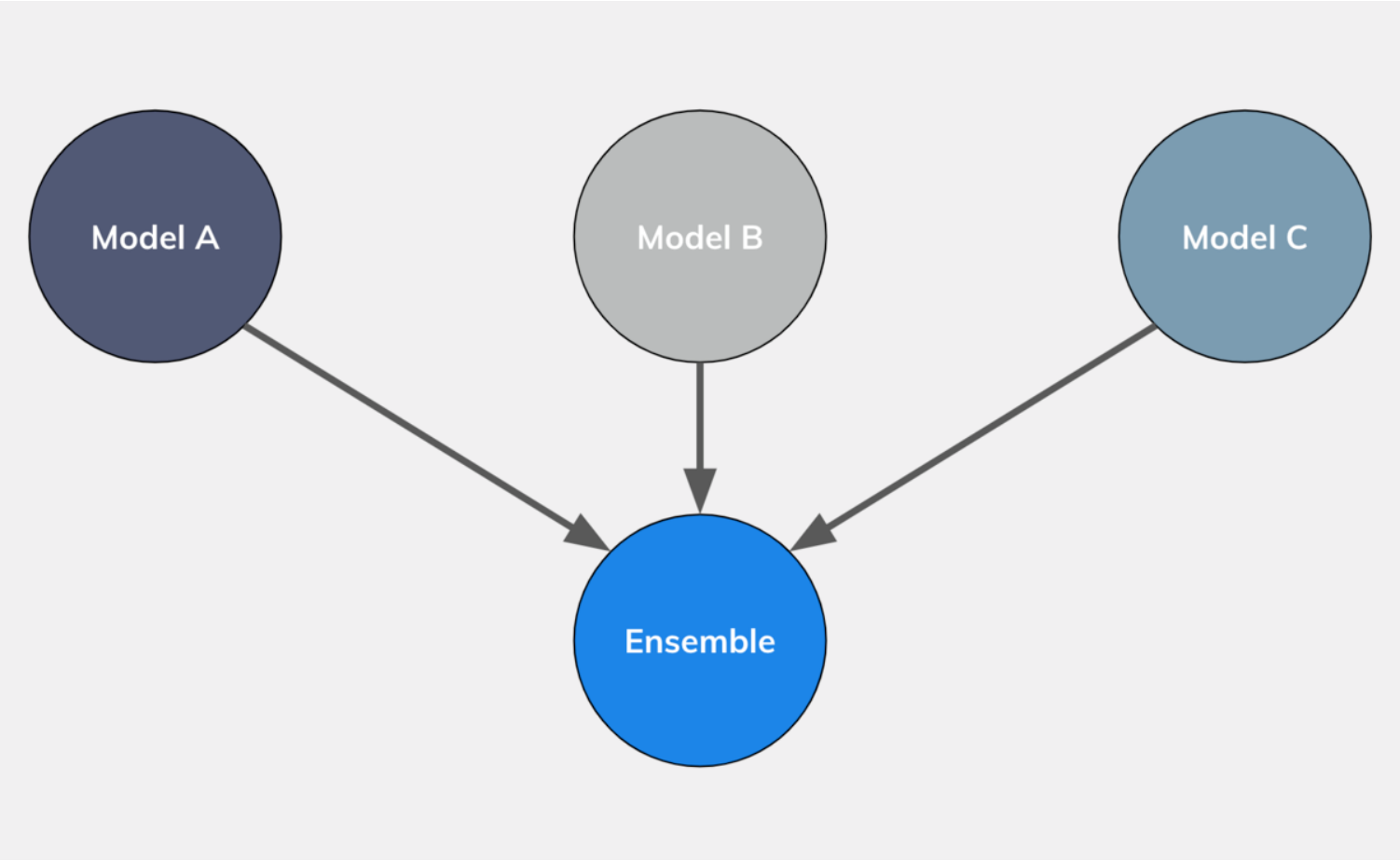
Classification and Regreesion Trees



Ensemble methods

- Ensemble methods are a powerful alternative to complex algorithms because they try to exploit the statistical concept of majority vote.
- Many weak learners can be trained to capture different elements and make their own predictions, which are not globally optimal, but using a sufficient number of elements, it's statistically probable that a majority will evaluate correctly.
- In particular, we're going to discuss random forests of decision trees and some boosting methods that are slightly different algorithms that can optimize the learning process by focusing on misclassified samples or by continuously minimizing a target loss function.

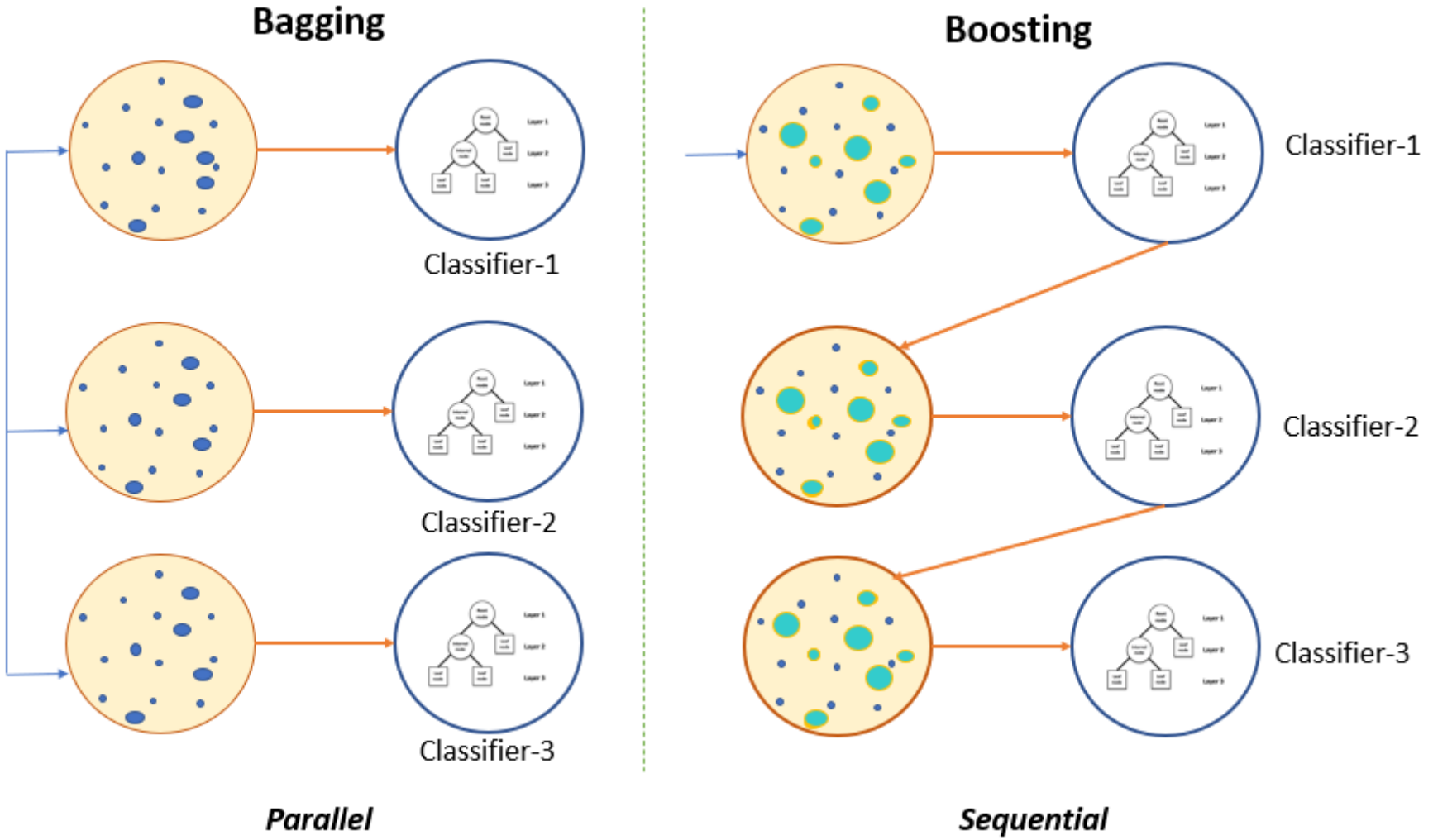
Ensemble methods



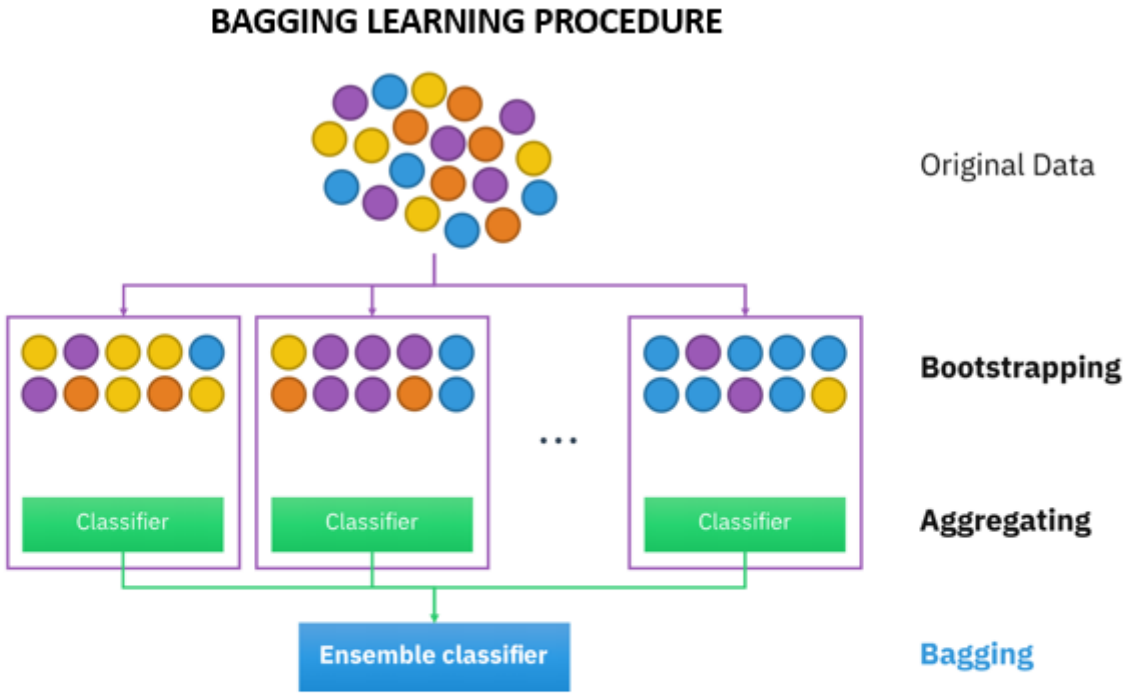
Ensemble methods

- Bagging involves fitting many decision trees on different samples of the same dataset and averaging the predictions.
- Stacking involves fitting many different models types on the same data and using another model to learn how to best combine the predictions.
- Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.

Ensemble methods



Ensemble methods

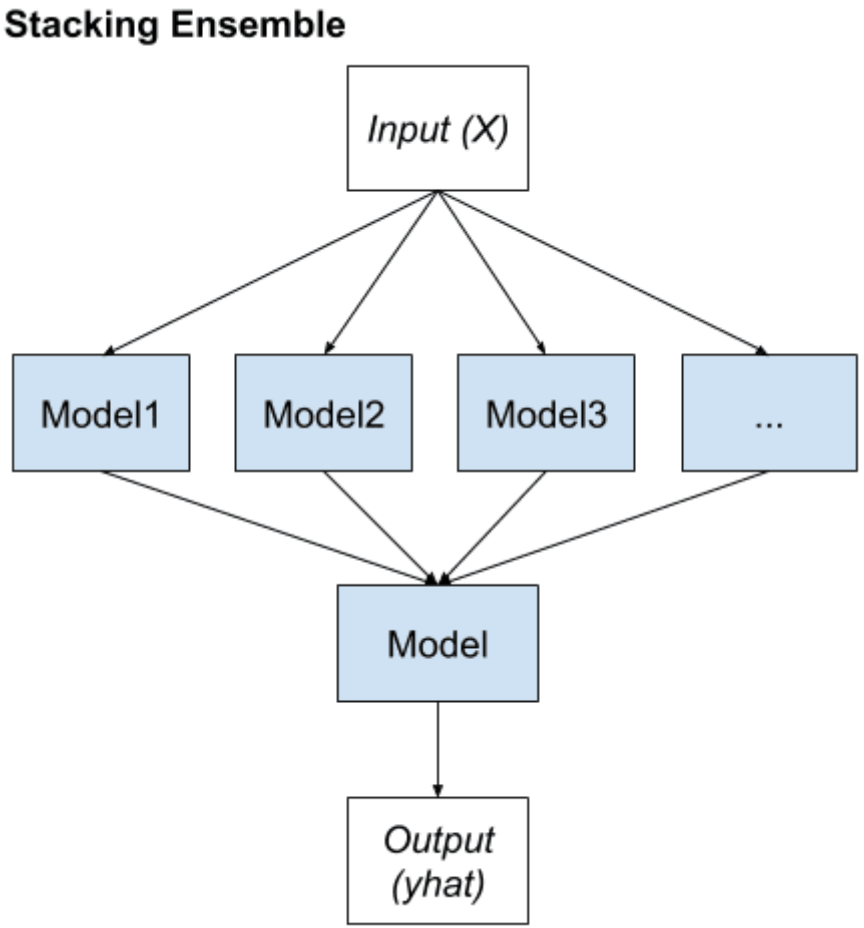


$$f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$$

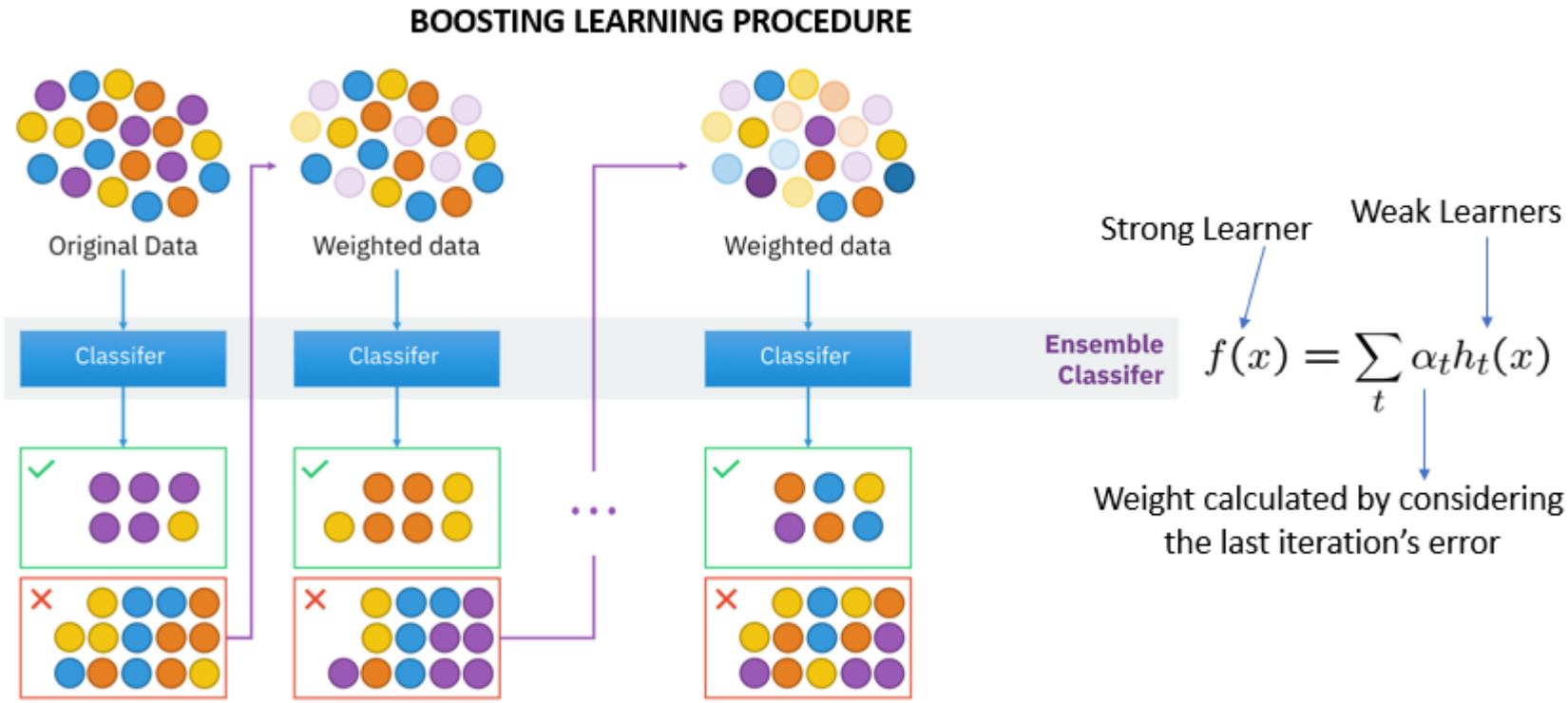
Weak Learners

Generates Bootstrapping Sets

Ensemble methods

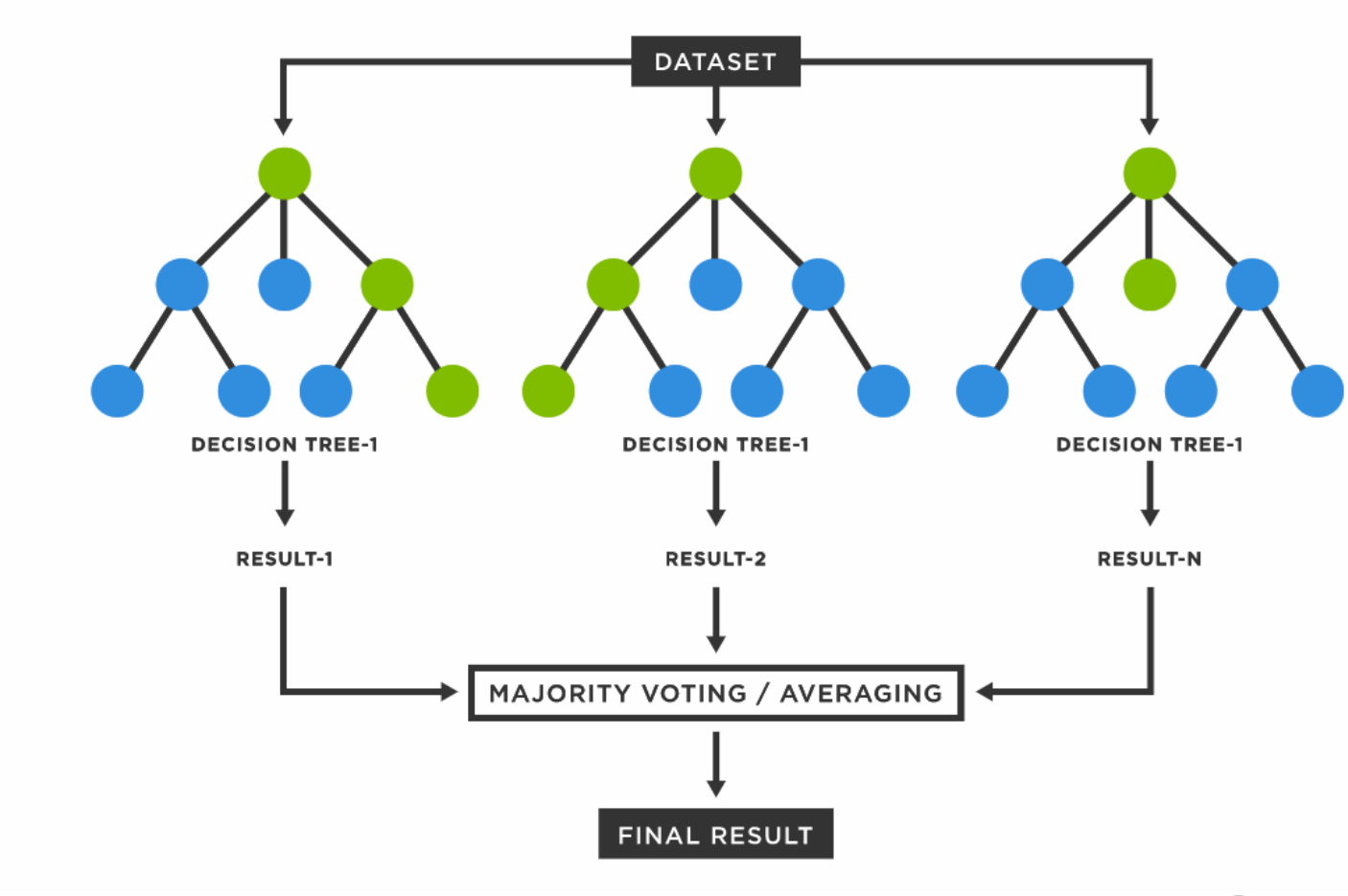


Ensemble methods



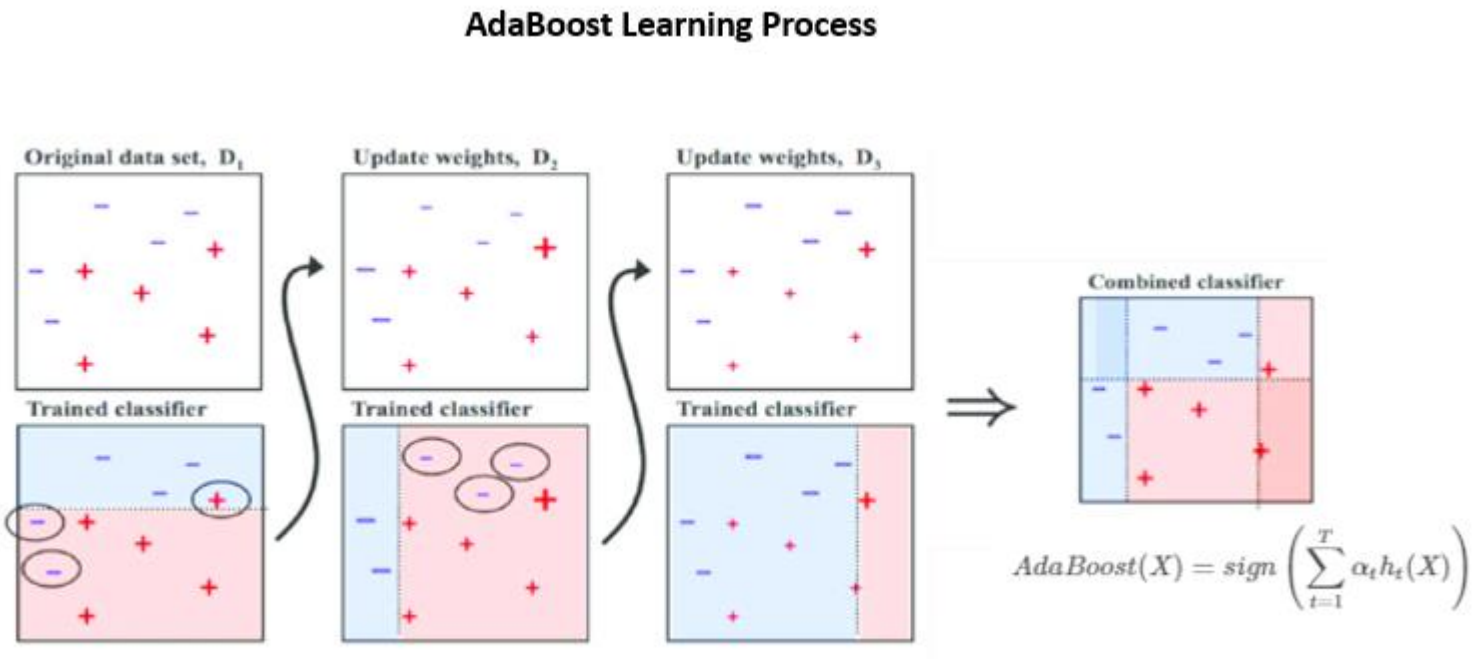
Ensemble methods examples

- Bagging methods – Random forest



Ensemble methods examples

- Boosting methods - Gradient Descent Boosting, AdaBoost, and XGboost



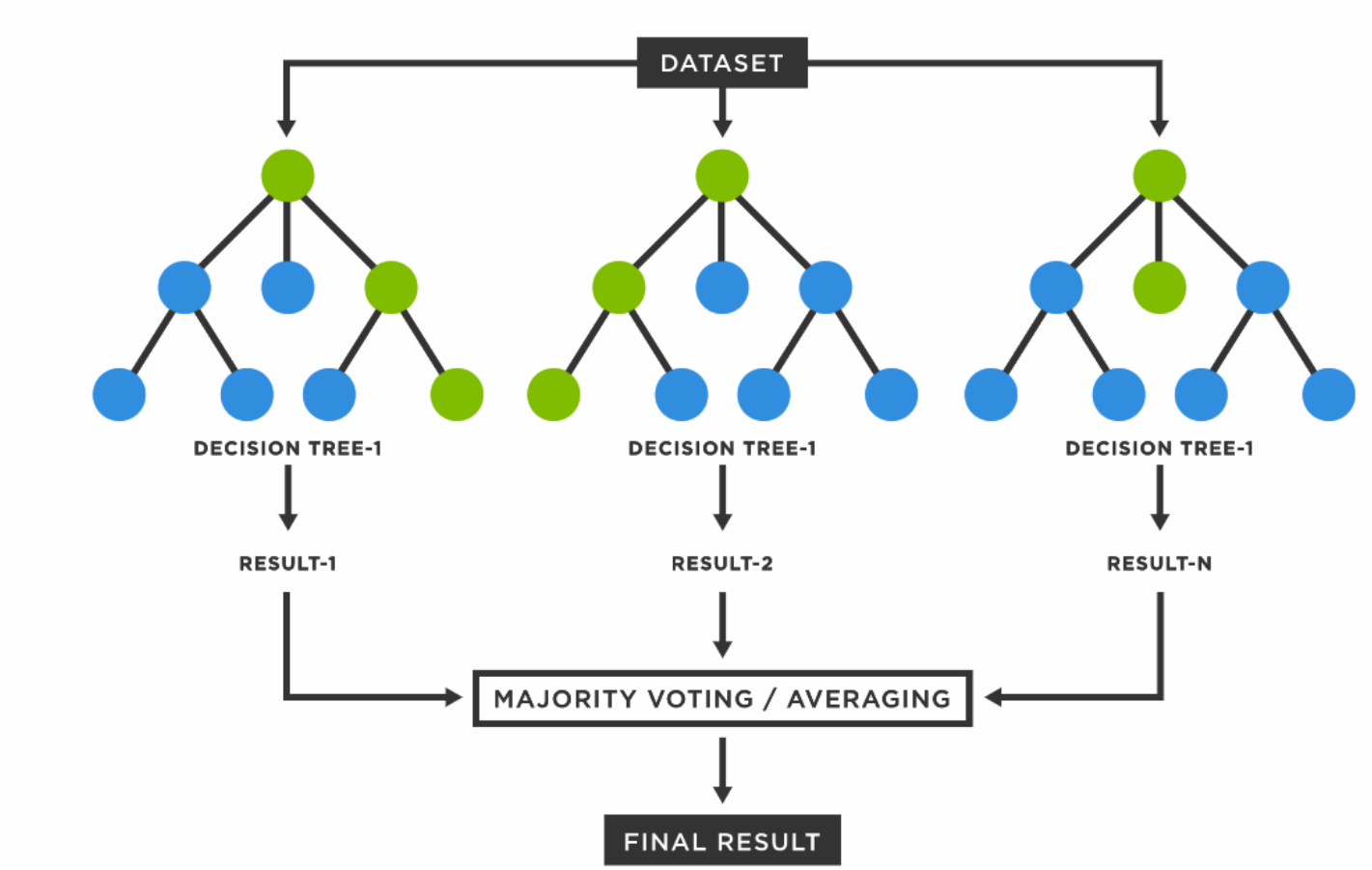
Random Forest

- A random forest is a set of decision trees built on random samples with a different policy for splitting a node: Instead of looking for the best choice, in such a model, a random subset of features (for each tree) is used, trying to find the threshold that best separates the data.
- As a result, there will be many trees trained in a weaker way and each of them will produce a different prediction.
- There are two ways to interpret these results; the more common approach is based on a majority vote.

Random Forest

- However, scikit-learn implements an algorithm based on averaging the results, which yields very accurate predictions.
- Even if they are theoretically different, the probabilistic average of a trained random forest cannot be very different from the majority of predictions; therefore the two methods often drive to comparable results.

Random Forest



Random Forest

- Random Forest Classifier
- Random Forest Regressor

Random Forest Regressor example

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split
import pandas as pd

houses = pd.read_csv("USA_Housing.csv")

houses_target = houses['Price']
houses_data = houses.iloc[:,:5]
print(houses_data)

X_train, X_test, Y_train, Y_test = train_test_split(houses_data.values, houses_target.values, test_size=0.3)

rf=RandomForestRegressor(max_depth=10,random_state=0)
clf=rf.fit(X_train,Y_train)

print("R2={:.2f}".format(r2_score(Y_test,clf.predict(X_test))))
print("MAE={:.2f}".format(mean_absolute_error(Y_test, clf.predict(X_test))))
```

Random Forest Classifier example

```
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

iris=datasets.load_iris()
X_train,X_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.5,random_st
ate=0)
rf=RandomForestClassifier(max_depth=10,random_state=0)
clf=rf.fit(X_train,y_train)
y_pred=clf.predict(X_test)

print(clf.score(X_test,y_test))
print(confusion_matrix(y_test,y_pred))
```

Exercises

- Use the datasets available on Moodle for regression and classification
- Do a exploratory/statistical analysis of the dataset
- Divide the dataset in train and test
- Apply the decision trees for the different tasks
- Obtain the predictions and scores

References

- Bonaccorso, G. (2020). Mastering Machine Learning Algorithms: Expert techniques for implementing popular machine learning algorithms, fine-tuning your models, and understanding how they work. Packt Publishing Ltd.
- Bonaccorso, G. (2018). Machine Learning Algorithms: Popular algorithms for data science and machine learning. Packt Publishing Ltd.
- Lee, W. M. (2019). Python machine learning. John Wiley & Sons.
- Hauck, T. (2014). scikit-learn Cookbook. Packt Publishing Ltd.
- Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). Introduction to linear regression analysis. John Wiley & Sons.
- Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (Vol. 398). John Wiley & Sons.

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.