Decision Trees and Ensemble Methods

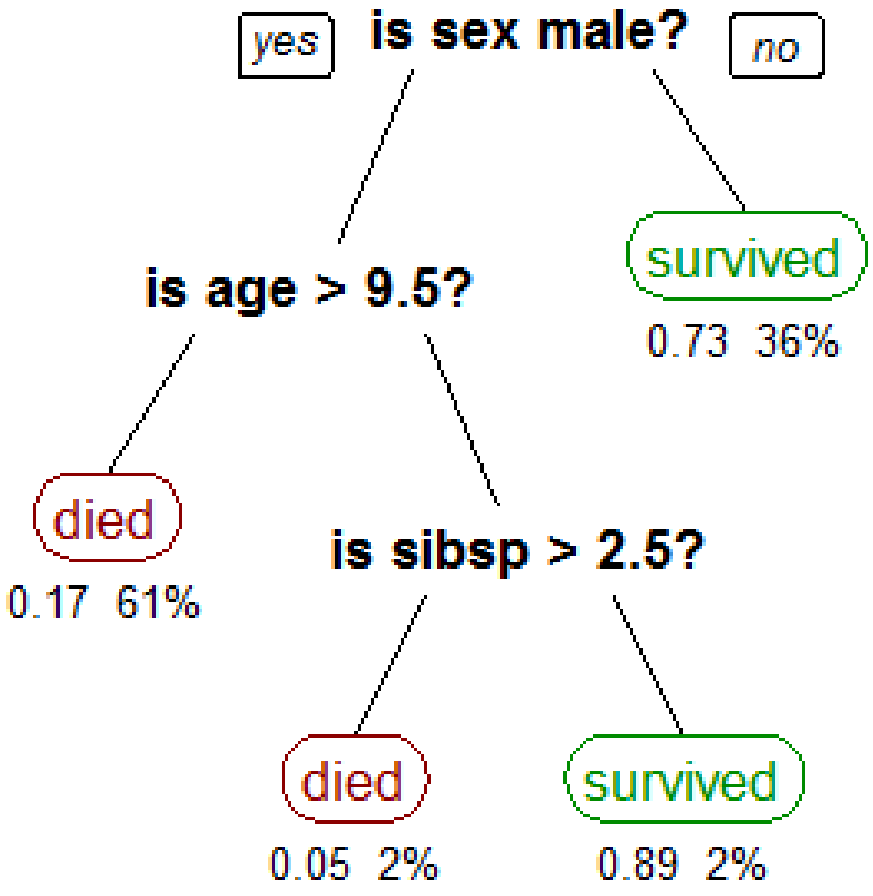
Decision Trees

Decision Trees

- Tool for exploratory data analysis
- Hierarchical representation of data by a sequence of tests allowing to predict a variable

Decision Trees

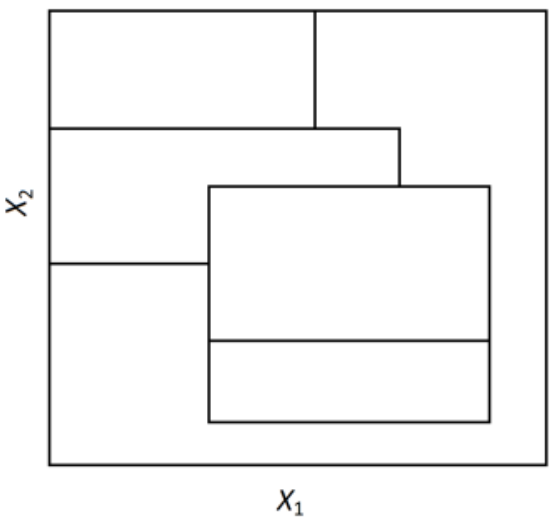
Who should be saved from disaster in Titanic?

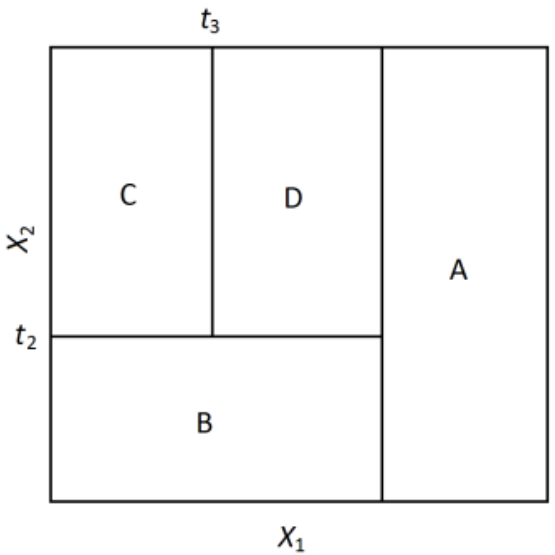


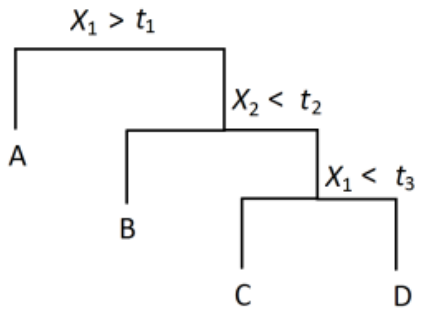
Decision Trees
Learning

- Each node correspond to a feature and tests it
- We then generate several leafs at each iteration corresponds to a partition in the space of input variables

Decision Trees



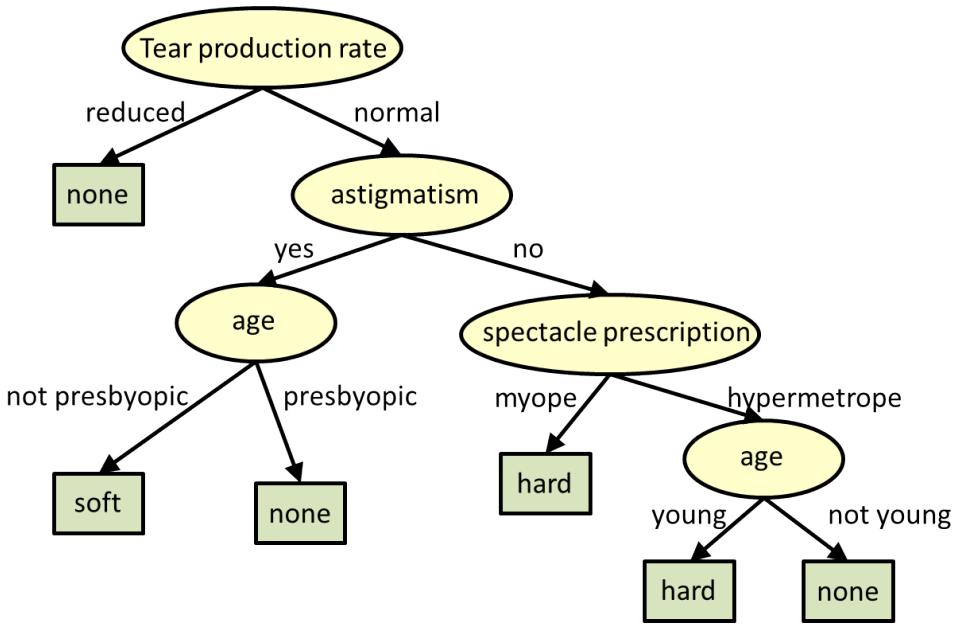




Decision Trees
Learning

- Inputs : points of the `feature space` characterized by numerical or categorical variables
- Target : class (classification) or value (regression)

Decision Trees



Decision Trees

With Python on Iris

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
clf.predict(iris.data[:1, :])
clf.predict_proba(iris.data[:1, :])
```

Decision Trees
Pro and cons

Pros

- White box model
- Few preprocessing
- Low numerical costLe cout dutilisation des arbres est logarithmique
- We can use an inputs cataegorical and numerical variables
- One can deal with outliers and missing values

Decision Trees
Pro and cons

Cons : instability

- Few changes in data yield very different decision trees
- If we change the nodes near the root, the decision tree could be totally different
- Estimators with high variance

Bagging

Training set

- Training set : (xi, yi), $xi \in \mathbb{R}^p$, $y_i \in \mathbb{R}$
- We are given $G(x)$ a prediction model learned on a data sample $z = \{(x_i, y_i)\}$ (e.g. decision tree)

Bagging

- We draw at random in the training set B sample z_i. These samples are the so called bootstrap samples
- For each sample, we estimate $G_i(x)$
- Regression : $G(x) = (1/B) \sum_{i=1}^{B} G_i(x)$
- Classification : $G(x) = \text{majority vote}(G_1(x), \cdots, G_B(x))$

Bagging

```
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
bagging = BaggingClassifier(KNeighborsClassifier(),
max_samples=0.5, max_features=0.5)
```

Bagging

```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.data.shape)
X=digits.data
y=digits.target
```

Bagging

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.90)
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
Z = clf.predict(X_test)
accuracy=clf.score(X_test,y_test)
```

Random Forests

- Assume that the dataset is huge as well as the number of variables
- If we also draw at random the variables : random forest

Random Forests

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.90)
clf = RandomForestClassifier(n_estimators=200)
clf.fit(X_train, y_train)
Z = clf.predict(X_test)
accuracy=clf.score(X_test,y_test)
```