FOUNDATIONS OF STATISTICAL ANALYSIS & MACHINE LEARNING

Amric Trudel
amric.trudel@epita.fr

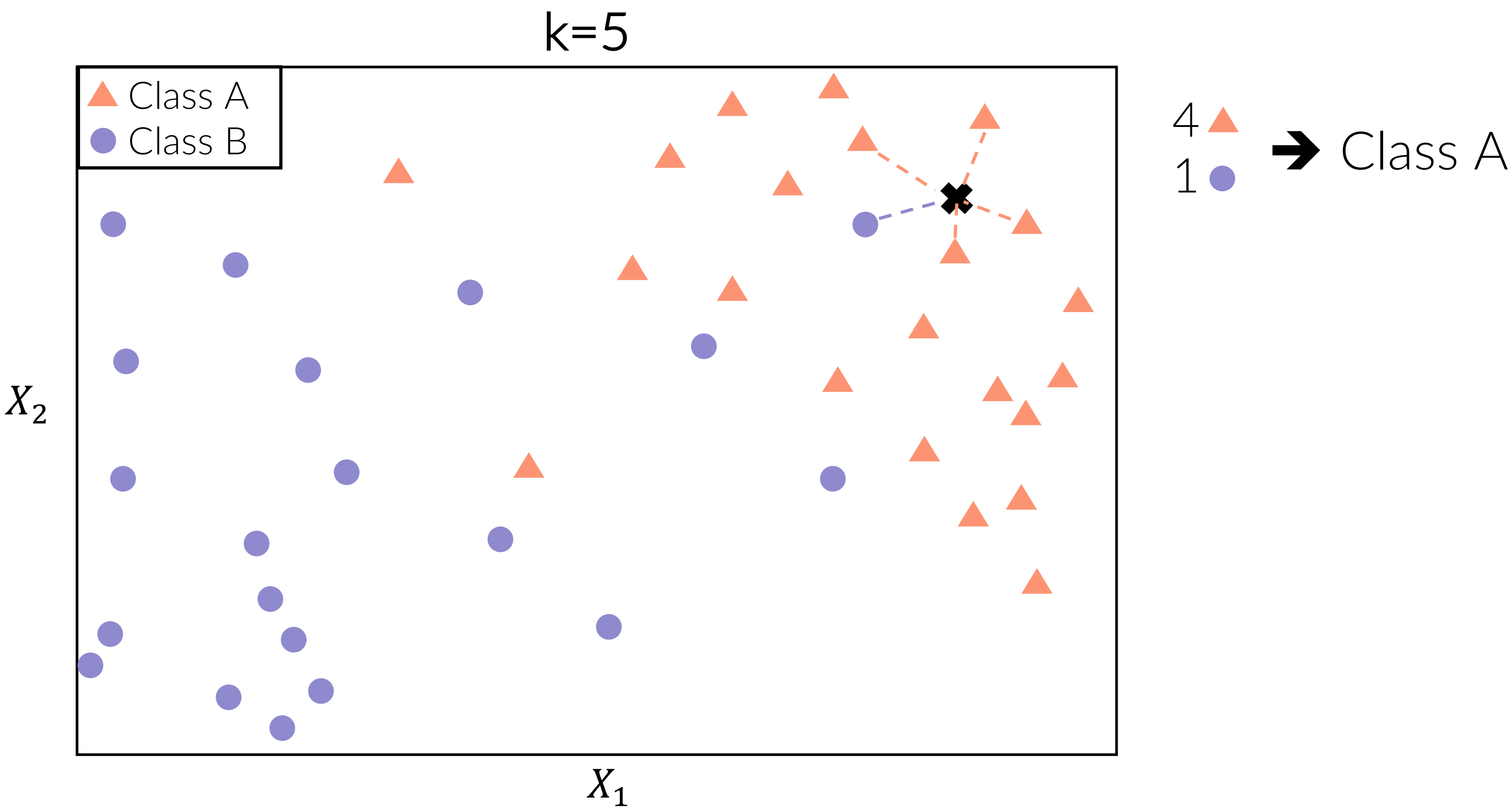
COURSE PROGRAM

Structure

PREPARATION	Data exploration
	Data preprocessing
REGRESSION	Linear regression with one variable
	Multiple and polynomial regression
CLASSIFICATION	Logistic regression
	Classification model assessment
	k-NN, Decision Tree, SVM
CLUSTERING	k-means, hierarchical clustering
DIMENSIONALITY REDUCTION	Principal Components Analysis
ALL NOTIONS	Final assignment

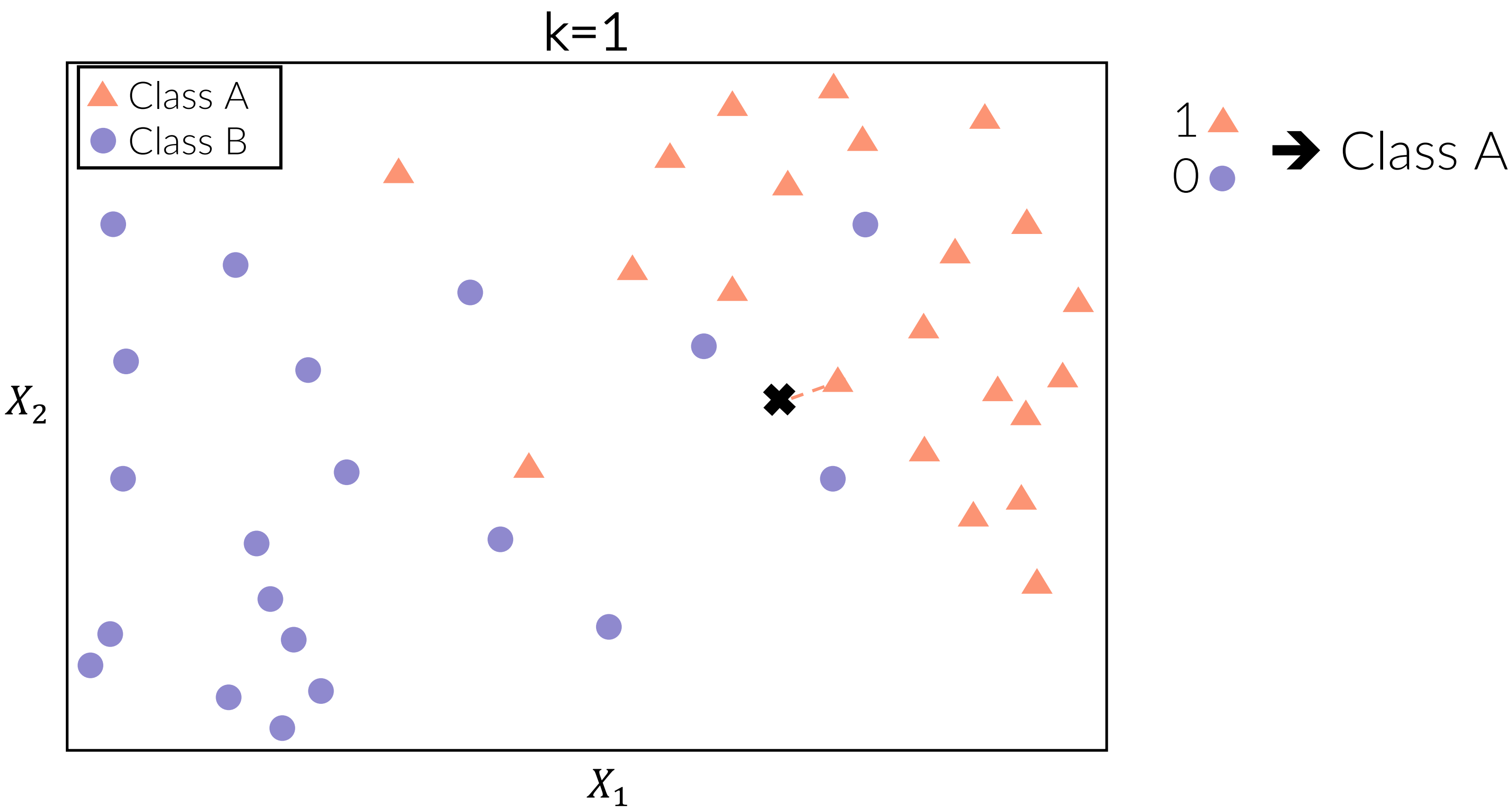
K-NEAREST NEIGHBORS

Illustration



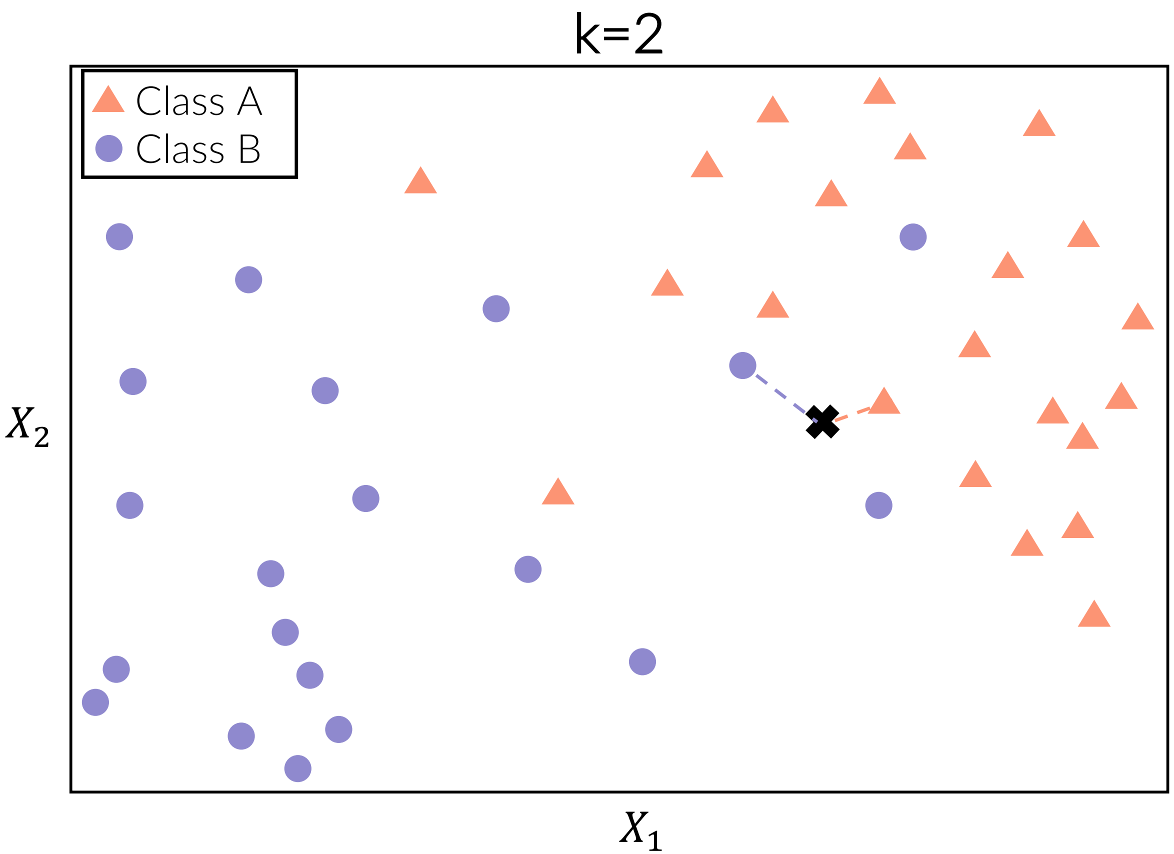
K-NEAREST NEIGHBORS

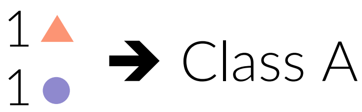
Illustration



K-NEAREST NEIGHBORS

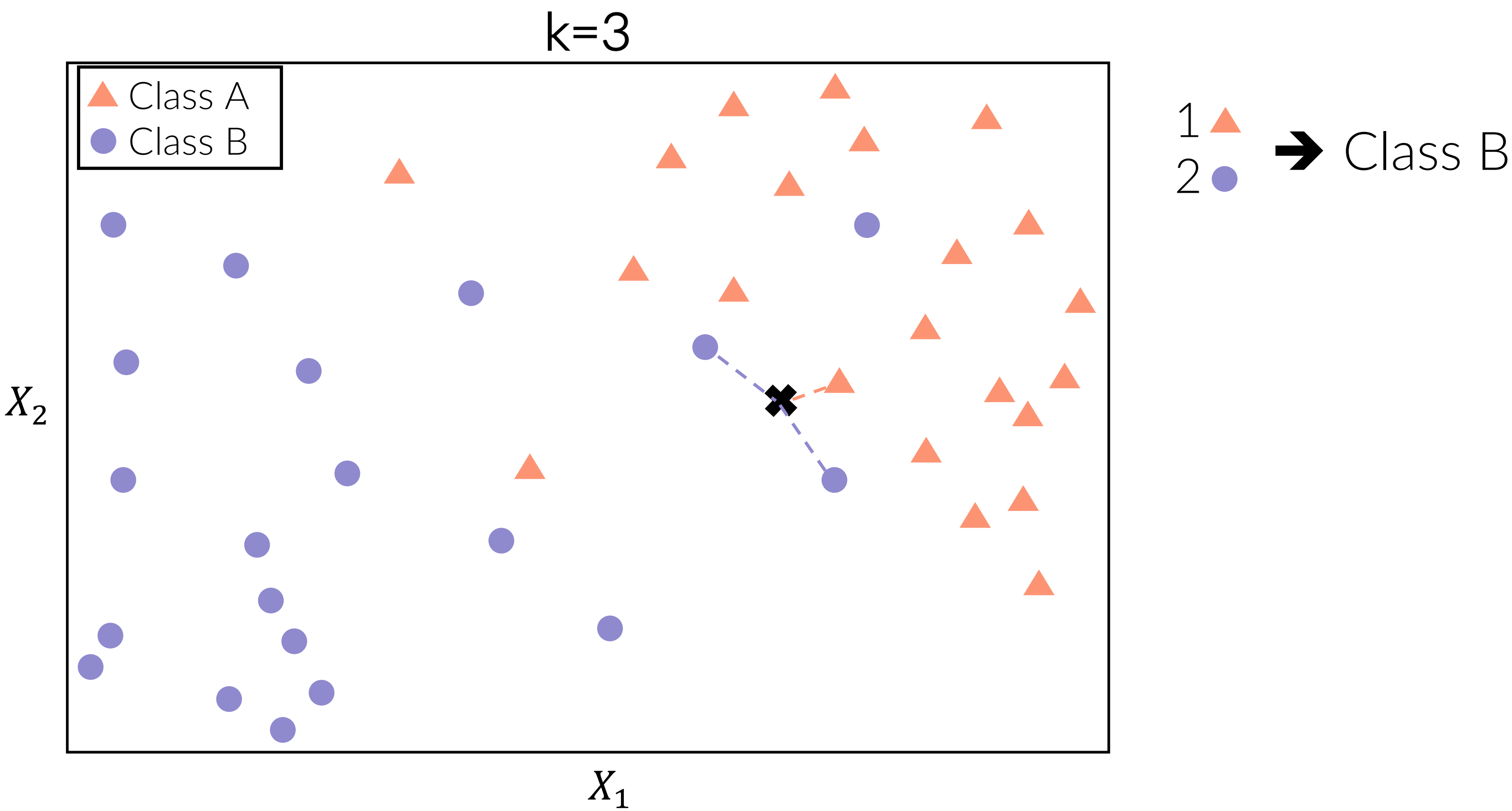
Illustration





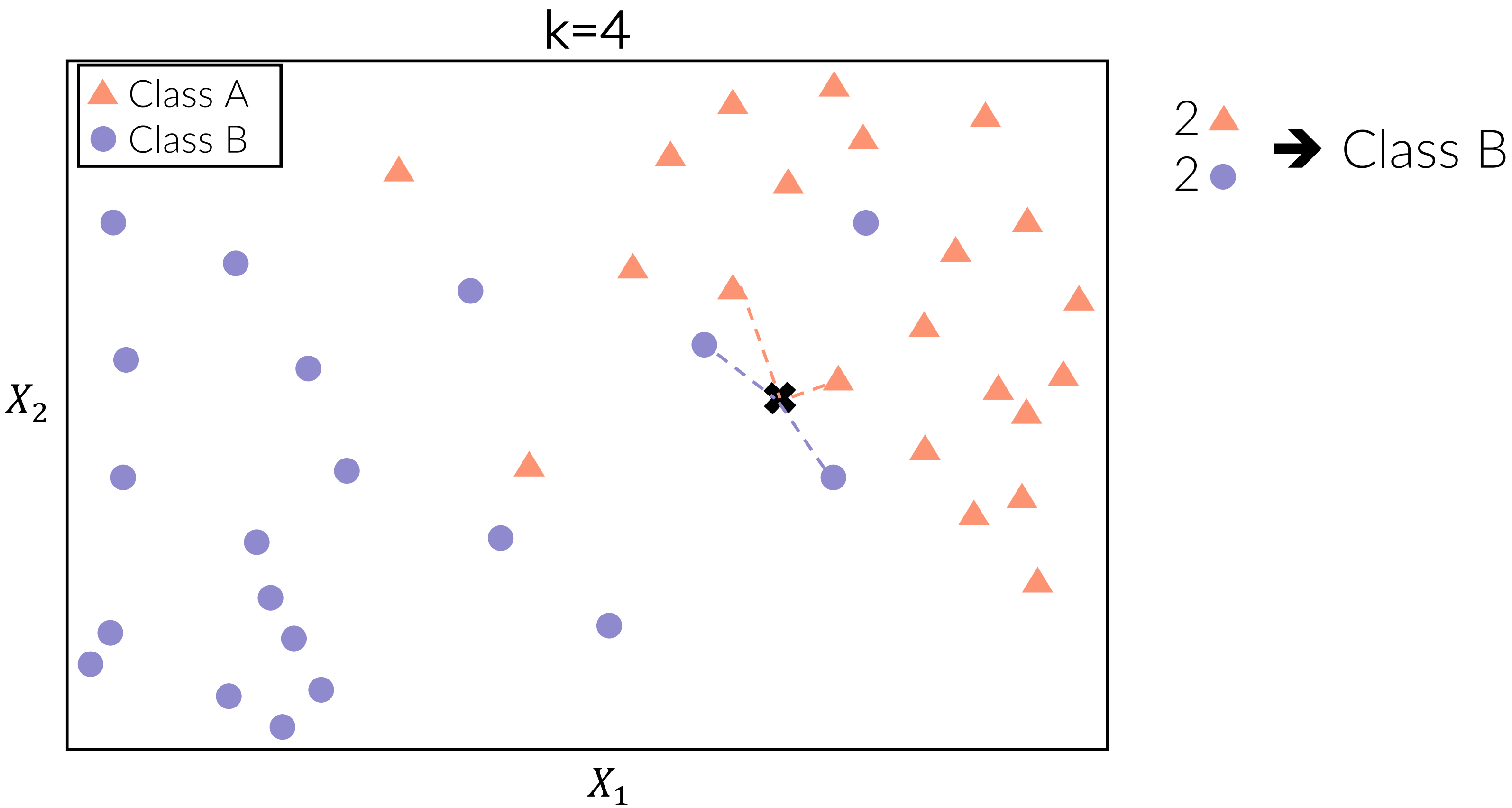
K-NEAREST NEIGHBORS

Illustration



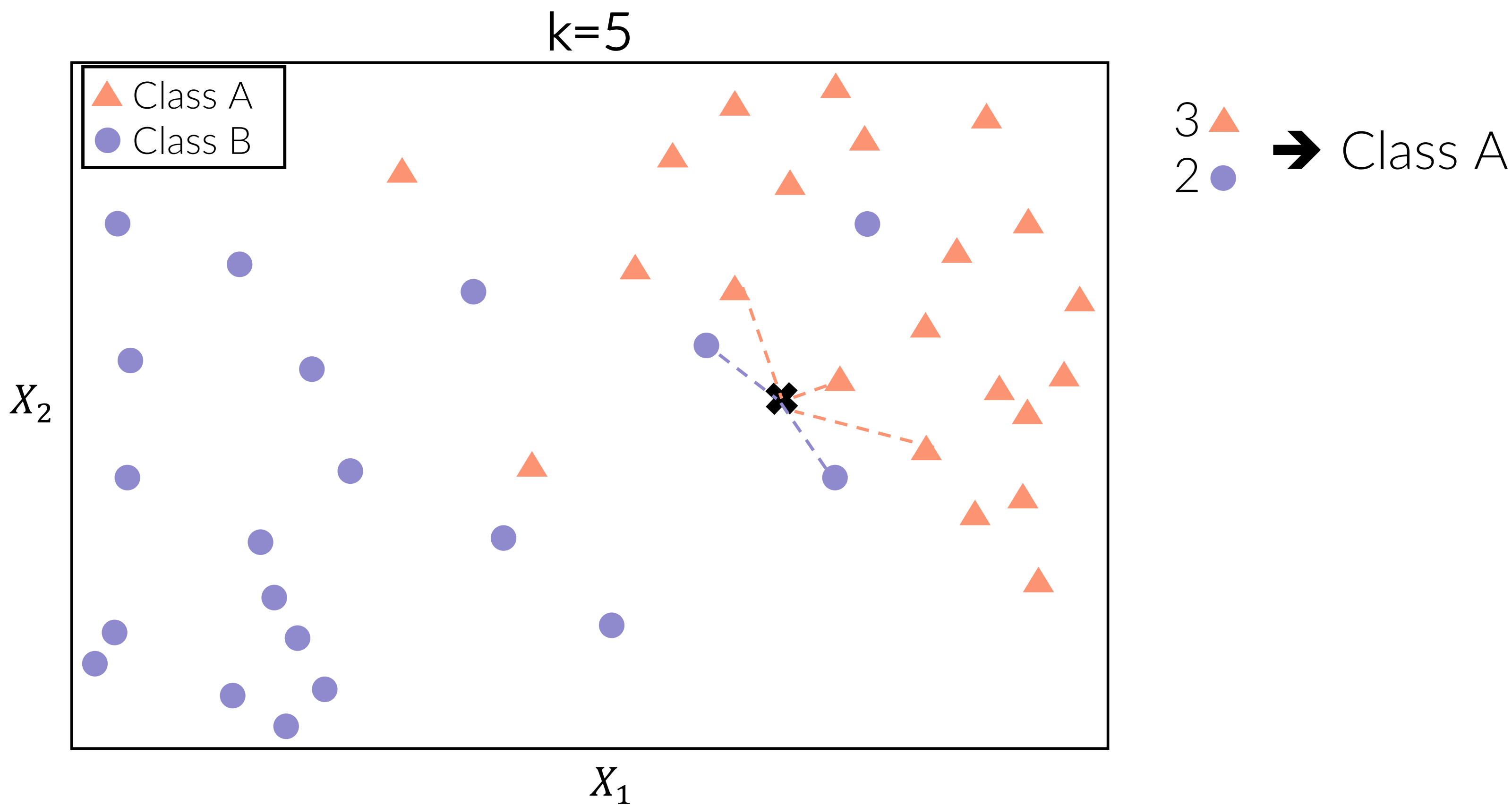
K-NEAREST NEIGHBORS

Illustration



K-NEAREST NEIGHBORS

Illustration



K-NEAREST NEIGHBORS

Process

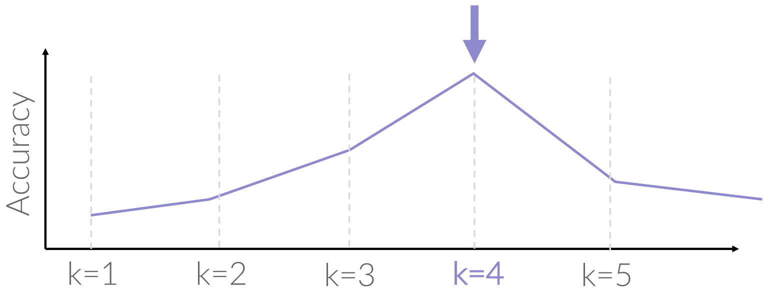
- Pick a value for k
- For each point from the training data set, compute the distance to each observation
- Select the k nearest points and predict the class as the most popular one in that k points.

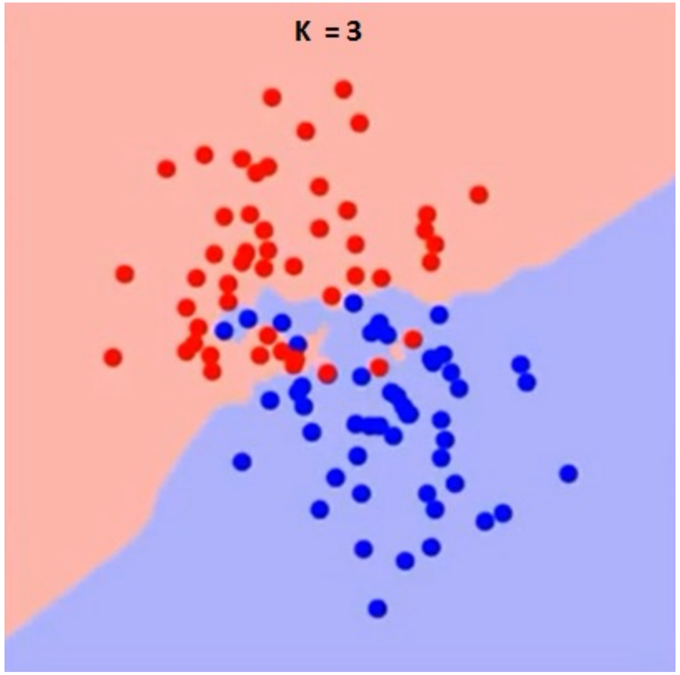
K-NEAREST NEIGHBORS

Parameter setting

- Distance:
  - ℓ_2-norm (Euclidian), ℓ_1-norm, etc.
  - /!\ Scale effect: *The curse of dimensionality*

- k:
  - Low value: sensitivity to noise (over-fitting)
  - High value: overly generalized
  - Pick k that maximize accuracy





K-NEAREST NEIGHBORS

Python implementation

- Training a k-NN model:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean')
knn.fit(X_train, y_train)
```

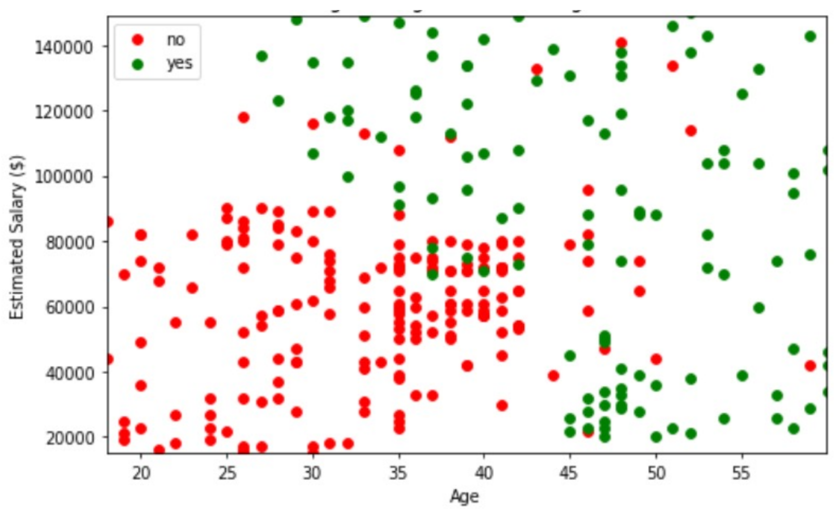
- Using the model for predicting:

```
y_pred = knn.predict(X_test)
```

DECISION TREE

Process

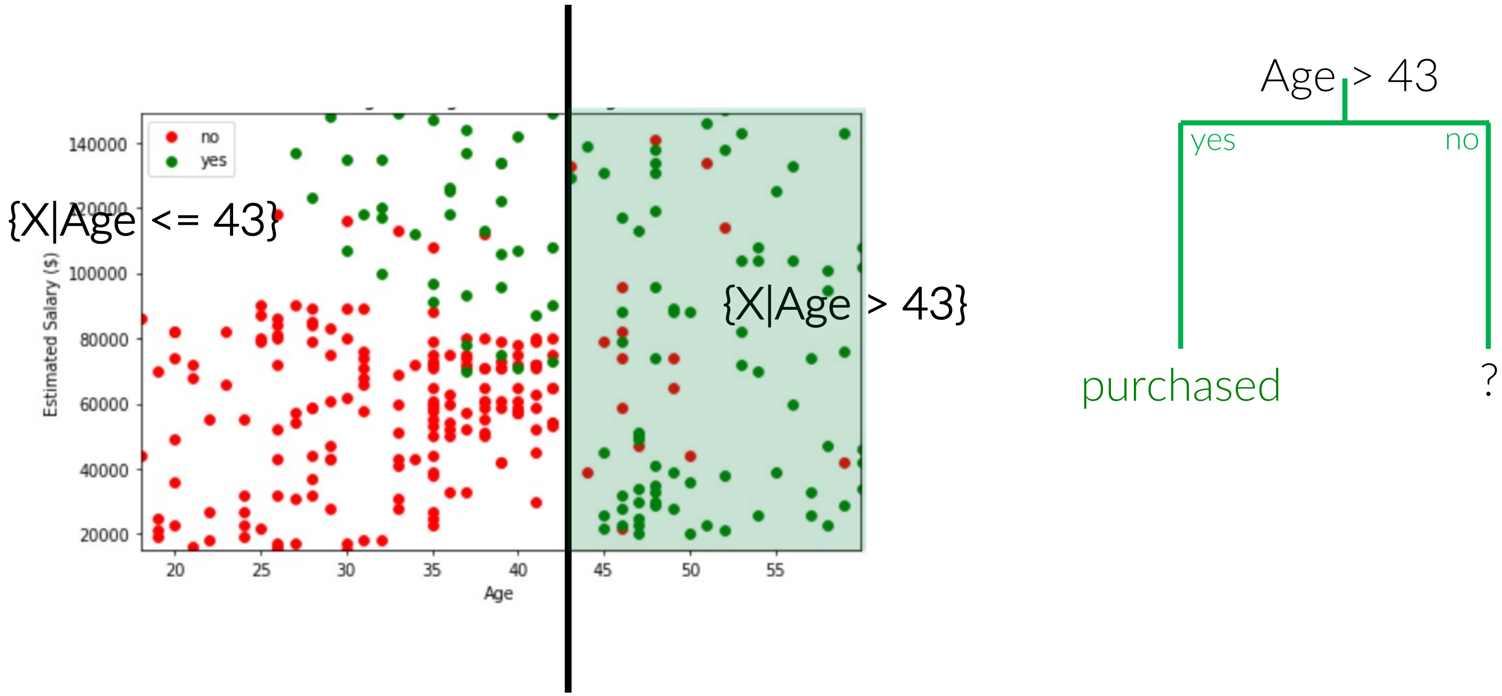
- Objective: split the feature space into distinct and non-overlapping boxes of maximum "purity"



DECISION TREE

Process

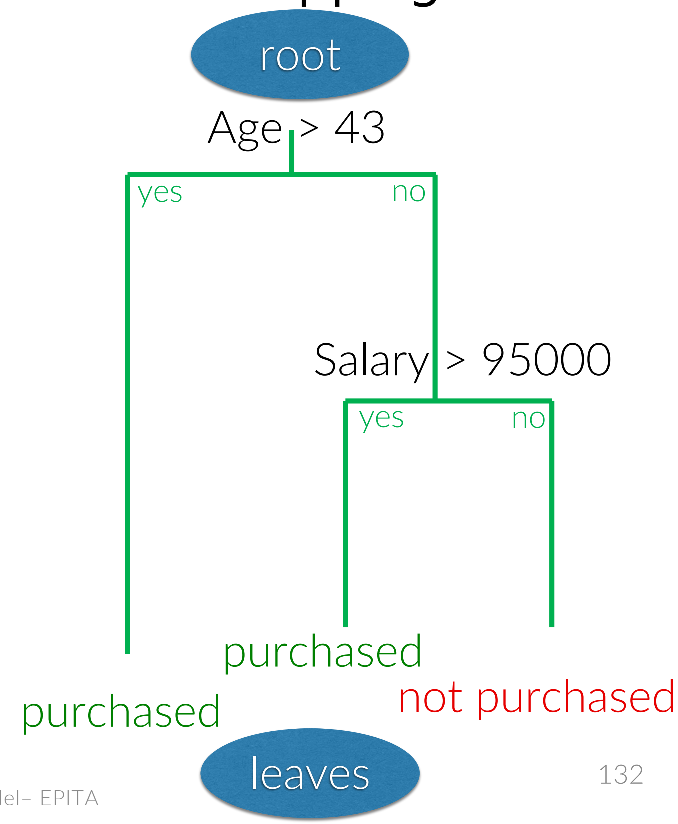
- We divide the predictor space into distinct and non-overlapping boxes



DECISION TREE

Process

- We divide the predictor space into distinct and non-overlapping boxes



DECISION TREE

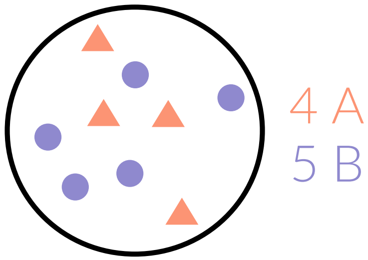
Classification rules setting

- Choose an attribute from the data set
- Calculate the significance of the attribute in splitting of data
- Split data based on the value of the best attribute
- Iterate until a stopping criterion is reached

DECISION TREE

Cross entropy

- Cost function: **cross entropy** $E = -\sum_{k=1}^{K} \hat{p}_k \log_2(\hat{p}_k)$
  - Measure of randomness
  - Value between zero (purity of separation) and one (complete randomness)

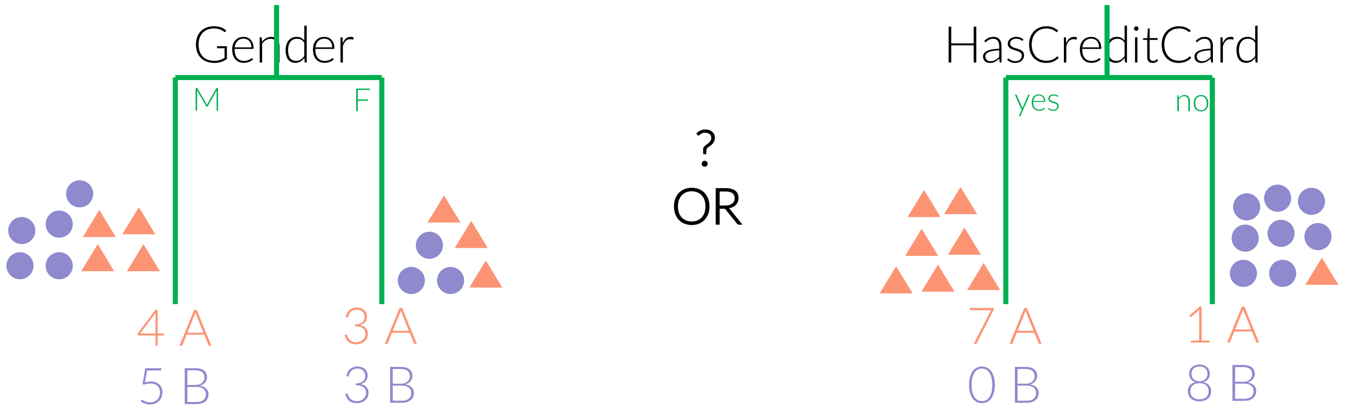


$$E = -\frac{4}{9}\log_2\left(\frac{4}{9}\right) - \frac{5}{9}\log_2\left(\frac{5}{9}\right) = 0.99$$

DECISION TREE

Cross entropy

- Weighted cross entropy to measure information gain and make split decision



$E = 0.99$ $E = 1$

$E = 0$ $E = 0.50$

This split maximizes the purity/predictiveness of the leaves

$$\text{weighted } E = \frac{9}{15}0.99 + \frac{6}{15}1 = 0.99$$

$$\text{weighted } E = \frac{7}{15}0 + \frac{9}{15}0.50 = 0.30$$

DECISION TREE

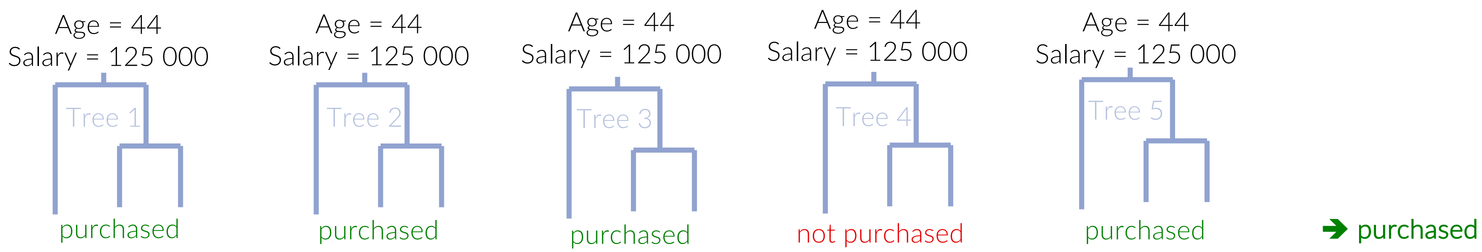
Ensemble learning

“A methodology to combine a set of models, each solves the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions than can be obtained from using a single model”.

DECISION TREE

Random Forests

- Extension of Decision Tree to improve performance (Ensemble method)
- Method:
  - Define the number N_t of trees you want to build
  - Pick N_p random data points from the training set
  - Build the Decision Tree associated to these N_p data points
  - Repeat second and third steps N_t times
  - For the class prediction of a new data points, provide the average across the N_t predictions given by each Decision Tree



DECISION TREE

Python implementation

- Training a Decision Tree model:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = 'entropy')
dt.fit(X_train, y_train)
```

- Random Forest version:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 10, criterion = 'entropy')
```

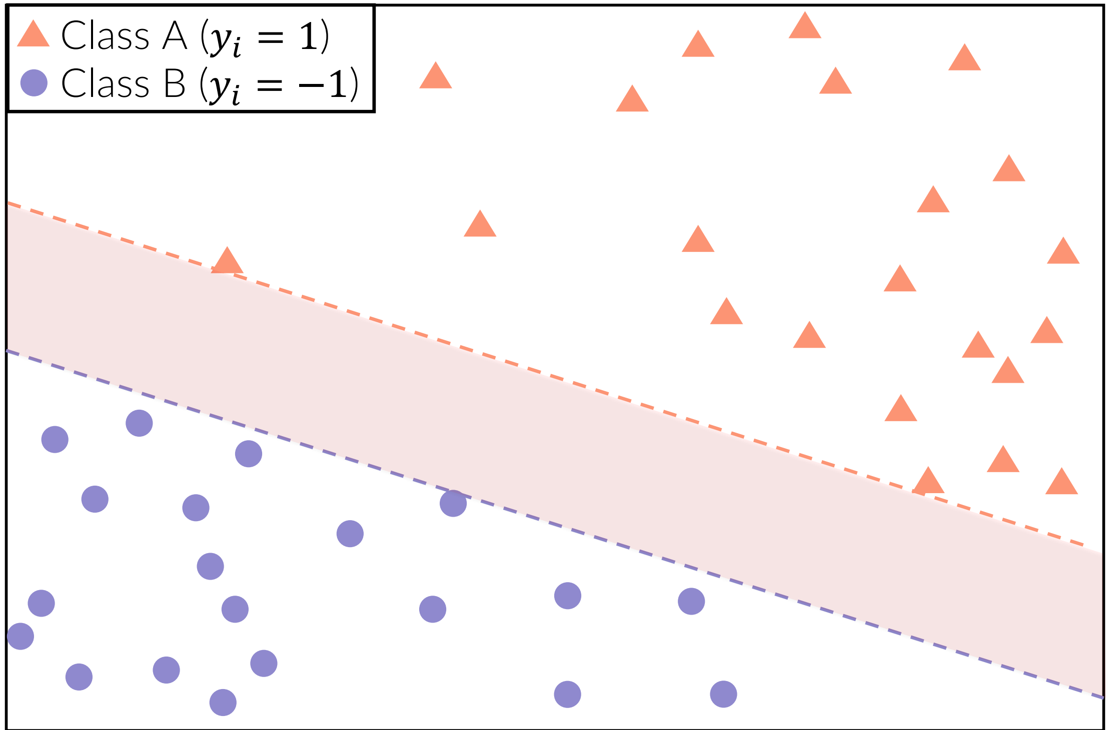
- Using the model for predicting:

```
y_pred = dt.predict(X_test)
```

SUPPORT VECTOR MACHINES

Principle of support vector

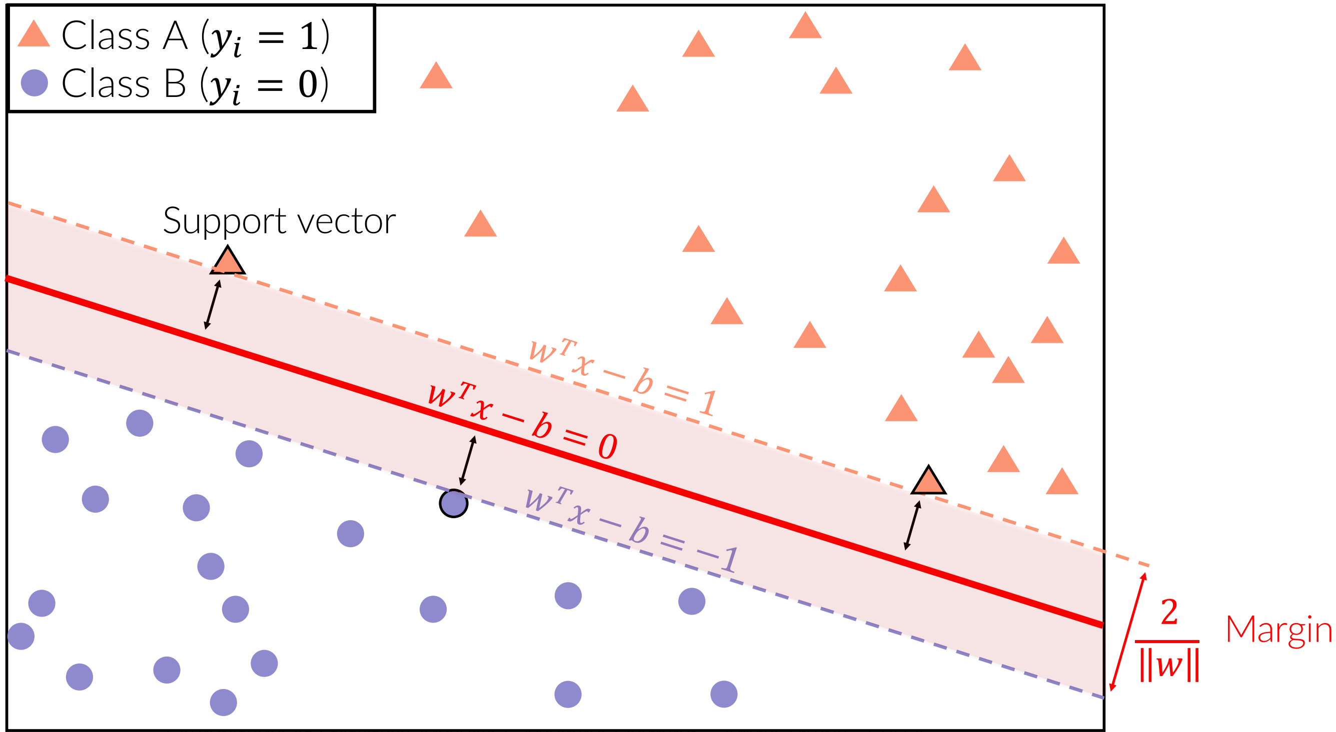
- If the training set is linearly separable, there is a gap between categories



SUPPORT VECTOR MACHINE

Principle of support vector

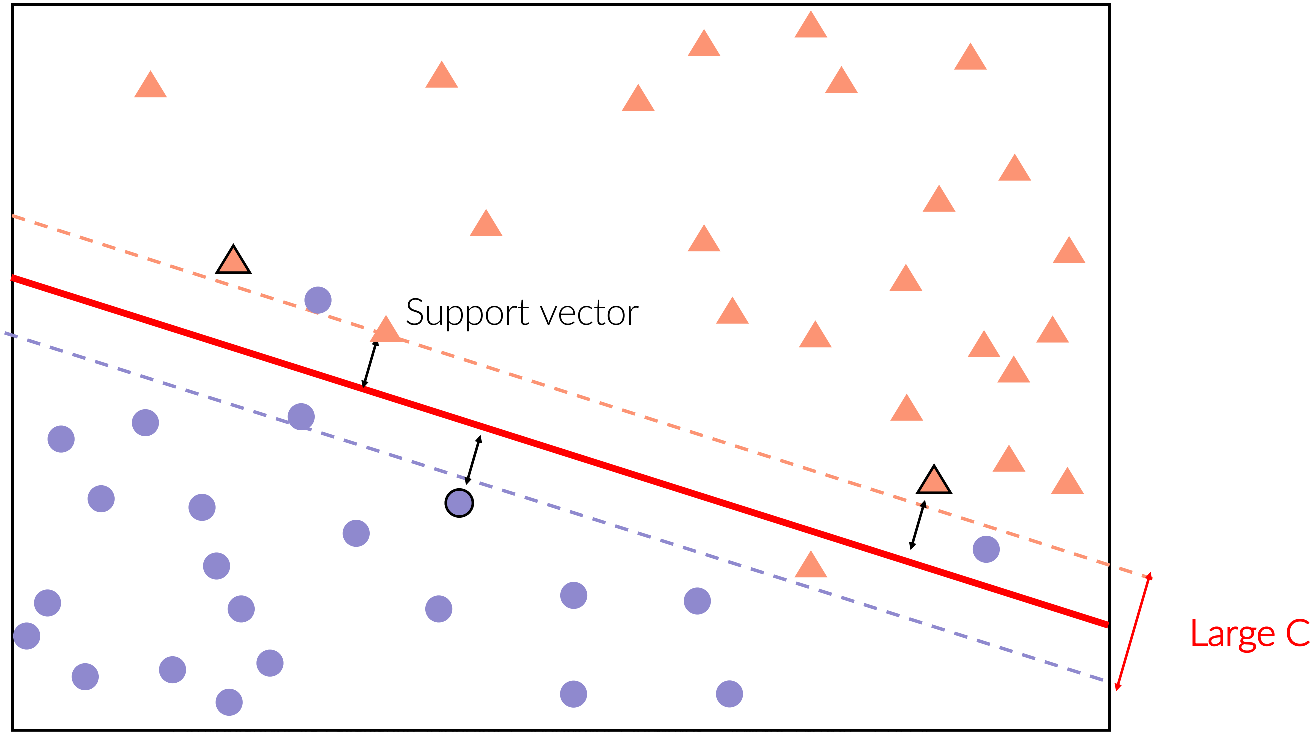
- **Hard margin**: minimize $\|w\|$
  subject to $(w^T x_i - b) \geq 1$ if $y_i = 1$ for $i = 1, \dots, n$
  $(w^T x_i - b) \leq 1$ if $y_i = 0$



SUPPORT VECTOR MACHINE

Principle of support vector

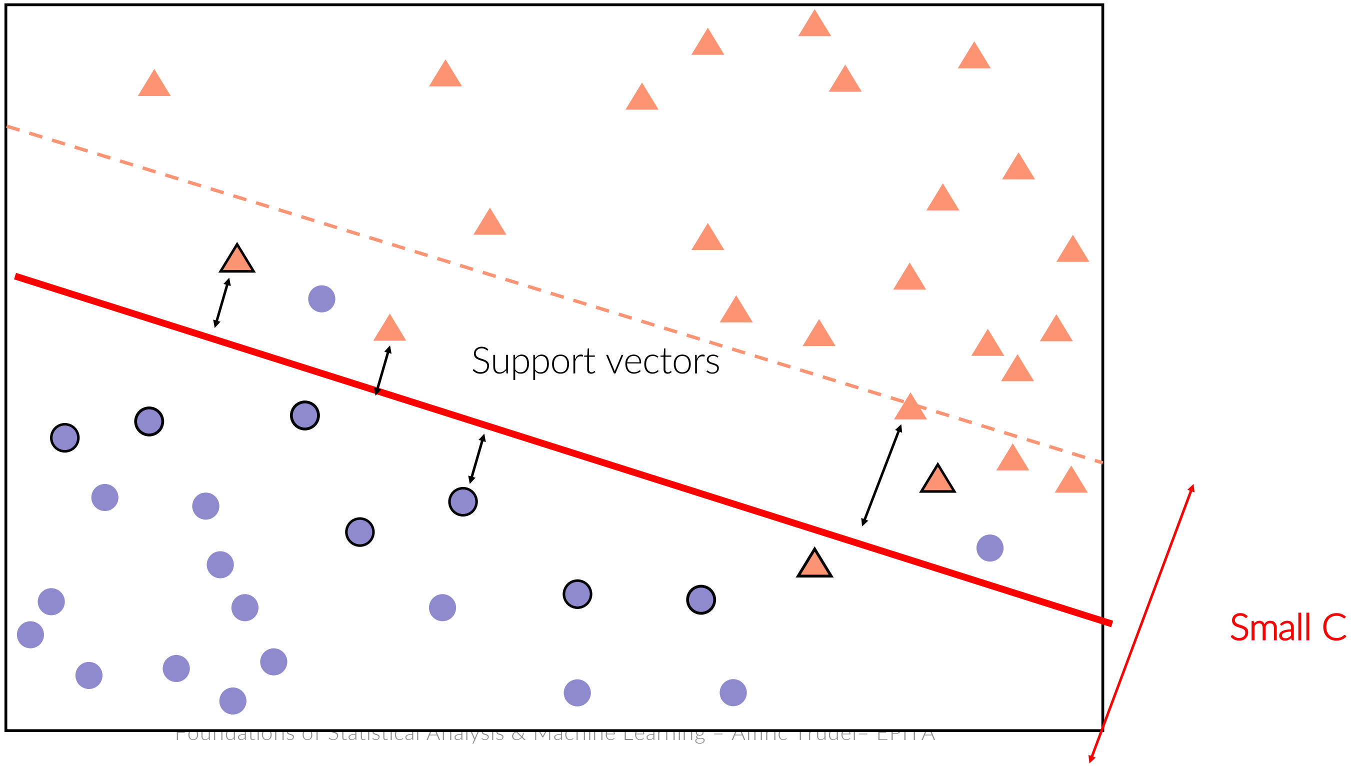
- Extension for a training set that is not linearly separable:
  - **Soft margin**: using C as a regularization parameter



SUPPORT VECTOR MACHINE

Principle of support vector

- Extension for a training set that is not linearly separable:
  - **Soft margin**: using C as a regularization parameter



SUPPORT VECTOR MACHINE

Python implementation

- Training a SVM model for classification:

```
from sklearn.svm import SVC
svm = SVC(kernel = 'linear’, C = 1.0)
```

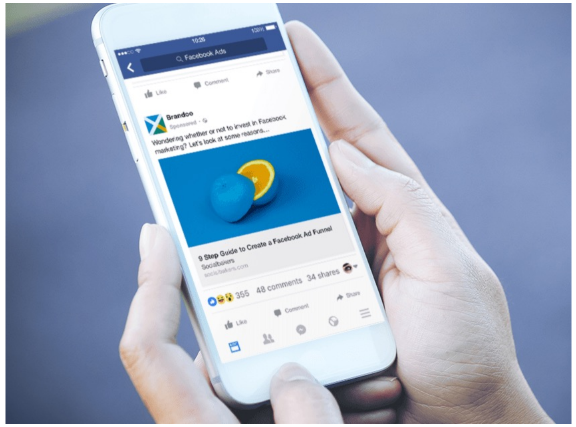
- Using the model for predicting:

```
y_pred = svm.predict(X_test)
```

K-NN, DECISION TREES, SVM

Example of implementation

- Data set: user profiles and sales information
- Objectives:
  - Train a k-NN, a Decision Tree and a SVM models to predict purchase based on profile information
  - Check visually the results on decision areas
  - Compare the performances of the different models



	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	no
2	15810944	Male	35	20000	no
3	15668575	Female	26	43000	no
4	15603246	Female	27	57000	no
5	15804002	Male	19	76000	no
6	15728773	Male	27	58000	no
7	15598044	Female	27	84000	no
8	15694829	Female	32	150000	yes
9	15600575	Male	25	33000	no
10	15727311	Female	35	65000	no
11	15570769	Female	26	80000	no
12	15606274	Female	26	52000	no
13	15746139	Male	20	86000	no

SUPPORT VECTOR MACHINE

Student practice

- Data set: breast cancer diagnosis based on tumor characteristics
- Objectives:
  - Train a k-NN, a Decision Tree and a SVM models on two predictors
  - Visualize the results
  - Compare the performances

mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	tumor type
14.69	13.98	98.22	656.1	0.10310	0.18360	0.14500	0.06300	0.2086	0.07406	benign
13.17	18.66	85.98	534.6	0.11580	0.12310	0.12260	0.07340	0.2128	0.06777	malignant
12.95	16.02	83.14	513.7	0.10050	0.07943	0.06155	0.03370	0.1730	0.06470	benign
18.31	18.58	118.60	1041.0	0.08588	0.08468	0.08169	0.05814	0.1621	0.05425	malignant
15.13	29.81	96.71	719.5	0.08320	0.04605	0.04686	0.02739	0.1852	0.05294	malignant
16.16	21.54	106.20	809.8	0.10080	0.12840	0.10430	0.05613	0.2160	0.05891	malignant
19.19	15.94	126.30	1157.0	0.08694	0.11850	0.11930	0.09667	0.1741	0.05176	malignant
18.08	21.84	117.40	1024.0	0.07371	0.08642	0.11030	0.05778	0.1770	0.05340	malignant