Apprentissage automatique utilisant Scikit-Learn

Nesrine Ammar

26-02-2019

PROGRES

Introduction

Introduction

Définition

- Machine Learning is the science of programming computers so they can learn from data.

- ML is the field of study that gives computers the ability to learn without being explicity programmed.

[Arthur Samuel, 1959]

Introduction

Apprentissage automatique (classificateur)

Chien

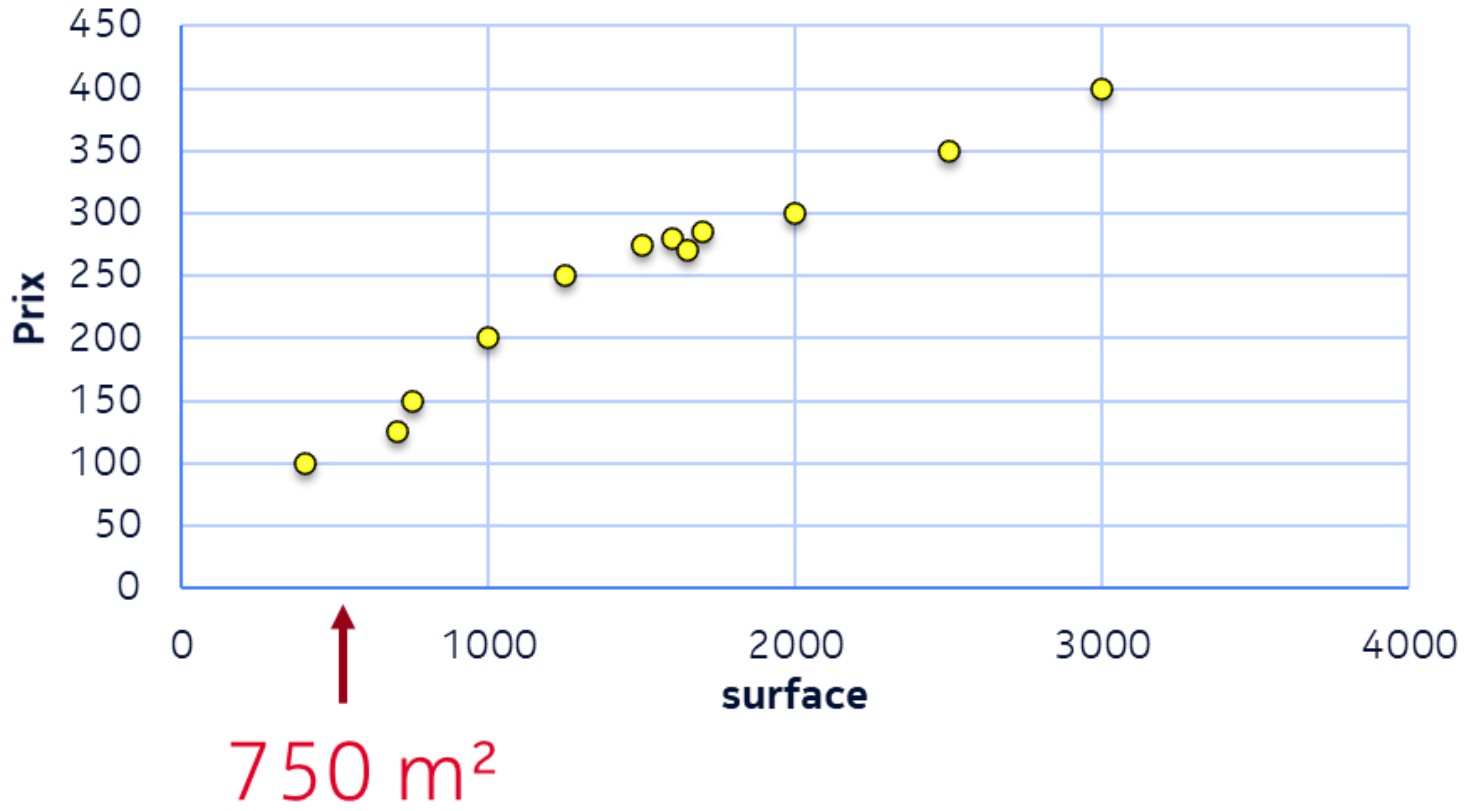
Introduction

- Un **bon modèle de machine learning**, c'est un modèle qui généralise.

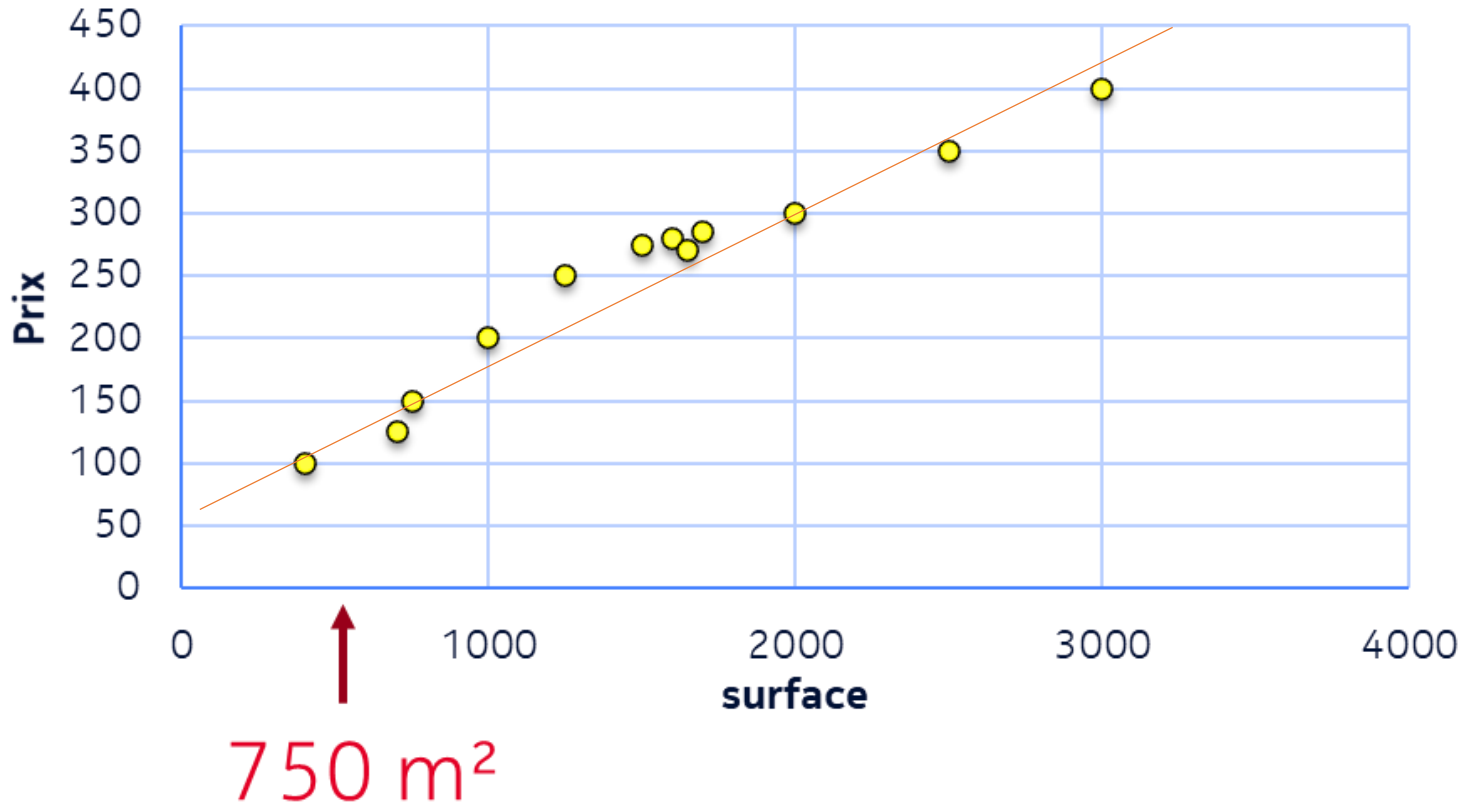
Qu'est-ce que c'est, déjà, la généralisation ?

- La **généralisation**, c'est la capacité d'un modèle à faire des prédictions non seulement sur les données que vous avez utilisées pour le construire, mais surtout sur **de nouvelles données** : c'est bien pour ça que l'on parle **d'apprentissage**
- L'objectif du machine learning est de **trouver un modèle** qui effectue une **approximation** de la réalité (le phénomène à l'origine des données), à l'aide de laquelle on va pouvoir effectuer des prédictions.

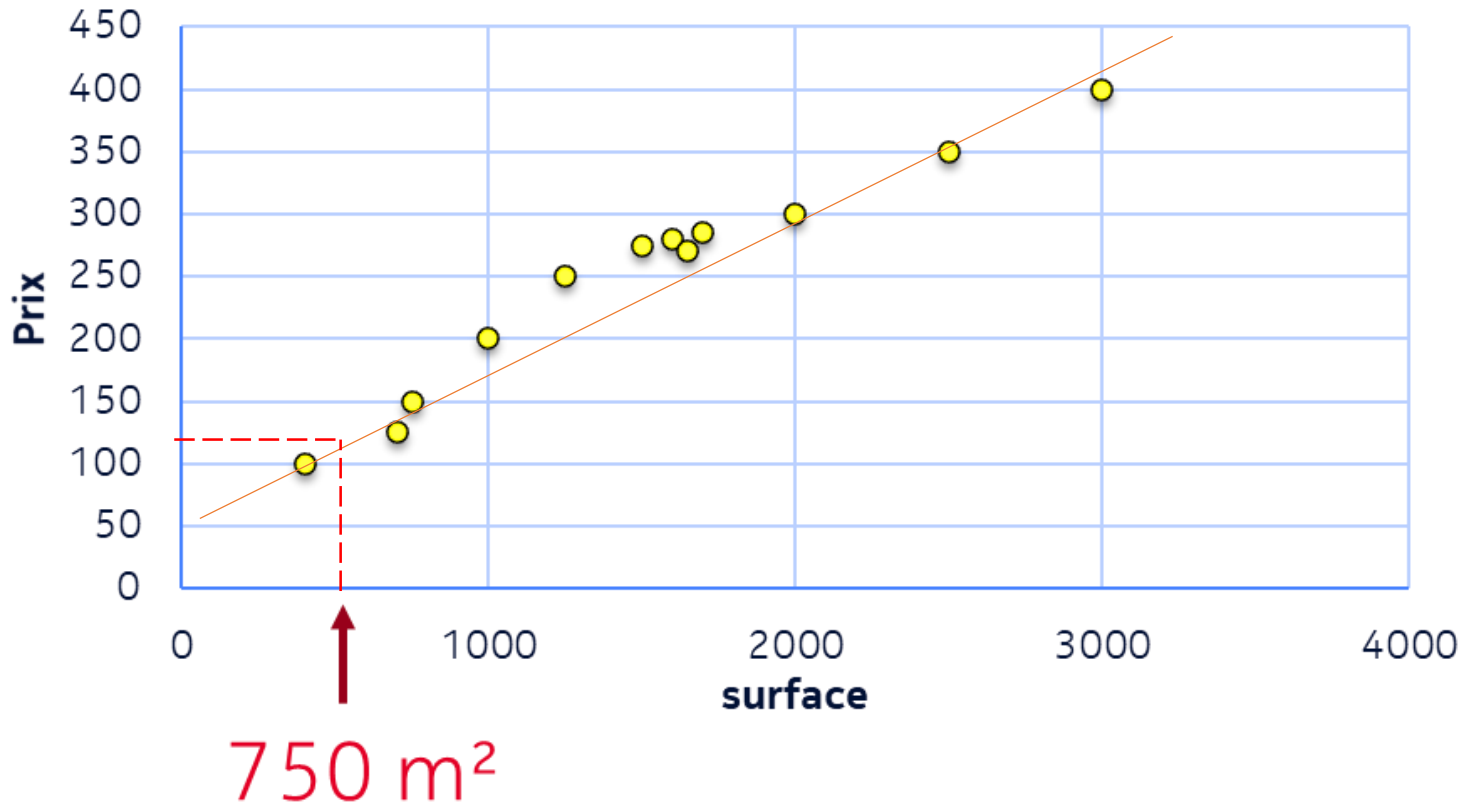
Apprentissage supervisé



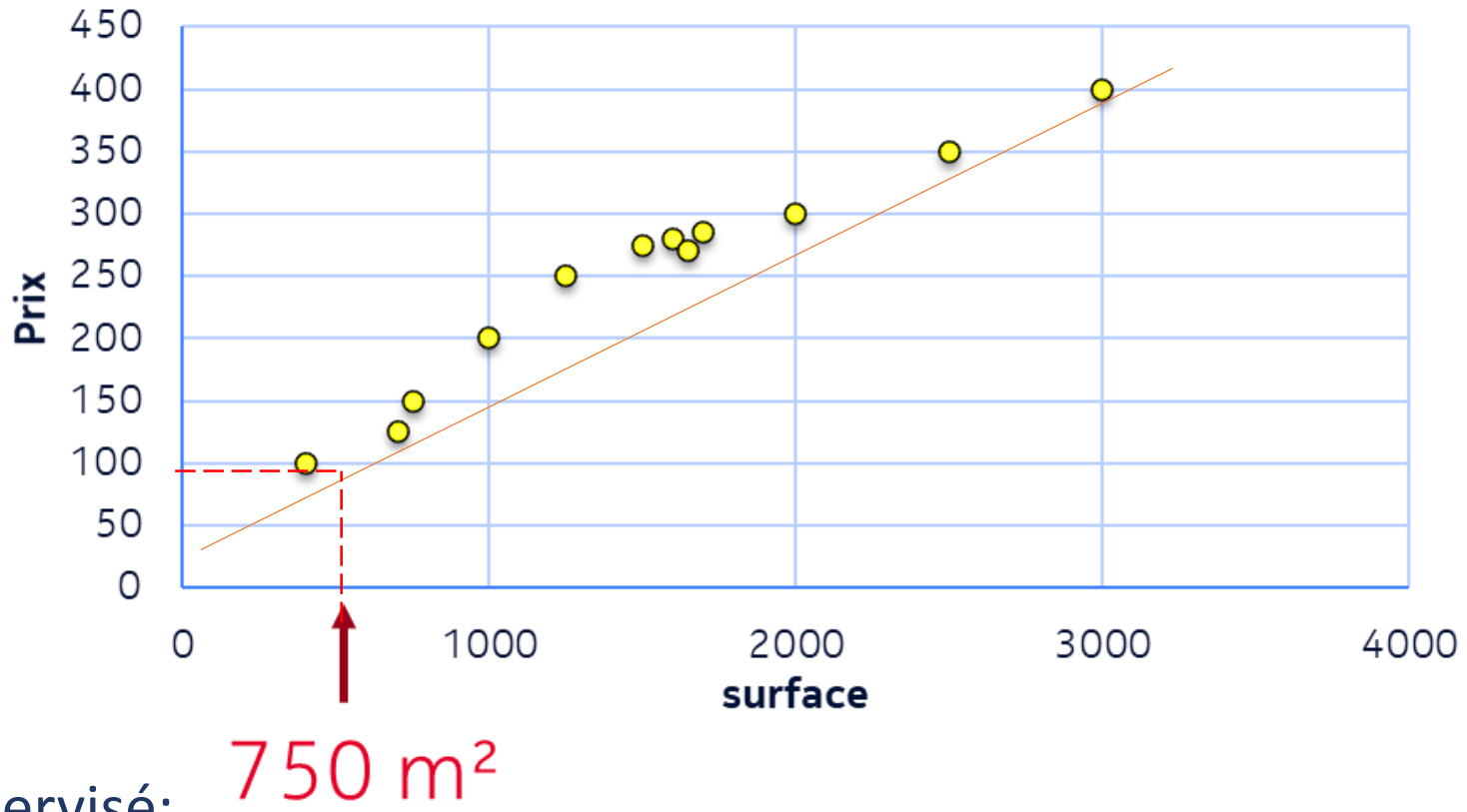
Apprentissage supervisé



Apprentissage supervisé



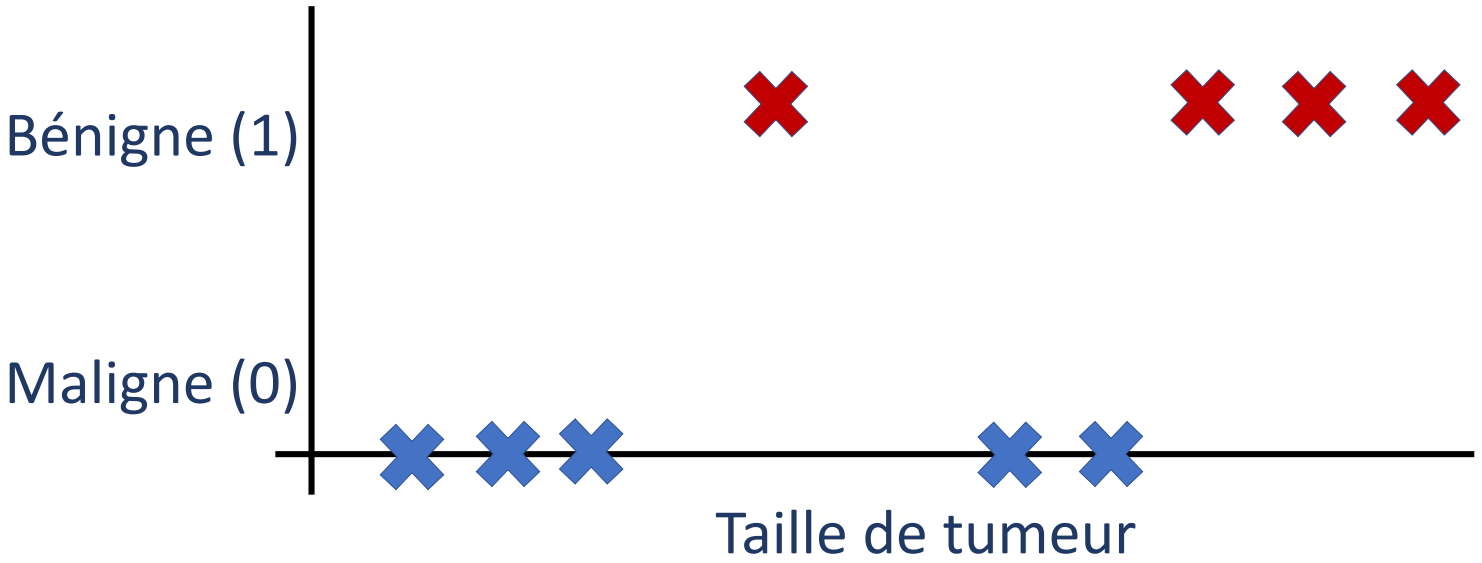
Apprentissage supervisé



Apprentissage supervisé:
La valeur exacte est donnée
Les données sont labélisées

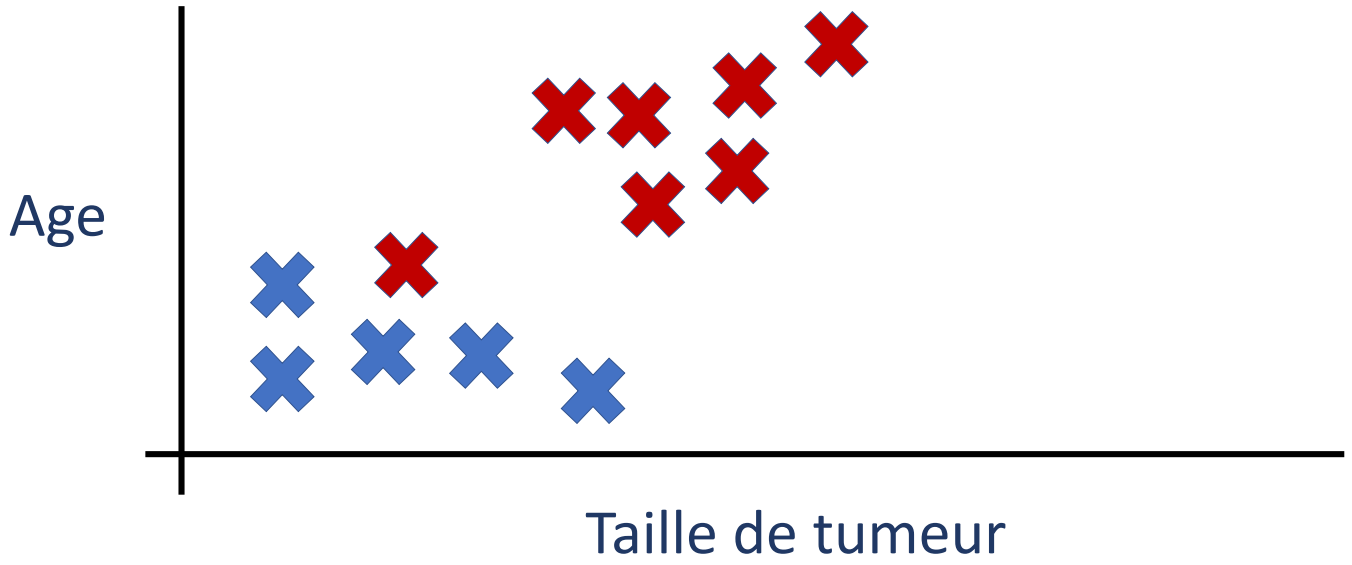
Problème de régression:
Valeurs sont continues

Apprentissage supervisé

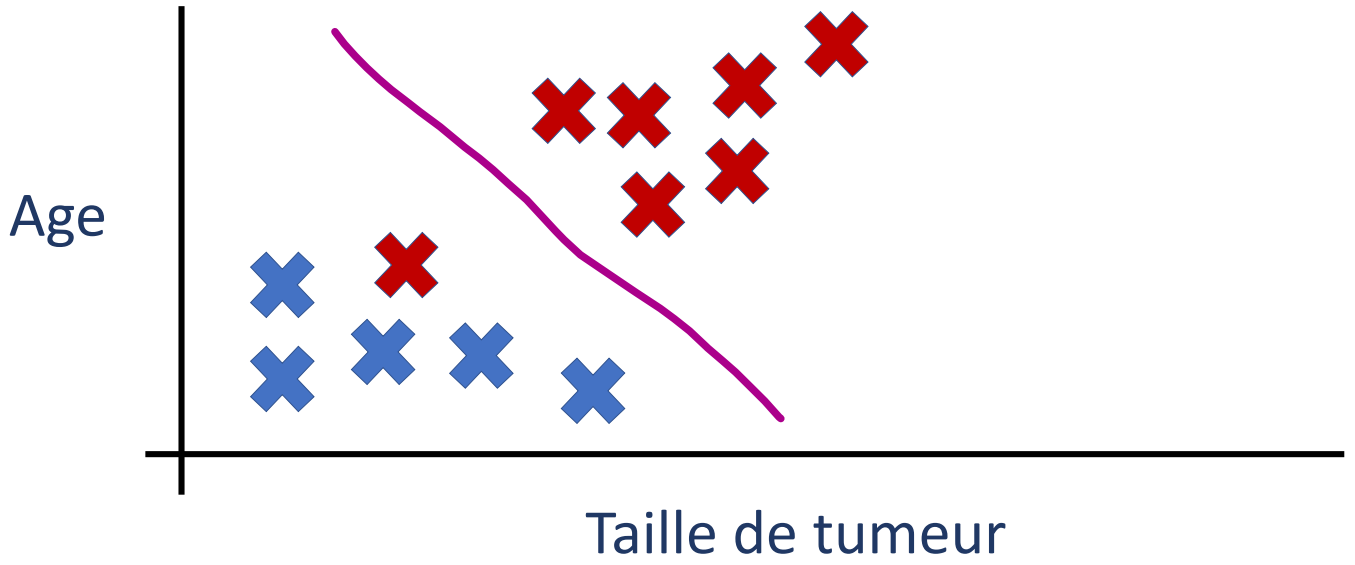


Problème de classification:
Valeurs sont discrètes

Apprentissage supervisé avec 2 attributs



Apprentissage supervisé



Type de cellule
Taux des globules
Type de sang
…

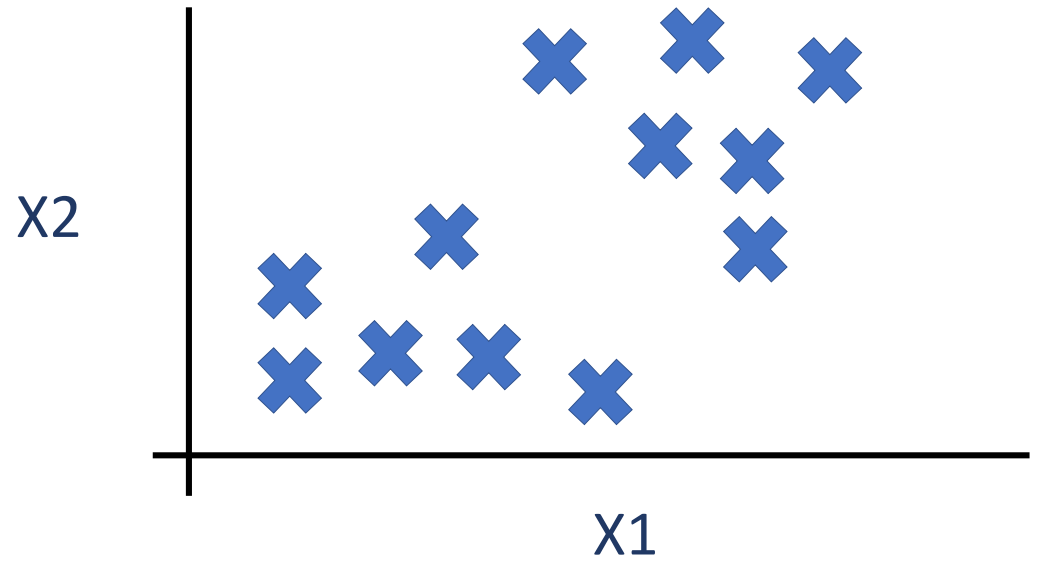
Apprentissage supervisé

2 types d’apprentissage supervisé: Régression vs classification

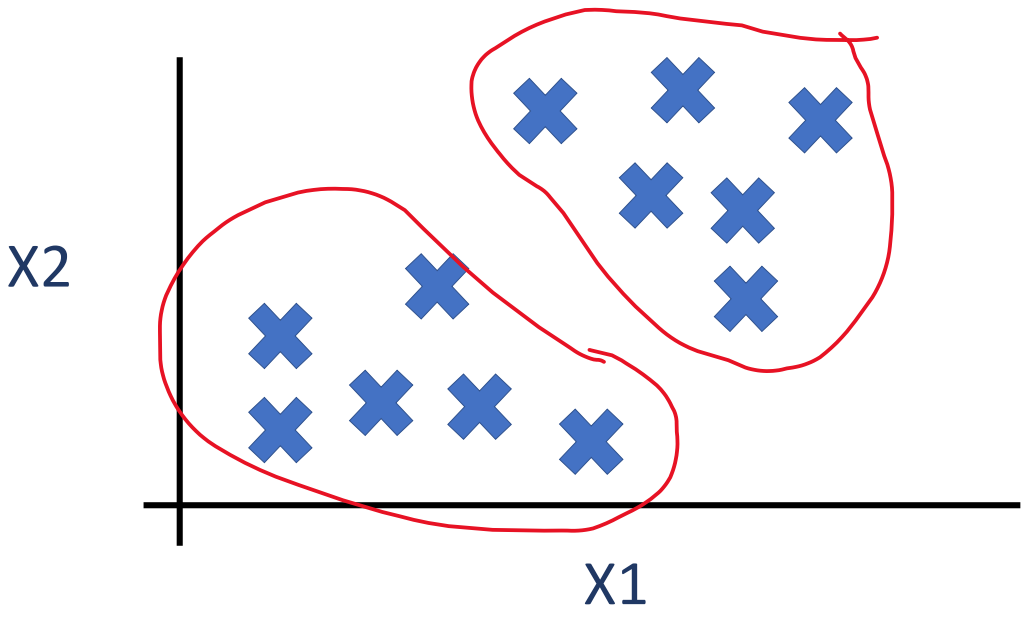
Les algorithmes les plus connus:

- K-Nearest Neighbors
- Linear regression
- Logistic regression
- Support vector machine
- Decision tree
- Random forest
- Neural networks

Apprentissage non-supervisé



Apprentissage non-supervisé

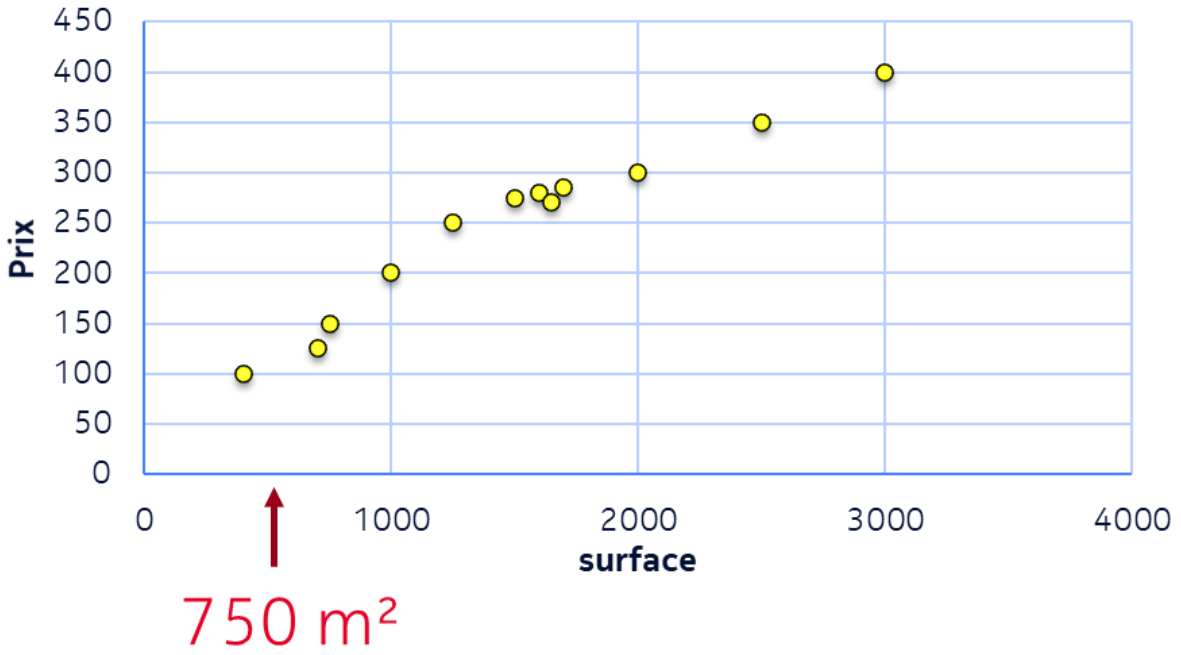


- Inférer des connaissances sur les données
- Les classes ne sont pas définies
- Exemples de tâches associées à l'apprentissage non-supervisé: Clustering (regroupement): construire de classes automatiquement en fonction des données disponibles

Problème d’apprentissage: estimation du prix d’une maison

Pour chaque exemple, le model retourne la valeur exacte

Problème de régression:
Le modèle retourne une valeur continue



Problème d'apprentissage: Régression linaire

Surface(X)	Prix (Y)
2000	480
600	150
500	128
2965	576
1678	321
…	…
…	…

Données d'apprentissage

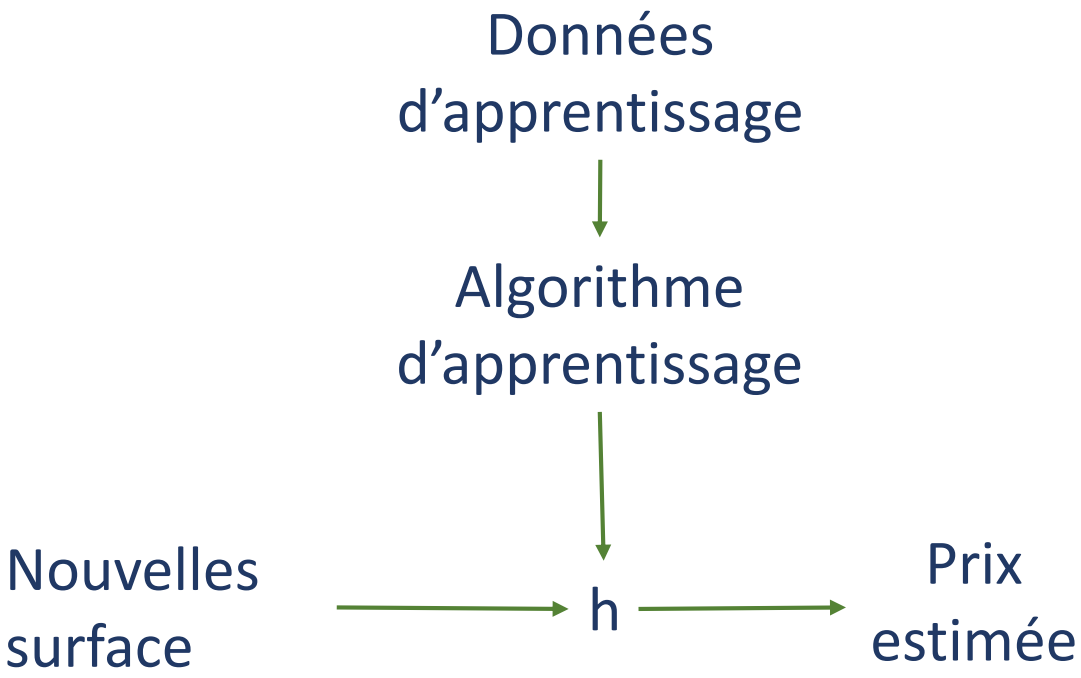
Notation:

- m nombre des exemples des données
- X attributs (valeurs d'entrées)
- Y sortie des données
- (x,y) est 1 exemple d'apprentissage

La régression linéaire:

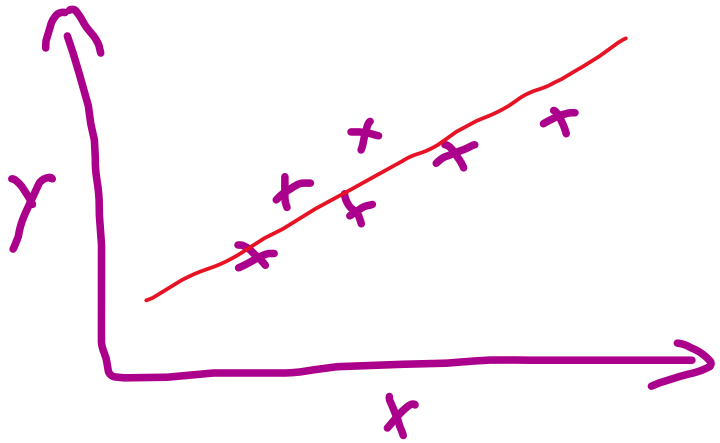
S'appuie sur l'hypothèse que les données proviennent d'un phénomène qui à la forme d'une droite, c'est à dire qu'il existe une relation linéaire entre l'entrée (les observations) et la sortie (les prédictions).

Problème d'apprentissage: Régression linaire



Comment représenter h ?

$$h_\theta(x) = \theta_0 + \theta_1 x$$



Algorithme: Régression linaire avec un seul attribut

Problème d'apprentissage: Régression linaire

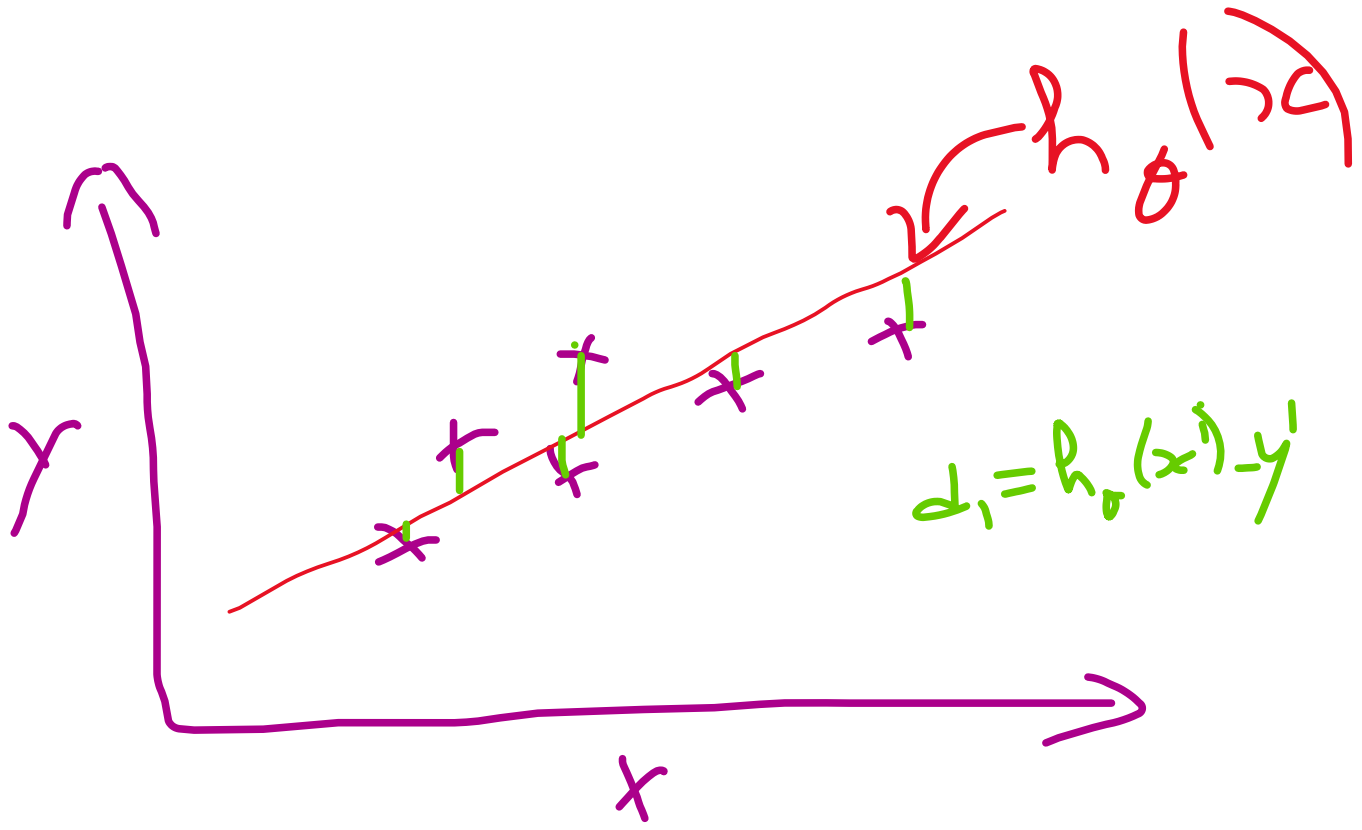
Comment choisir Theta?

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Choisir les valeurs de theta pour que h soit très proche de y dans nos données d'apprentissage (x,y)

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum^{m} (h_\theta(x') - y')^2$$

minimize $J(\theta_0, \theta_1)$



minimize (θ_1, θ_2) $\frac{1}{m} \sum^{m} (h_\theta(x') - y')^2$

Qu’est ce qu’un bon modèle d’apprentissage

- Qu’est ce que le sous-apprentissage et le sur-apprentissage?
- En apprentissage supervisé, le but est de produire des modèles qui généralisent, c’est-à-dire qui sont capables de faire de bonnes prédictions sur de nouvelles données

- De bonnes performances sur le jeu d’entraînement ne garantissent pas que le modèle sera capable de généraliser !

- On cherche à développer un modèle qui soit suffisamment complexe pour bien capturer la nature des données (et éviter ainsi le sous-apprentissage), mais suffisamment simple pour éviter le sur-apprentissage.

Un projet ML de bout en bout (Problème de régression)

1. Comprendre le problème
2. Obtenir les données
3. Analyser et visualiser les données
4. Préparer les données pour l’apprentissage
5. Choisir un algorithme d’apprentissage
6. Former le modèle
7. Régulariser les hyperparamètres du modèle
8. Tester le modèle obtenu

Télécharger les données

```
Import sklearn
Import pandas as pd
Import numpy as np
Import os

#load the data

Def load_housing(housing_path=path):
        csv_path = os.path.join(path, ‘housing.csv’)
        dataset = pd.read_csv(csv_path)
```

Télécharger les données

Name	[Housing units]	Occupied	Vacant	[Median value]	[Median rooms]	[HU: 1 detached]	[HU: 1 attached]
Alabama	2,190,638	1,842,174	348,464	$123,800	5.7	1,501,302	36,331
Alaska	307,820	251,678	56,142	$246,300	4.7	193,912	23,411
American Samoa	10,963	9,688	1,275	$68,200	4.5	8,443	604
Arizona	2,874,548	2,387,246	487,302	$162,900	5.2	1,826,015	140,446
Arkansas	1,329,139	1,132,488	196,651	$108,700	5.4	926,350	21,996
California	13,781,929	12,617,280	1,164,649	$371,400	5.1	8,017,091	960,230
Colorado	2,238,624	1,998,314	240,310	$239,400	5.7	1,408,086	155,702
Connecticut	1,490,381	1,356,206	134,175	$274,500	5.7	882,955	79,922
Delaware	411,250	339,046	72,204	$232,900	6.2	240,246	60,677
District of Columbia	300,798	267,415	33,383	$454,500	4.2	35,925	76,489
Florida	9,051,851	7,217,508	1,834,343	$156,200	5.1	4,902,074	568,390
Georgia	4,114,496	3,540,690	573,806	$148,000	5.8	2,733,507	152,236
Guam	50,567	42,026	8,541	$216,100	4.5	27,241	7,321
Hawaii	524,852	450,299	74,553	$504,500	4.6	282,060	43,361
Idaho	675,421	585,259	90,162	$160,500	5.7	493,472	19,506
Illinois	5,299,433	4,778,633	520,800	$175,700	5.6	3,104,574	309,821
Indiana	2,811,617	2,492,183	319,434	$122,700	5.7	2,043,948	102,693
Iowa	1,348,151	1,232,228	115,923	$126,300	5.9	996,023	50,174
Kansas	1,240,529	1,112,335	128,194	$129,400	5.8	899,161	57,241

Visualiser les données

```
Import sklearn
Import pandas as pd
Import matplot.pyplot as plt

housing = load_housing()
housing.head()                                  → Premiers 5 lignes
housing.info()                                  → numéro des colonnes/lignes/les attributs…
housing[‘ocean proximity’].value_counts()       →  nombre d’exemples par catégorie de l’attribut
housing.hist(bins=50, figsize=(20,15))          → histogramme par chaque attribut
Plt.show()
```

Préparer les données de teste

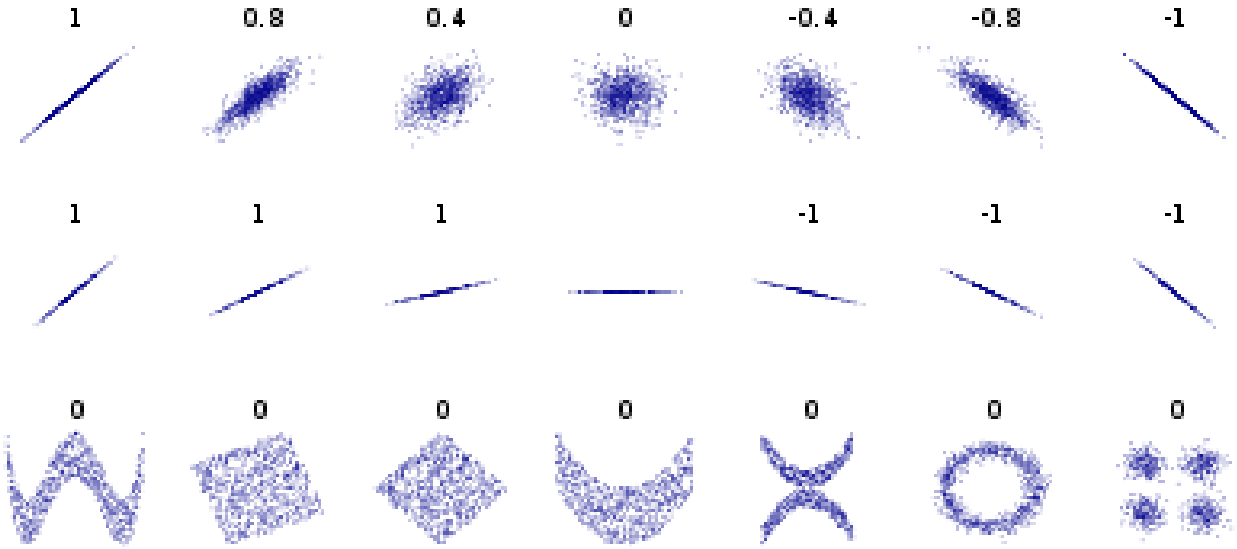
```
Import sklearn
Import pandas as pd
Import numpy as np
Import matplot.pyplot as plt

Housing.shape()
train_set, test_set = housing[:400], housing[400:]

From sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

Visualiser la corrélation



Import sklearn
Import pandas as pd
Import numpy as np
Import matplot.pyplot as plt

train_set.plot(kind='scatter', x='longitude', y='latitude')
train_set. plot(kind='scatter', x='longitude', y='latitude', alph=0.1)

corr_matrix= train_set.corr()
corr_matrix['rooms'].sort_values(ascending=False)

Apprentissage: LinearRegression

```
Import sklearn
Import pandas as pd
Import numpy as np

y_train= train_set['labels']
X_train= train_set.drop('labels', axis=1)

From sklearn.linear_model import LinearRegression
```

→ Select a linear model

```
Lin_reg =LinearRegression()
```

→ Fit to our data

```
Lin_reg.fit(X_train, y_train)
```

Apprentissage: LinearRegression

```
Import sklearn
Import pandas as pd
Import numpy as np

Some_data = X_train[:3]
Some_labels = y_train[:3]
Print ('predictions:', lin_reg.predict(some_data))

>> Predictions: [210,000 , 311,000, 210,000]
Print('labels:' list(some_labels))

>> Labels: [286,000 , 340,000 , 198,000]
```

Apprentissage: LinearRegression

```
Import sklearn
Import pandas as pd
Import numpy as np

From sklearn.metrics import mean_squared_error
```

→ Test on the train set using mean squareerror (make predictions on train set)

```
housing_predictions = lin_reg.predict(X_train)
Lin_mse = mean_squared_error(y_train, housing_predictions)
Print lin_mse
>> 68,000
```

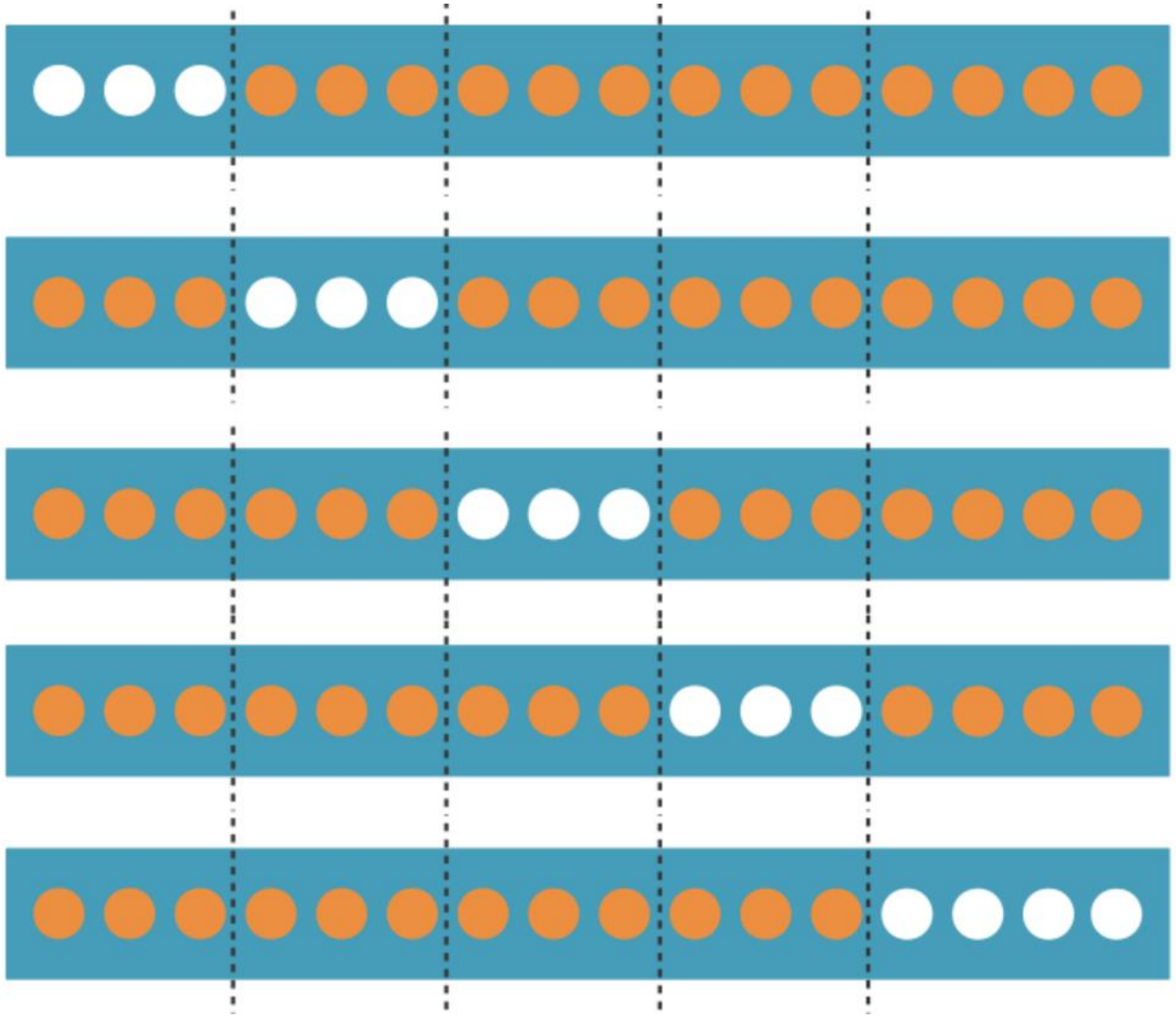
Apprentissage: DecisionTree

```
Import sklearn
Import pandas as pd
Import numpy as np
From sklearn.linear_model import DecisionTree

dtree= DecisionTree()
dtree.fit(train_set,train_labels)

Housing_predictions = dtree.predict(X_train)
Lin_mse = mean_squared_error(y_train, housing_predictions)
Print lin_mse
>> 0,0
```

Apprentissage: Validation croisée



La validation croisée va nous permettre d'utiliser l'intégralité de notre jeu de données pour l'entraînement et pour la validation

On découpe le jeu de données en k parties (folds en anglais) à peu près égales. Tour à tour, chacune des k parties est utilisée comme jeu de test. Le reste (autrement dit, l'union des k-1 autres parties) est utilisé pour l'entraînement.

Exemple: Une cross-validation à 5 folds : Chaque point appartient à 1 des 5 jeux de test (en blanc) et aux 4 autres jeux d'entraînements (en orange)

Apprentissage: Validation croisée

```
Import sklearn
Import pandas as pd
From sklearn.model_selection import cross_val_score

Scores = cross_val_score(dtree, y_X_train, y_train, scoring='neg_mean_square_error ',  cv=4)
Print scores
>> [68,000, 70,000, 67,000 , 74,000]
Print scores.mean()
>>
```

Apprentissage: Méthodes

train_test_split(*arrays, **options)

fit(X, y)	Fit linear model.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict using the linear model
score(X, y)	Returns the coefficient of determination R^2 of the prediction.
set_params(**params)	Set the parameters of this estimator.

mean_squared_error(*y*, *y_pred*)

cross_val_score(estimator, X, y=None, groups=None, scoring=None, cv='warn')

Notes

- Il ne faut jamais évaluer un modèle sur des points qui ont été utilisés pour l'entraîner.
- On sépare donc les données entre un jeu d'entraînement, sur lequel on apprend le modèle, et un jeu de test, sur lequel on l'évalue.
- Pour utiliser l'intégralité de nos données pour entraîner et pour tester, et pour éviter un biais potentiel lié au fait de faire une évaluation unique, on préfère faire une validation croisée.
- Dans le cas d'un problème de classification, on fait attention à stratifier la validation croisée pour éviter d'introduire des biais

Hyperparamètres

- La plupart des algorithmes ont des hyperparamètres, comme le nombre de feuilles pour un algorithme d'arbre de décision, qu'il faut se fixer.
- Le choix de l'algorithme et les hyperparamètres permettent de construire le modèle le plus adapté à notre problème
- Exemple de hyperparamètres pour Decision Tree:

Min_samples_split	Nombre min qu'un nœud doit avoir avant de faire la division
Min_samples_leaf	Nombre min de feuilles qu'un nœud doit avoir
Max_leaf_nodes	Nombre maximum de feuilles par nœud
Max_features	Nombre max d'attributs utilisés par un nœud pour faire la division
Max_depth	La profondeur maximal d'un arbre

```
dtree= DecisionTree(max_depth=2)
dtree.fit(train_set,train_labels)
```

Evaluer sur les données de test

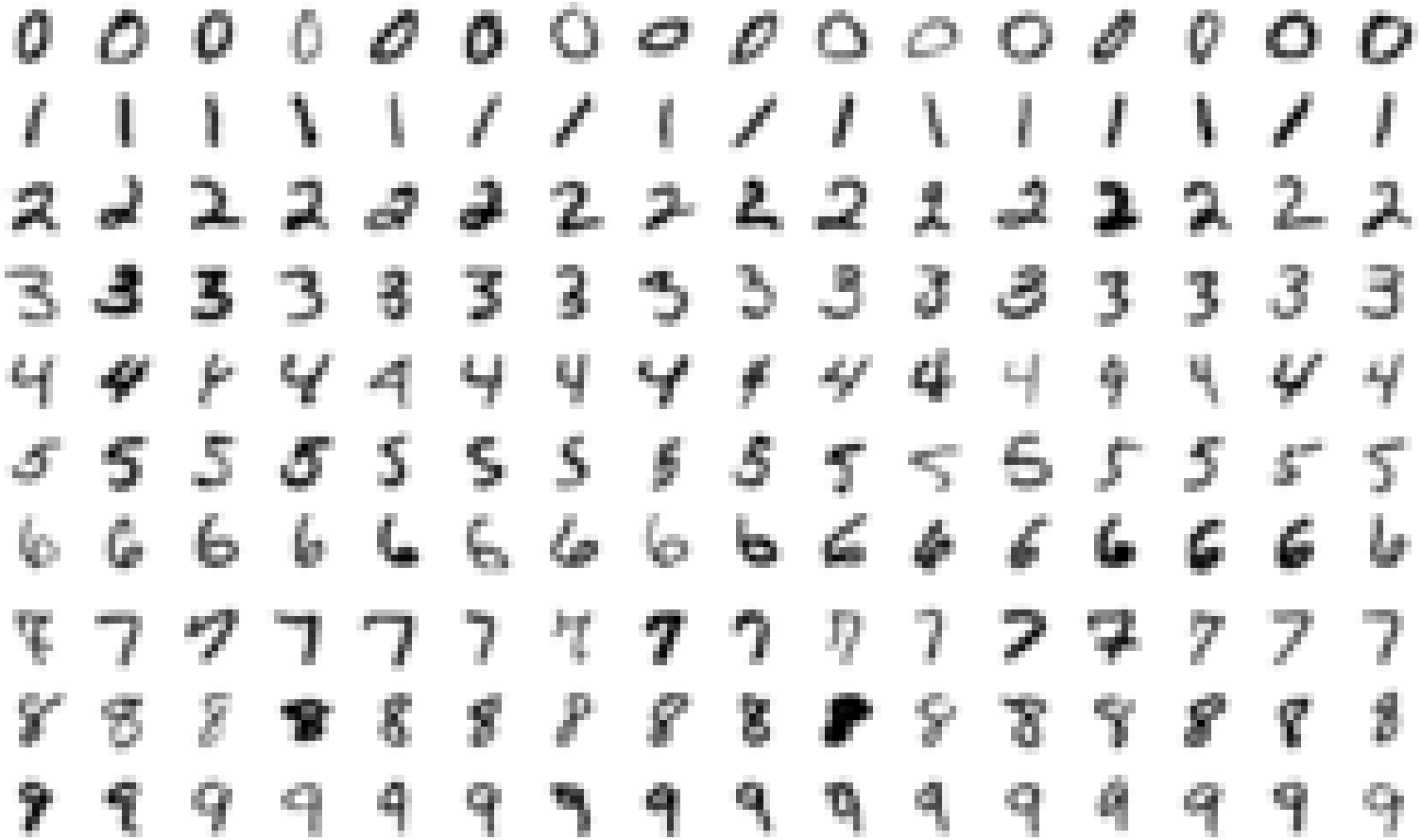
```
Import sklearn
Import pandas as pd
Import numpy as np
From sklearn.linear_model import DecisionTree

predictions = lin_reg.predict(X_test)

mae = np.mean(abs(predictions - y_test))

print('LinearRegression Performance on the test set: MAE = %0.4f' % mae)
```

Problème de classification



- Les données d'apprentissage composé d'image de chiffres
- 9 classes
- Classification multi classe, on utilise dans ce cas la technique (one vs all) consiste à entrainer un classificateur par classe, pour 9 classe nous aurons 9 classificateurs.
- One vs all: des classificateurs binaires

Classificateur binaire

```
Import sklearn
Import pandas as pd

dataset = pd.read_csv(csv_path)
y= dataset['labels']
X= dataset('labels', axis=1)
Print X.shape, y.shape
>> (70000, 784), (70000,)

X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]

Y_train_5 = (y_train ==5)
Y_test_5 = (y_test == 5)
```

Classificateur binaire

```
From sklearn.linear_model import SGDClassifier

Sgd_clf = SGDClassifier()
Sgd_clf.fit(X_train, y_train_5)

Sgd_clf.predict([un chiffre 5])
>> True

From sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring=accuracy)
>> [0.9502, 0.96565, 0.96495]
```

Multi classe

```
From sklearn.linear_model import SGDClassifier

Sgd_clf = SGDClassifier()
Sgd_clf.fit(X_train, y_train)

Sgd_clf.predict([un chiffre 5])
>> 5

Sgd_clf.predict_proba([un chiffre])
>> [0.1, 0 , 0, 0.1, 0, 0.8 , 0, 0, 0, 0]
```

Pandas

```
>>> import pandas as pd
>>> import numpy as np
>>> df = pd.read_csv('data/sample_data.csv', index_col=0)
>>> df
```

	state	color	food	age	height	score
Jane	NY	blue	Steak	30	165	4.6
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0
Penelope	AL	white	Apple	4	80	3.3
Dean	AK	gray	Cheese	32	180	1.8
Christina	TX	black	Melon	33	172	9.5
Cornelia	TX	red	Beans	69	150	2.2

```
>>> index = df.index
>>> columns = df.columns
>>> values = df.values

>>> index
Index(['Jane', 'Niko', 'Aaron', 'Penelope', 'Dean', 'Christina',
       'Cornelia'], dtype='object')

>>> columns
Index(['state', 'color', 'food', 'age', 'height', 'score'],
      dtype='object')

>>> values
array([['NY', 'blue', 'Steak', 30, 165, 4.6],
       ['TX', 'green', 'Lamb', 2, 70, 8.3],
       ['FL', 'red', 'Mango', 12, 120, 9.0],
       ['AL', 'white', 'Apple', 4, 80, 3.3],
       ['AK', 'gray', 'Cheese', 32, 180, 1.8],
       ['TX', 'black', 'Melon', 33, 172, 9.5],
       ['TX', 'red', 'Beans', 69, 150, 2.2]], dtype=object)
```

Pandas

```
>>> df['food']
```

	food
Jane	Steak
Niko	Lamb
Aaron	Mango
Penelope	Apple
Dean	Cheese
Christina	Melon
Cornelia	Beans

```
>>> df[['color', 'food', 'score']]
```

	color	food	score
Jane	blue	Steak	4.6
Niko	green	Lamb	8.3
Aaron	red	Mango	9.0
Penelope	white	Apple	3.3
Dean	gray	Cheese	1.8
Christina	black	Melon	9.5
Cornelia	red	Beans	2.2

```
df[['height', 'color']]
```

	height	color
Jane	165	blue
Niko	70	green
Aaron	120	red
Penelope	80	white
Dean	180	gray
Christina	172	black
Cornelia	150	red

```
>>> df.loc[['Niko', 'Penelope']]
```

	state	color	food	age	height	score
Niko	TX	green	Lamb	2	70	8.3
Penelope	AL	white	Apple	4	80	3.3

```
>>> df.loc['Niko':'Dean']
```

	state	color	food	age	height	score
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0
Penelope	AL	white	Apple	4	80	3.3
Dean	AK	gray	Cheese	32	180	1.8

```
>>> df.loc[:'Aaron']
```

	state	color	food	age	height	score
Jane	NY	blue	Steak	30	165	4.6
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0

Pandas

```
>>> df.loc[['Dean', 'Cornelia'], ['age', 'state', 'score']]
```

	age	state	score
Dean	32	AK	1.8
Cornelia	69	TX	2.2

```
>>> df.loc['Jane':'Penelope', ['state', 'color']]
```

	state	color
Jane	NY	blue
Niko	TX	green
Aaron	FL	red
Penelope	AL	white

```
>>> df.loc[['Penelope','Cornelia'], :]
```

	state	color	food	age	height	score
Penelope	AL	white	Apple	4	80	3.3
Cornelia	TX	red	Beans	69	150	2.2