REPUBLIQUE DE DJIBOUTI

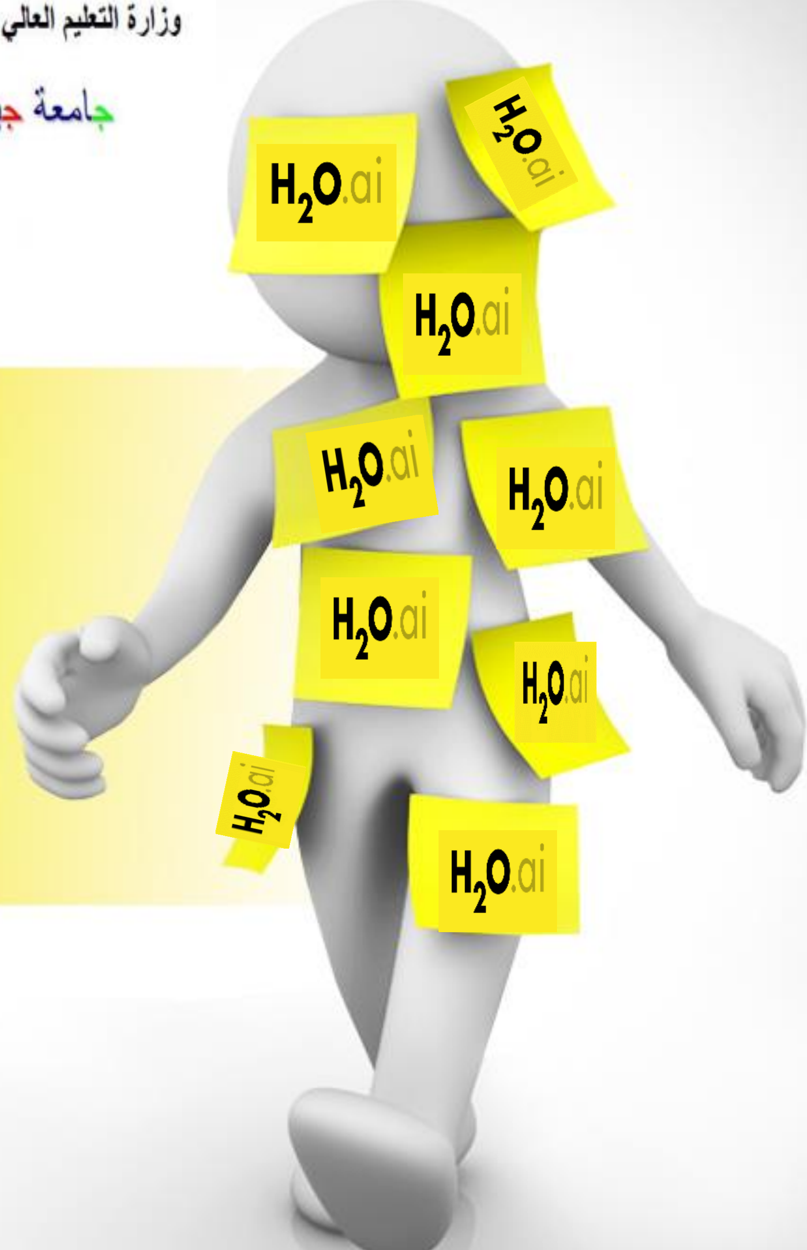
Ministère de l'Enseignement Supérieur et de la Recherche

UNIVERSITE DE DJIBOUTI

جمهورية جيبوتي

وزارة التعليم العالي والبحث العلمي

جامعة جيبوتي



Thème : Classification H2O

Master 2 Data Science

Encadré par :

M. Engelbert MEPHU NGUIFO

Réaliser par :

Ali HASSAN ALI

Nouradine ABDI MAHAMOUD

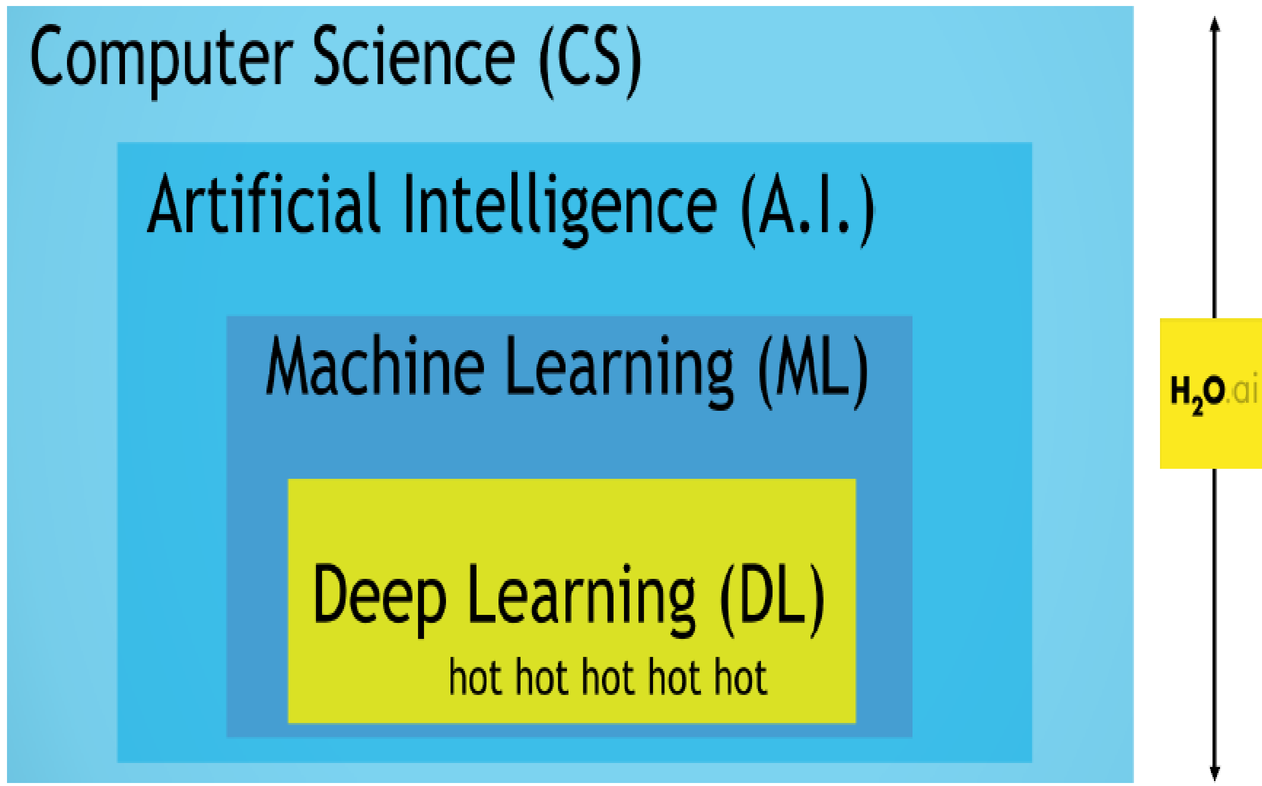
Plan

1. INTRODUCTION DE H2O
2. PRESENTATION DE H2O
3. ARCHITECTURE DE H2O
4. INSTALLATION DE H2O SOUS PYTHON
5. PRESENTATION DES ALGORIHTMES
6. AVANTAGE DE H2O
7. QU’EST-CE QUE LE MACHINE LEARNING ?
8. PRINCIPE DE BASE DE MACHINE LEARNING
9. METHODE DE SUPERVISE
10. H2O FLOW
11. OBJECTIFS D’APPRENTISSAGE
12. AutoML
13. KNN POUR LE TRAITEMENT DE BIG DATA
14. CONCLUSION

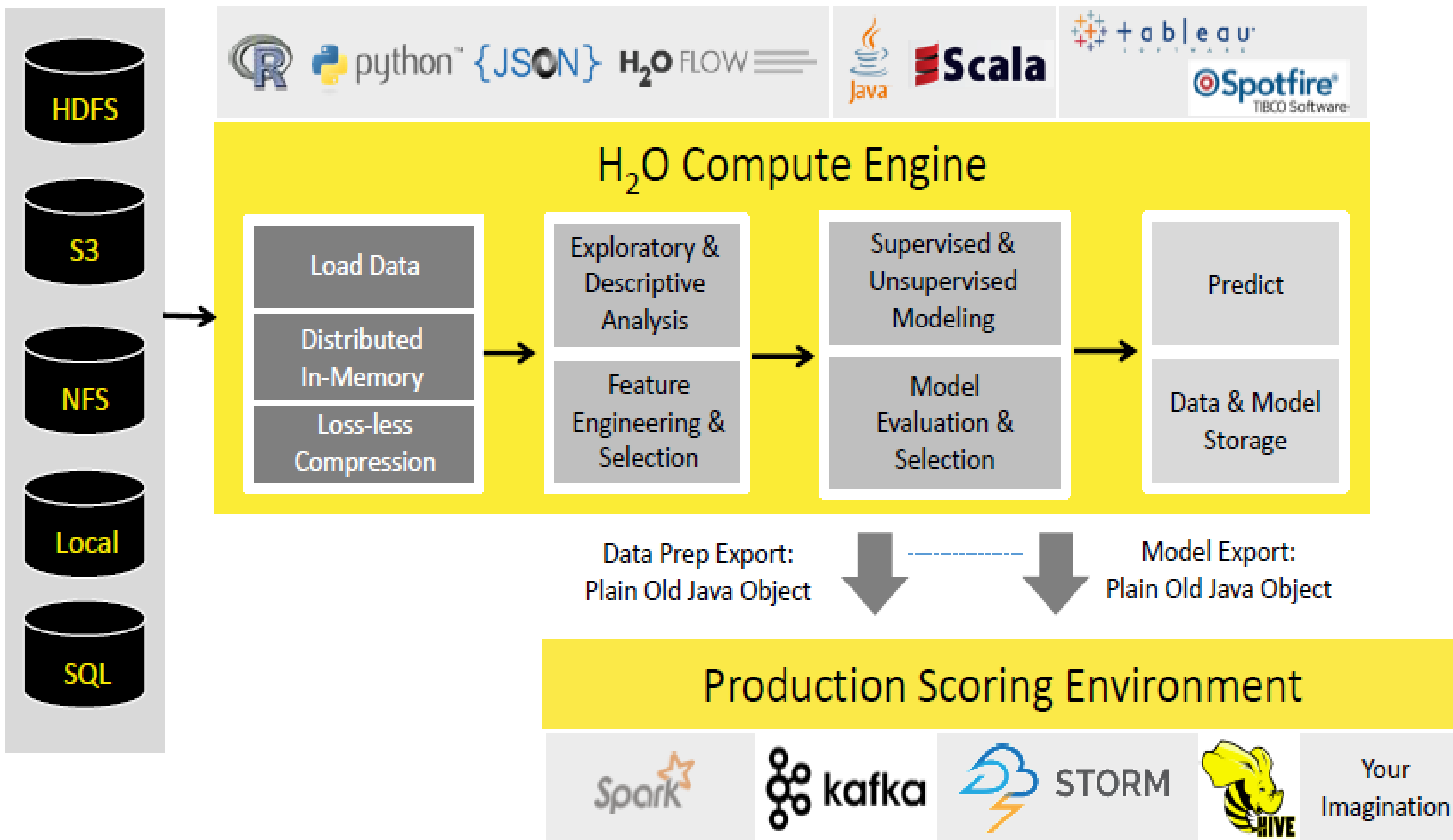
1. INTRODUCTION DE H2O

- H2O est un logiciel basé sur Java pour la modélisation de données et le calcul général.
- Application open source rapide, évolutive pour machine / apprentissage profond.
- Grâce à la compression en mémoire, H2O gère des milliards de lignes de données en mémoire, même sur un petit cluster,
- la plate-forme H2O comprend des interfaces pour Python, R, Scala, Java, JSON et CoffeeScript / JavaScript, ainsi qu'une interface Web intégrée, Flow.
- H2O est conçu pour fonctionner en mode autonome, sur Hadoop ou dans un cluster Spark

2. PRESENTATION DE H2O

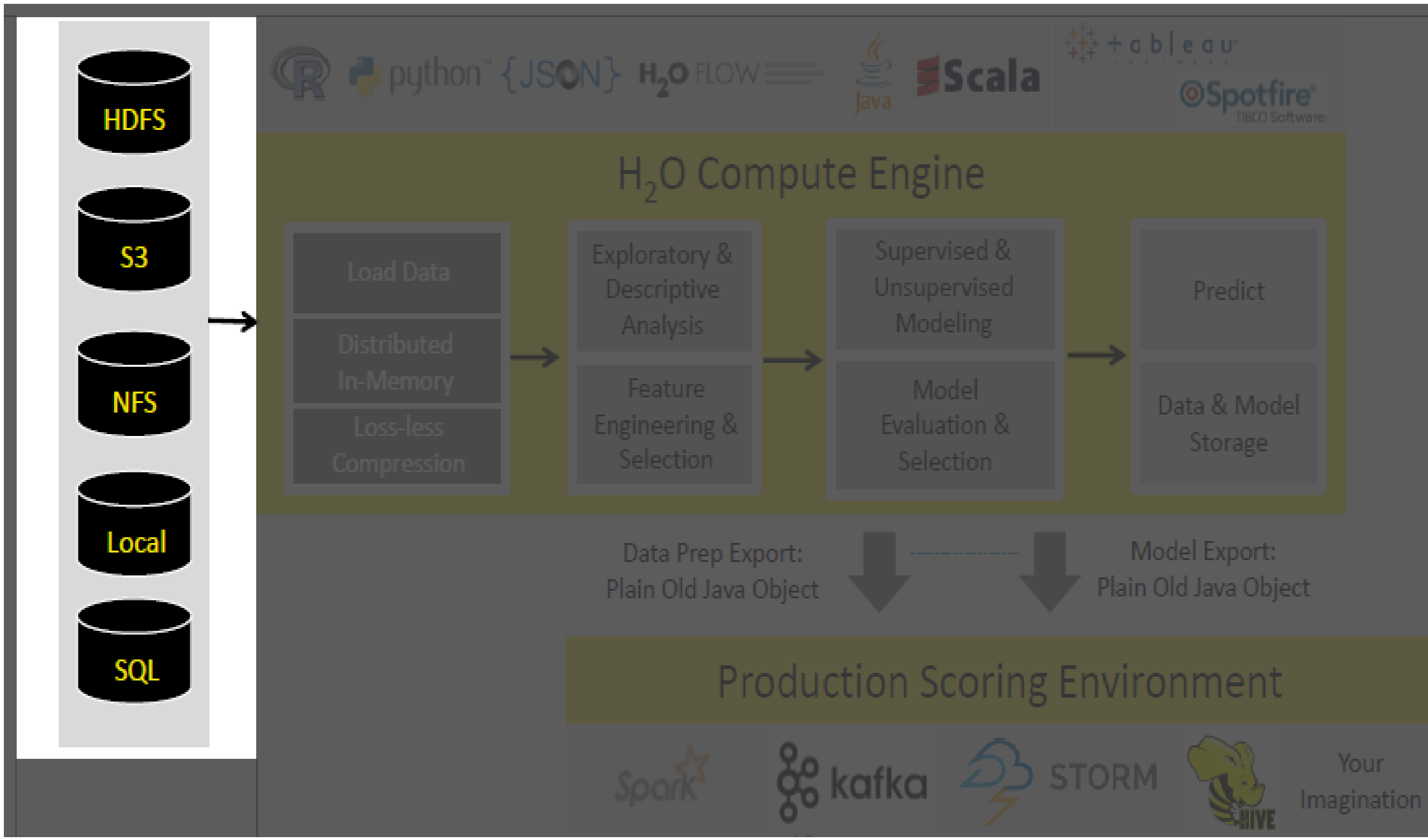


3. ARCHITECTURE DE H2O



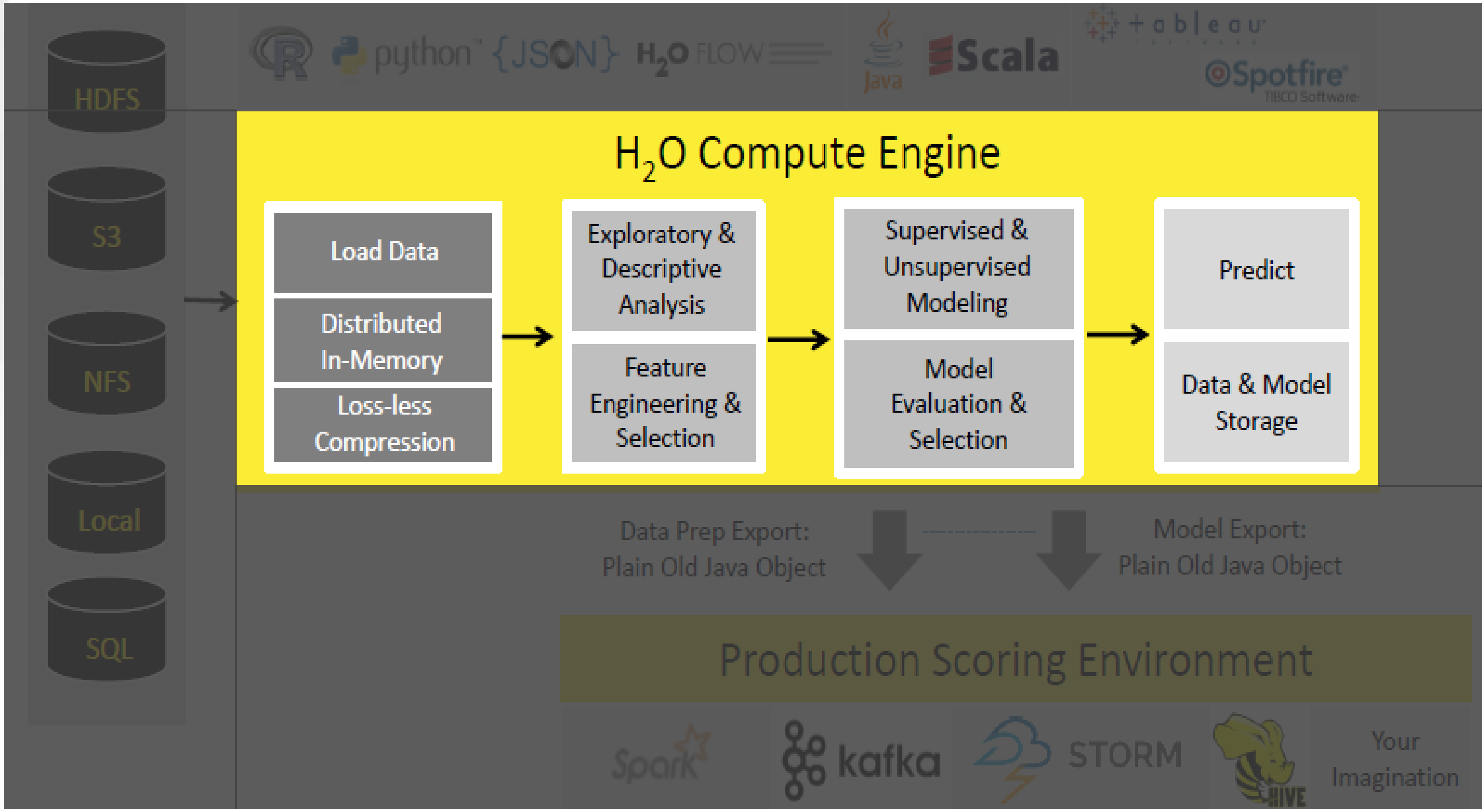
3. ARCHITECTURE DE H2O

Importer des données à partir de plusieurs sources



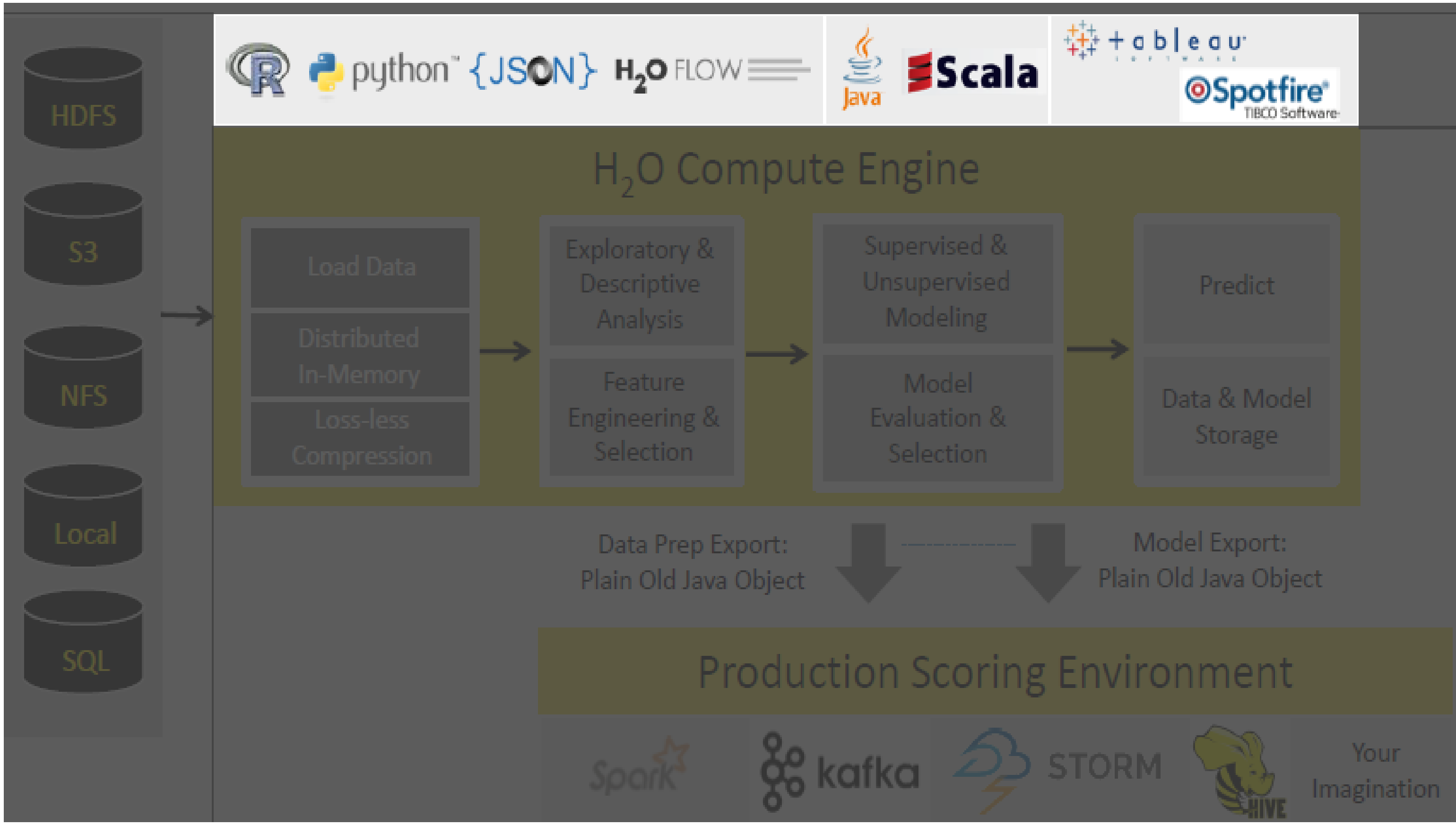
3. ARCHITECTURE DE H2O

Moteur de calcul rapide, évolutif et distribué
écrit en Java



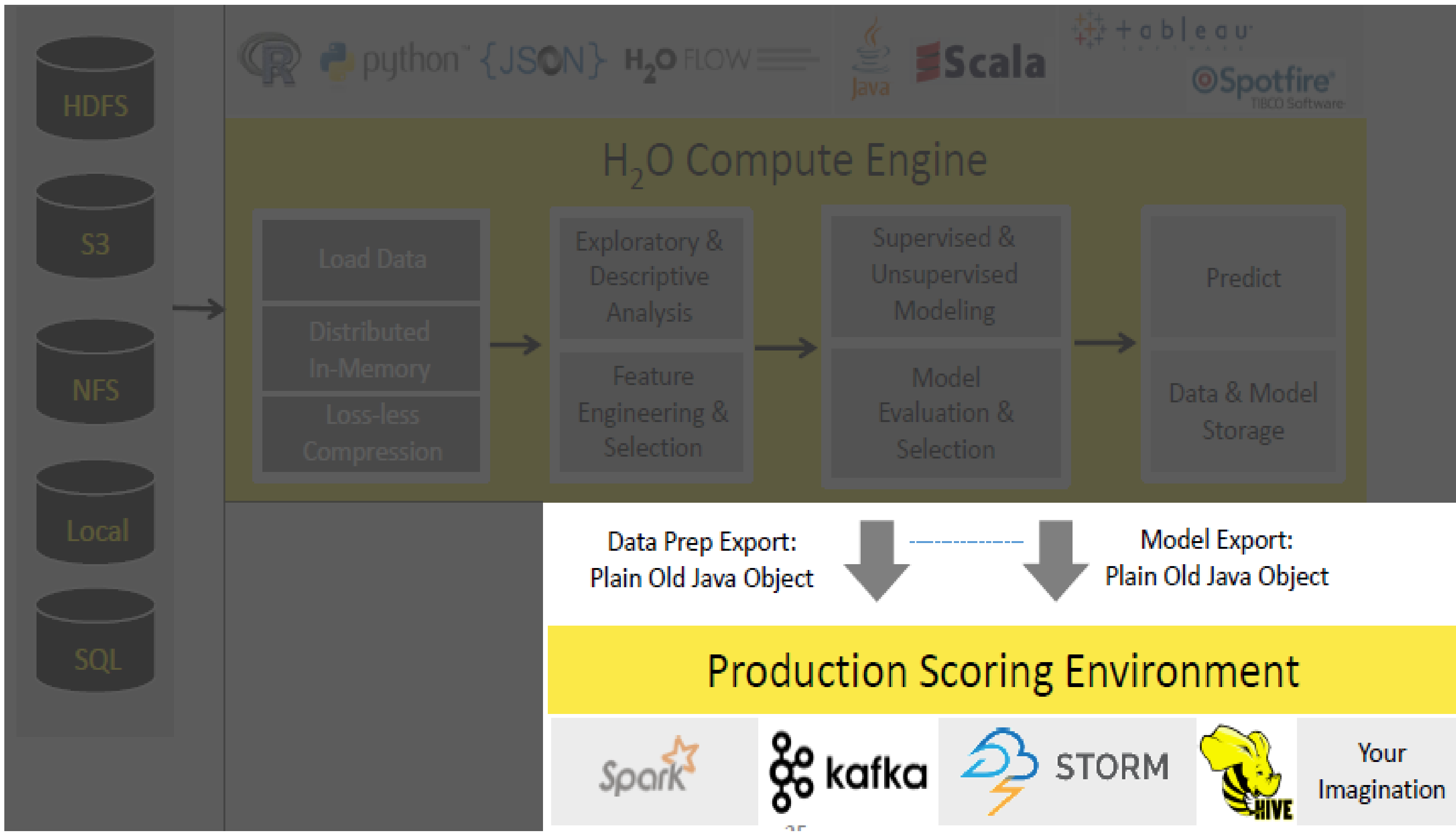
3. ARCHITECTURE DE H2O

Interfaces multiples



3. ARCHITECTURE DE H2O

Exporter des modèles autonomes pour la production



4. INSTALLATION DE H2O SOUS PYTHON

- Le moyen le plus simple d'installer directement H2O consiste à utiliser un package Python. Pour charger un package H2O récent à partir de PyPI, exécutez:

```
Anaconda Prompt (Anaconda3)

(base) C:\Users\USER>pip installer h2o
```

5. PRESENTATIONS DES ALGORITHMES

Supervised Learning

Statistical Analysis

- **Generalized Linear Models:** Binomial, Gaussian, Gamma, Poisson and Tweedie
- **Naïve Bayes**

Ensembles

- **Distributed Random Forest:** Classification or regression models
- **Gradient Boosting Machine:** Produces an ensemble of decision trees with increasing refined approximations

Deep Neural Networks

- **Deep learning:** Create multi-layer feed forward neural networks starting with an input layer followed by multiple layers of nonlinear transformations

Unsupervised Learning

Clustering

- **K-means:** Partitions observations into k clusters/groups of the same spatial size. Automatically detect optimal k

Dimensionality Reduction

- **Principal Component Analysis:** Linearly transforms correlated variables to independent components
- **Generalized Low Rank Models:** extend the idea of PCA to handle arbitrary data consisting of numerical, Boolean, categorical, and missing data

Anomaly Detection

- **Autoencoders:** Find outliers using a nonlinear dimensionality reduction using deep learning

6. AVANTAGE DE H2O

- Fondation pour le calcul d'algorithmes distribués en mémoire - Trames de données distribuées et compression en colonnes.
- Tous les algorithmes sont distribués dans H2O: GBM, GLM, DRF, Deep Learning et plus. Itérations de réduction de map-reduce.
- Fonctionnalités «out-of-box» pour tous les algorithmes et interface uniforme dans tous les langages: R, Python, Java
- Conçu pour toutes les tailles d'ensembles de données, en particulier les données volumineuses
- Code Java hautement optimisé pour les exportations de modèles
- Expertise interne pour tous les algorithmes

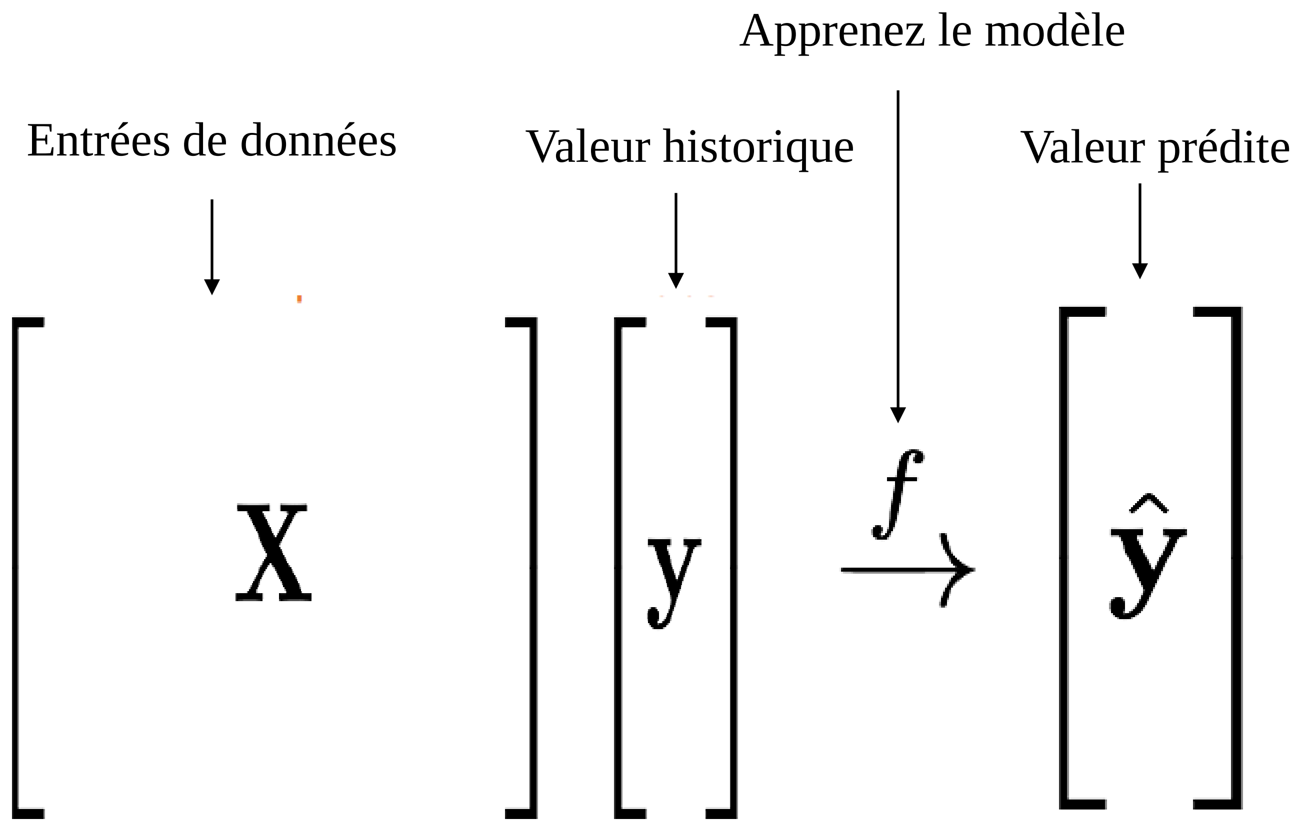
7. QU'EST-CE QUE LE MACHINE LEARNING ?

A field of study that gives computers the ability to learn without being explicitly programmed.

-- Arthur Samuel, 1959

8. PRINCIPE DE BASE DE MACHINE LEARNING

Apprendre à partir des données

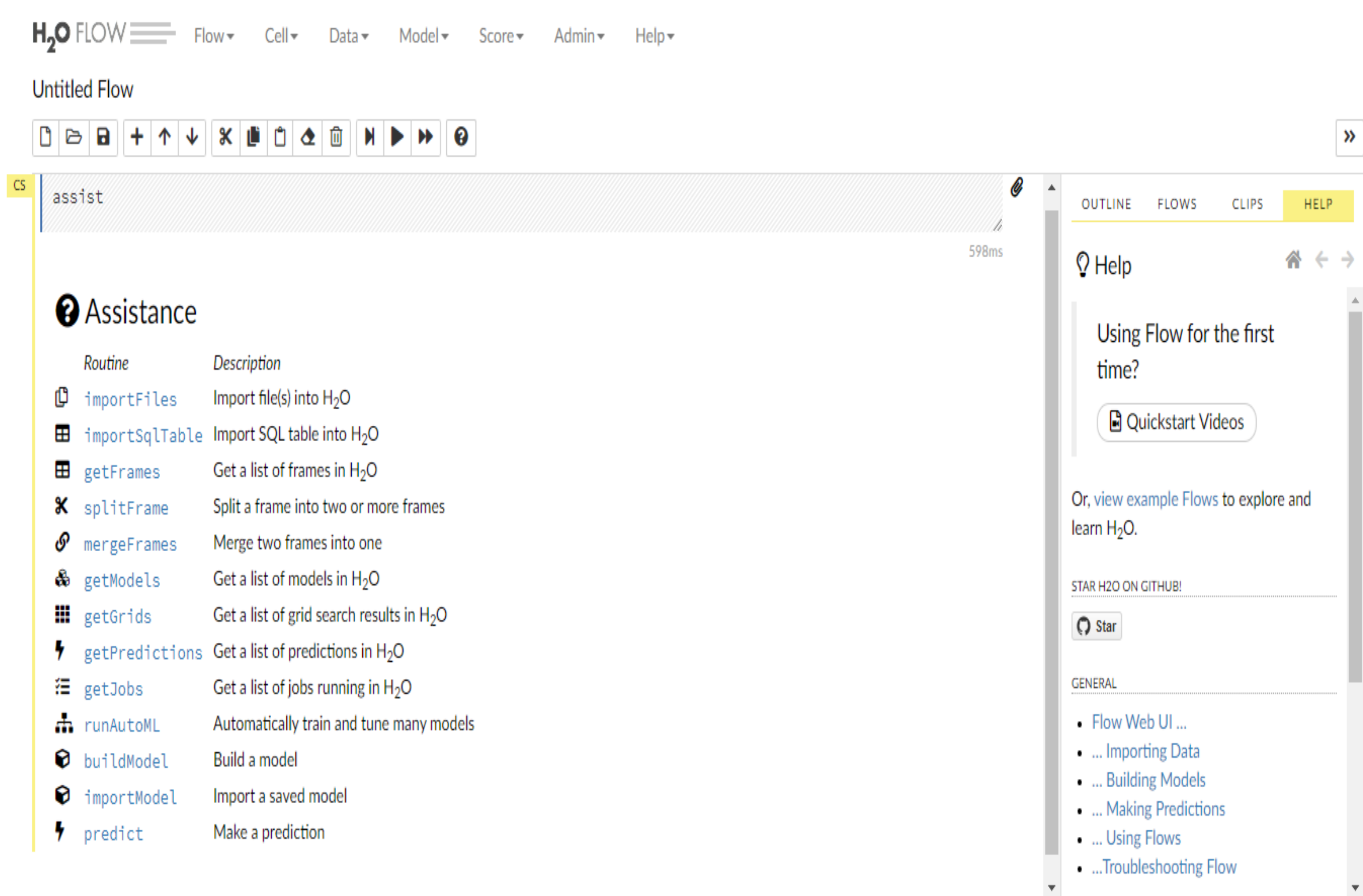


9. METHODE DE SUPERVISE

Classification: Un client fera-t-il un achat? Oui ou non

H2O algos:
Penalized linear models
Naive Bayes
Random forest
Gradient increase
Neural networks
Stacked sets

10. H2O FLOW



H2O FLOW
Flow
Cell
Data
Model
Score
Admin
Help
Untitled Flow
CS
assist
598ms
Assistance
Routine
Description
importFiles
Import file(s) into H2O
importSqlTable
Import SQL table into H2O
getFrames
Get a list of frames in H2O
splitFrame
Split a frame into two or more frames
mergeFrames
Merge two frames into one
getModels
Get a list of models in H2O
getGrids
Get a list of grid search results in H2O
getPredictions
Get a list of predictions in H2O
getJobs
Get a list of jobs running in H2O
runAutoML
Automatically train and tune many models
buildModel
Build a model
importModel
Import a saved model
predict
Make a prediction
OUTLINE
FLOWS
CLIPS
HELP
Help
Using Flow for the first time?
Quickstart Videos
Or, view example Flows to explore and learn H2O.
STAR H2O ON GITHUB!
Star
GENERAL
Flow Web UI ...
... Importing Data
... Building Models
... Making Predictions
... Using Flows
...Troubleshooting Flow

11. OBJECTIFS D’APPRENTISSAGE

- Démarrez et connectez-vous à un cluster H2O local depuis Python.
- Importez des données à partir de cadres de données Python, de fichiers locaux ou Web.
- Effectuer une transformation et une exploration de données de base.
- Former des modèles de classification à l'aide de divers algorithmes d'apprentissage H2Omachine.
- Évaluez les modèles et faites des prédictions.
- Améliorez les performances en optimisant et en empilant.

CLUSTER H2O LOCAL

Importer le module
H2O

```
In [1]: #importation du package
        import h2o
```

```
In [2]: #démarrage H2O
        h2o.init()
```

Démarrer un cluster H2O local
signifie
en utilisant TOUTES les
ressources du processeur

```
In [2]: #démarrage H2O
        h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321 ..... not found.
Attempting to start a local H2O server...
; Java HotSpot(TM) Client VM (build 25.45-b02, mixed mode)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\h2o\backend\server.py:385: UserWarning:   You have a 32-bit version of Java. H2O wor
ks best with 64-bit Java.
  Please download the latest 64-bit Java SE JDK from Oracle.

  warn("  You have a 32-bit version of Java. H2O works best with 64-bit Java.\n"
```

```
  Starting server from C:\ProgramData\Anaconda3\lib\site-packages\h2o\backend\bin\h2o.jar
  Ice root: C:\Users\ADMINI~1\AppData\Local\Temp\tmpk3gp1b0m
  JVM stdout: C:\Users\ADMINI~1\AppData\Local\Temp\tmpk3gp1b0m\h2o_Administrateur_started_from_python.out
  JVM stderr: C:\Users\ADMINI~1\AppData\Local\Temp\tmpk3gp1b0m\h2o_Administrateur_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

H2O_cluster_uptime:	02 secs
H2O_cluster_timezone:	Europe/Paris
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.32.0.2
H2O_cluster_version_age:	2 months and 11 days
H2O_cluster_name:	H2O_from_python_Administrateur_nasxk8
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	247.5 Mb
H2O_cluster_total_cores:	0
H2O_cluster_allowed_cores:	0
H2O_cluster_status:	accepting new members, healthy
H2O_connection_url:	http://127.0.0.1:54321
H2O_connection_proxy:	{"http": null, "https": null}
H2O_internal_security:	False
H2O_API_Extensions:	Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
Python_version:	3.7.4 final

Informations du cluster

IMPORTATION DE DONNÉES DANS H2O

Direction des données

Prétraitement des données

```
In [3]: #changer le répertoire courant
        import os
        os.chdir("C:/Users/Administrateur/Documents/H2O ipynb")
```

```
In [4]: #chargement des données
        cardio = h2o.import_file("cardio.csv")

Parse progress: |██████████████████████████████████████████████| 100%
```

Importer des données dans le cluster H2O

```
In [5]: #affichage des premières valeurs
        print(cardio.head(10))
```

id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	18393	2	168	62	110	80	1	1	0	0	1	0
1	20228	1	156	85	140	90	3	1	0	0	1	1
2	18857	1	165	64	130	70	3	1	0	0	0	1
3	17623	2	169	82	150	100	1	1	0	0	1	1
4	17474	1	156	56	100	60	1	1	0	0	0	0
8	21914	1	151	67	120	80	2	2	0	0	0	0
9	22113	1	157	93	130	80	3	1	0	0	1	0
12	22584	2	178	95	130	90	3	3	0	0	1	1
13	17668	1	158	71	110	70	1	1	0	0	1	0
14	19834	1	164	68	110	60	1	1	0	0	0	0

Variable cible

Type de donnée

```
In [6]: #affichage du type
        print(type(cardio))

<class 'h2o.frame.H2OFrame'>
```

```
In [7]: #dimension
        print(cardio.shape)

(70000, 13)
```

Dimension de donnée

Description de donnée

```
In [9]: # Resume de donnée
        cardio.describe()
```

```
Rows:70000
Cols:13
```

	id	age	gender	height	weight	ap_hi	ap_lo	
type	int	int	int	int	real	int	int	
mins	0.0	10798.0	1.0	55.0	10.0	-150.0	-70.0	
mean	49972.41989999998	19468.86581428571	1.3495714285714273	164.35922857142862	74.20568999999999	128.81728571428567	96.63041428571428	1.36687
maxs	99999.0	23713.0	2.0	250.0	200.0	16020.0	11000.0	
sigma	28851.30232317291	2467.2516672414017	0.4768380155828637	8.210126364538034	14.395756678511377	154.01141945609132	188.4725302963903	0.68025
zeros	1	0	0	0	0	0	21	
missing	0	0	0	0	0	0	0	
0	0.0	18393.0	2.0	168.0	62.0	110.0	80.0	
1	1.0	20228.0	1.0	156.0	85.0	140.0	90.0	
2	2.0	18857.0	1.0	165.0	64.0	130.0	70.0	
3	3.0	17623.0	2.0	169.0	82.0	150.0	100.0	
4	4.0	17474.0	1.0	156.0	56.0	100.0	60.0	
5	8.0	21914.0	1.0	151.0	67.0	120.0	80.0	
6	9.0	22113.0	1.0	157.0	93.0	130.0	80.0	
7	12.0	22584.0	2.0	178.0	95.0	130.0	90.0	
8	13.0	17668.0	1.0	158.0	71.0	110.0	70.0	
9	14.0	19834.0	1.0	164.0	68.0	110.0	60.0	

```
In [11]: cardio["cardio"] = cardio["cardio"].asfactor()
```

```
In [12]: #cardio est bien un type facteur
         cardio['cardio'].isfactor()
```

```
Out[12]: [True]
```

```
In [13]: #nombre de niveaux (modalités) de "diabete"
         cardio['cardio'].levels()
```

```
Out[13]: [['0', '1']]
```

Subdivision en échantillons d'apprentissage et de test

```
In [14]: #subdivision
         cardioTrain,cardioTest = cardio.split_frame(ratios=[0.8],seed=1)

In [15]: #vérification train
         cardioTrain.shape

Out[15]: (56007, 13)

In [16]: #vérification test
         cardioTest.shape

Out[16]: (13993, 13)
```

Diviser l'ensemble de données afin que nous puissions mesurer les performances,

MODÈLES DE CLASSIFICATION

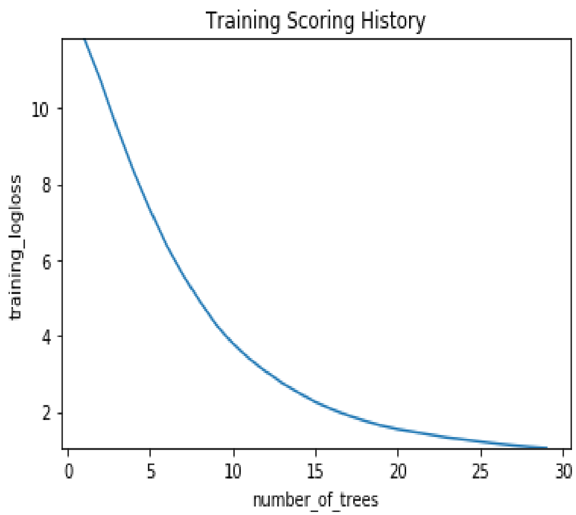
Modèle 1 : Random forest

```
In [17]: #random forest
         from h2o.estimators import H2ORandomForestEstimator

In [18]: x = cardioTrain.col_names[:-1]
         x

Out[18]: ['id',
          'age',
          'gender',
          'height',
          'weight',
          'ap_hi',
          'ap_lo',
          'cholesterol',
          'gluc',
          'smoke',
          'alco',
          'active']

In [19]: y=cardio.col_names[-1]
         y

Out[19]: 'cardio'

In [20]: #instanciation
         rf = H2ORandomForestEstimator(seed=1, nfolds=5, model_id="rf",
                   ntrees=200,
                   max_depth=30,
                   stopping_rounds=2,
                   stopping_tolerance=0.01,
                   score_each_iteration=True)
```

```
In [21]: #apprentissage
rf.train(x=x, y=y, training_frame=cardioTrain)
```

drf Model Build progress: |██| 100%

```
In [22]: #evolution de l'apprentissage
rf.plot()
```



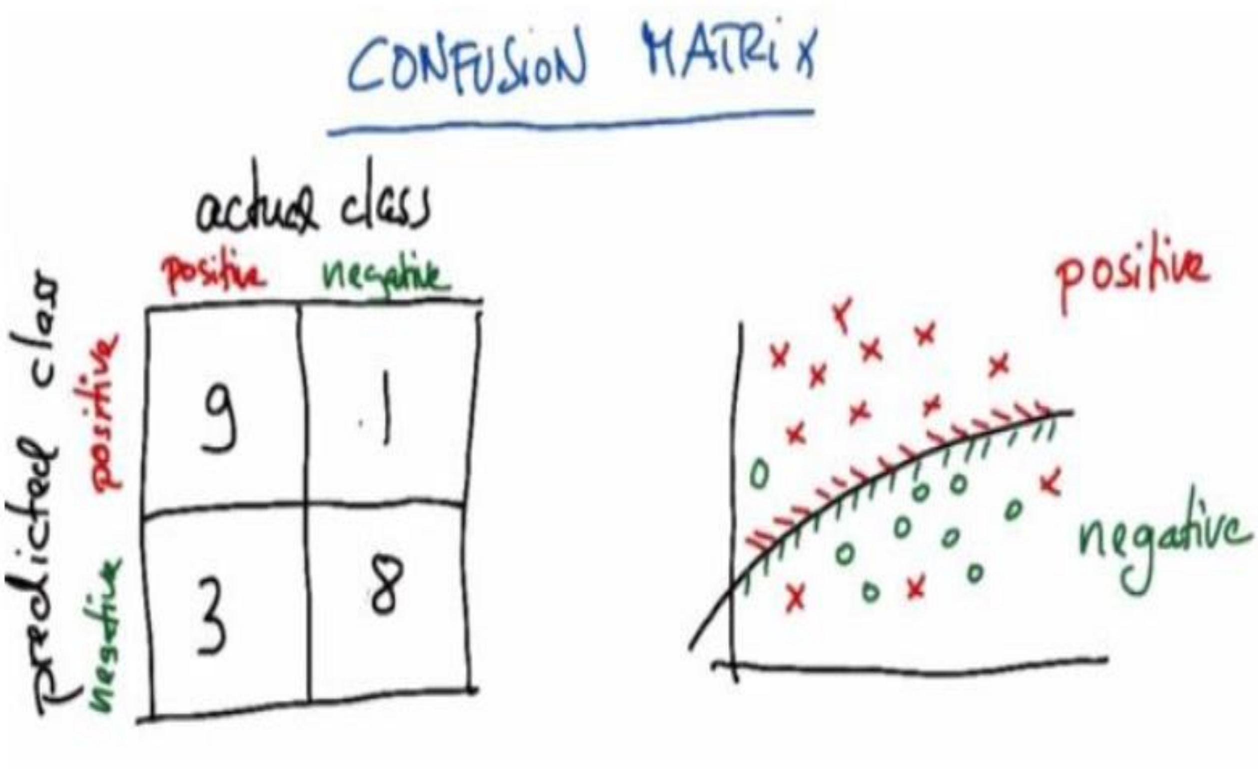

```
In [23]: #résumé
rf.summary()
```

Model Summary:

	number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
0	29.0	29.0	4029837.0	30.0	30.0	30.0	10438.0	11578.0	11052.207

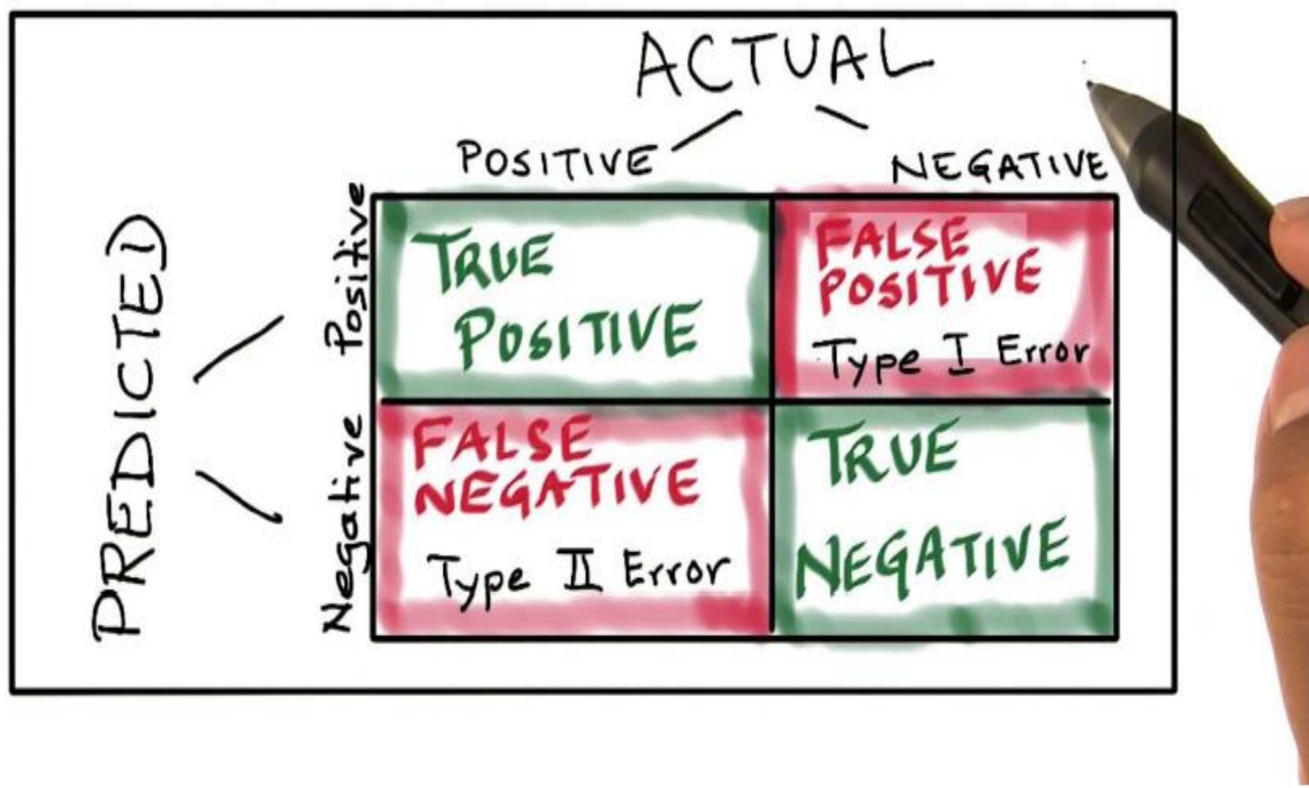
Out[23]:

Performance de classification-Matrice de confusion



MATRICE DE CONFUSION

Confusion Matrix



```
In [24]: #affichage
         rf.show()
```

```
Model Details
=============
H2ORandomForestEstimator :  Distributed Random Forest
Model Key:  rf


Model Summary:
```

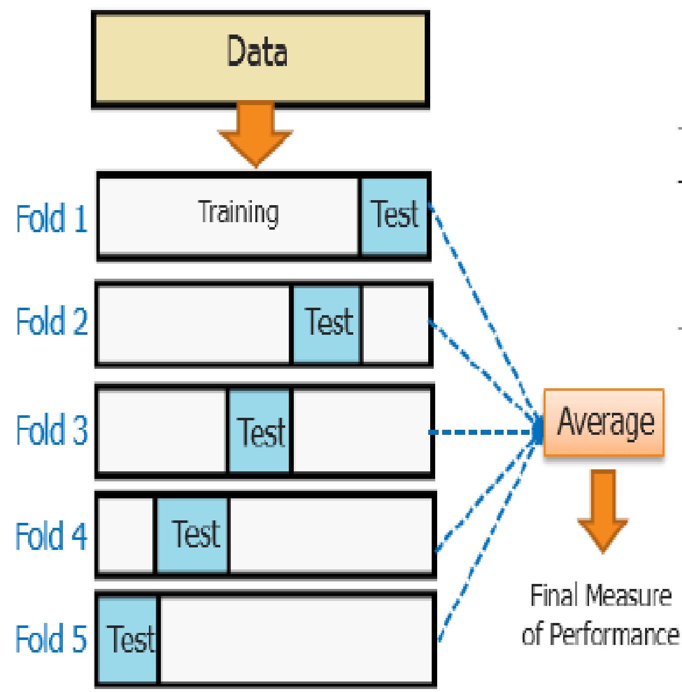
	number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
0	29.0	29.0	4029837.0	30.0	30.0	30.0	10438.0	11578.0	11052.207

```
ModelMetricsBinomial: drf
** Reported on train data. **

MSE: 0.2036909877312715
RMSE: 0.4513213796523177
LogLoss: 1.0472604630956661
Mean Per-Class Error: 0.29515391831013504
AUC: 0.7605602495399105
AUCPR: 0.7447028114740423
Gini: 0.521120499079821

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.36365493542527855:
```

		0	1	Error	Rate
0	0	15236.0	12739.0	0.4554	(12739.0/27975.0)
1	1	5221.0	22811.0	0.1863	(5221.0/28032.0)
2	Total	20457.0	35550.0	0.3207	(17960.0/56007.0)

Résumé du modèle

VALIDATION CROISÉE



- → Technique to validate models/classifiers
- → Method to estimate how accurately the model generalizes to unseen data i.e., how well it performs/predicts
- → K-fold CV
  - » Most popular
  - » k is typically set to 10
  - » Every sample/record is used both in training and test sets

Cross-Validation Metrics Summary:

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	accuracy	0.6965689	0.00500723	0.68885326	0.696102	0.7024347	0.6964141	0.6990405
1	auc	0.776263	0.004009069	0.7756584	0.7730693	0.77467716	0.77467036	0.78323966
2	aucpr	0.7597324	0.0045673577	0.7624629	0.7576347	0.7533	0.76004446	0.76522
3	err	0.3034311	0.00500723	0.31114677	0.30389795	0.29756528	0.3035859	0.30095956
4	err_count	3398.8	55.782616	3492.0	3407.0	3361.0	3378.0	3356.0
5	f0point5	0.6857879	0.0047237575	0.67857784	0.6858292	0.6918133	0.6867532	0.6859662
6	f1	0.72801626	0.004813609	0.72268105	0.72539693	0.72805244	0.72836924	0.73558146
7	f2	0.7758623	0.009927481	0.77291566	0.7698098	0.7682981	0.7753544	0.7929336
8	lift_top_group	1.7168936	0.034274522	1.7159022	1.7191662	1.6674864	1.7175714	1.7643415
9	logloss	0.6190586	0.011921326	0.61782265	0.6138325	0.6326343	0.6283477	0.602656
10	max_per_class_error	0.41858542	0.018358262	0.43287572	0.41042113	0.39257294	0.41845766	0.43859965
11	mcc	0.4039378	0.009937571	0.3893179	0.4013658	0.41261637	0.4025771	0.41381177
12	mean_per_class_accuracy	0.6964416	0.005059308	0.688799	0.69607353	0.702561	0.69586957	0.6989048
13	mean_per_class_error	0.3035584	0.005059308	0.31120095	0.30392647	0.29743895	0.30413043	0.3010952
14	mse	0.1934632	0.0018592428	0.19344145	0.19502705	0.19415678	0.19439487	0.19029588
15	pr_auc	0.7597324	0.0045673577	0.7624629	0.7576347	0.7533	0.76004446	0.76522
16	precision	0.66028196	0.0065765353	0.6520493	0.6617647	0.6695937	0.6615542	0.65644777
17	r2	0.2261432	0.007439009	0.22623402	0.21989174	0.22337154	0.22240287	0.23881578
18	recall	0.8114686	0.014946328	0.8104738	0.8025682	0.79769504	0.81019676	0.8364093
19	rmse	0.43984044	0.0021188757	0.4398198	0.44161868	0.44063225	0.44090235	0.43622914

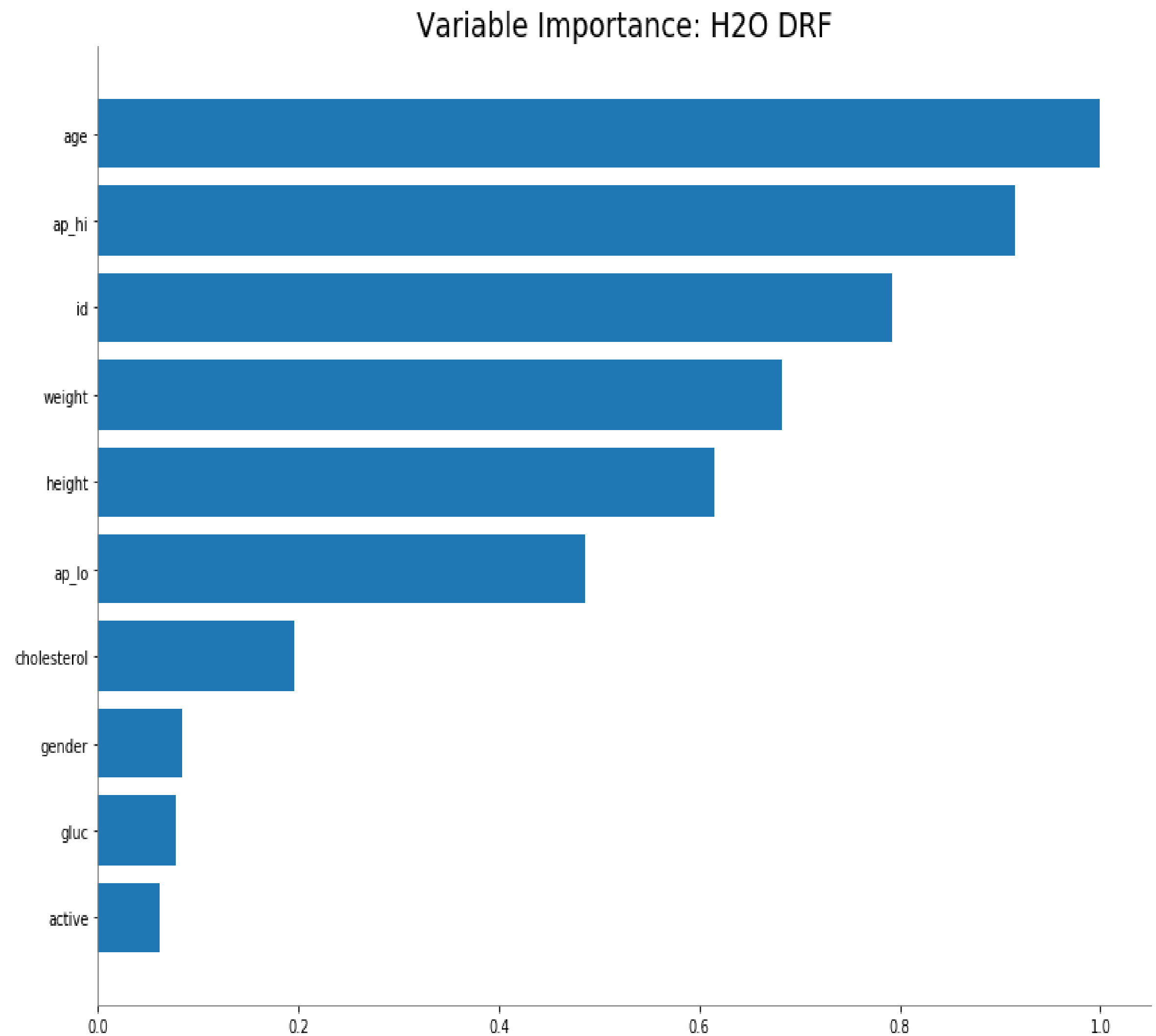
```
Scoring History:
```

	timestamp	duration	number_of_trees	training_rmse	training_logloss	training_auc	training_pr_auc	training_lift	training_classification_error
0	2021-01-28 20:20:30	37.490 sec	0.0	NaN	NaN	NaN	NaN	NaN	NaN
1	2021-01-28 20:20:30	37.736 sec	1.0	0.595357	11.820245	0.639459	0.607521	1.280344	0.502930
2	2021-01-28 20:20:31	37.908 sec	2.0	0.582589	10.722862	0.646468	0.615802	1.299885	0.500783
3	2021-01-28 20:20:31	38.074 sec	3.0	0.567442	9.468607	0.655637	0.624506	1.325145	0.501488
4	2021-01-28 20:20:31	38.232 sec	4.0	0.553045	8.329476	0.665199	0.634159	1.349055	0.373494
5	2021-01-28 20:20:31	38.389 sec	5.0	0.540499	7.311625	0.673797	0.644044	1.377476	0.377739
6	2021-01-28 20:20:31	38.648 sec	6.0	0.528180	6.373432	0.683367	0.653458	1.403100	0.368252
7	2021-01-28 20:20:32	38.883 sec	7.0	0.518854	5.599549	0.689959	0.661036	1.427178	0.379031
8	2021-01-28 20:20:32	39.141 sec	8.0	0.509595	4.909078	0.697837	0.669048	1.449207	0.365118
9	2021-01-28 20:20:32	39.319 sec	9.0	0.500646	4.295484	0.706597	0.678693	1.476650	0.359882
10	2021-01-28 20:20:32	39.481 sec	10.0	0.493900	3.798060	0.712705	0.685384	1.496073	0.356660
11	2021-01-28 20:20:32	39.653 sec	11.0	0.488737	3.402248	0.717434	0.691288	1.514332	0.355083
12	2021-01-28 20:20:33	39.852 sec	12.0	0.484087	3.063719	0.722092	0.696603	1.530289	0.352601
13	2021-01-28 20:20:33	40.030 sec	13.0	0.479819	2.757080	0.726400	0.702362	1.548330	0.351775
14	2021-01-28 20:20:33	40.296 sec	14.0	0.476096	2.503598	0.730547	0.707242	1.563059	0.347842
15	2021-01-28 20:20:33	40.567 sec	15.0	0.472560	2.257751	0.734510	0.712770	1.583384	0.347303
16	2021-01-28 20:20:34	40.823 sec	16.0	0.469250	2.069434	0.738626	0.717696	1.599167	0.345253
17	2021-01-28 20:20:34	41.088 sec	17.0	0.466733	1.902069	0.741331	0.720765	1.608862	0.343442
18	2021-01-28 20:20:34	41.370 sec	18.0	0.464577	1.763658	0.743916	0.723345	1.615277	0.341257
19	2021-01-28 20:20:34	41.646 sec	19.0	0.462719	1.638882	0.745926	0.726268	1.628309	0.326548

Variable Importances:

	variable	relative_importance	scaled_importance	percentage
0	age	51419.511719	1.000000	0.200374
1	ap_hi	47065.410156	0.915322	0.183406
2	id	40776.289062	0.793012	0.158899
3	weight	35104.593750	0.682710	0.136797
4	height	31632.037109	0.615176	0.123265
5	ap_lo	25020.437500	0.486594	0.097501
6	cholesterol	10056.023438	0.195568	0.039187
7	gender	4304.793945	0.083719	0.016775
8	gluc	3964.414307	0.077099	0.015449
9	active	3127.366943	0.060821	0.012187
10	smoke	2423.509521	0.047132	0.009444
11	alco	1723.715454	0.033523	0.006717

```
In [26]: #importance - graphique
         rf.varimp_plot()
```



In [27]:
```
#evaluation
rf.model_performance(cardioTest)
```

```
ModelMetricsBinomial: drf
** Reported on test data. **

MSE: 0.1932603156185623
RMSE: 0.4396138255543862
LogLoss: 0.61689784771367
Mean Per-Class Error: 0.27953074494813557
AUC: 0.7759074822259334
AUCPR: 0.7604561529082023
Gini: 0.5518149644518668

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.39063150461377766:
```

		0	1	Error	Rate
0	0	4245.0	2801.0	0.3975	(2801.0/7046.0)
1	1	1433.0	5514.0	0.2063	(1433.0/6947.0)
2	Total	5678.0	8315.0	0.3026	(4234.0/13993.0)

Évaluer les performances du modèle à l'aide de l'ensemble de test

In [28]:

```python
#prediction - de nouveau voir le seuil d'affectation
predRf = rf.predict(cardioTest).as_data_frame()
print(predRf.head(10))
```

```
drf prediction progress: |████████████████████████████████████████████████████████| 100%
   predict        p0        p1
0        1  0.206897  0.793103
1        0  0.655172  0.344828
2        0  0.724138  0.275862
3        1  0.467884  0.532116
4        1  0.176012  0.823988
5        1  0.052874  0.947126
6        1  0.258621  0.741379
7        0  0.775862  0.224138
8        0  0.671182  0.328818
9        0  0.741379  0.258621
```

In [29]:

```python
#scikit-learn
from sklearn import metrics
#F1-score
print(metrics.f1_score(cardioTest.as_data_frame()["cardio"],predRf.predict,pos_label=1))
```

```
0.7211862788773626
```

Modèle 2 : Gradient boosting

In [30]:
```
#gradient boosting
from h2o.estimators import H2OGradientBoostingEstimator
```

In [31]:
```
#instanciation
gb = H2OGradientBoostingEstimator(distribution="bernoulli", ntrees=50, max_depth=3, min_rows=2, learn_rate=0.2, nfolds=5)
```

In [32]:
```
#apprentissage
gb.train(x=x, y=y, training_frame=cardioTrain)
```

```
gbm Model Build progress: |██████████████████████████████████████████████████████| 100%
```

API pour d'autres algorithmes ML

Modèle 3 : Naive Bayes

In [41]:
```
#naive bayes
from h2o.estimators import H2ONaiveBayesEstimator
```

In [42]:
```
#instanciation
nb = H2ONaiveBayesEstimator(seed=100)
```

In [43]:
```
nb.train(x=x, y=y, training_frame=cardioTrain)
```

```
naivebayes Model Build progress: |██████████████████████████████████████████████████████| 100%
```

Modèle 4 : Perceptron simple

In [48]:
```python
#deep learning
from h2o.estimators import H2ODeepLearningEstimator
```

In [49]:
```python
#instanciation
ps = H2ODeepLearningEstimator(seed=100,epochs=1250,standardize=True,hidden=[],distribution="bernoulli")
```

In [50]:
```python
#apprentissage
ps.train(x=x, y=y, training_frame=cardioTrain)
```

```
deeplearning Model Build progress: |████████████████████████████████████████| 100%
```

In [51]:
```python
#structure du réseau
ps.summary()
```

```
Status of Neuron Layers: predicting cardio, 2-class classification, bernoulli distribution, CrossEntropy loss, 26 weights/biase
s, 4,6 KB, 39Â 295Â 113 training samples, mini-batch size 1
```

	layer	units	type	dropout	l1	l2	mean_rate	rate_rms	momentum	mean_weight	weight_rms	mean_bias	bias_rms
0	1	12	Input	0									
1	2	2	Softmax		0	0	0.00195111	0.00179441	0	-0.150255	1.86727	-0.111204	0.0887527

Out[51]:

12. AutoML

H2OAutoML

L’outil H2OAutoML (Automatic Machine Learning) correspond au même état d’esprit. Mieux même, il prétend détecter pour nous le meilleur modèle possible.

Modèle 5 : H2OAutoML

In [56]:
```python
#chargement de la classe
from h2o.automl import H2OAutoML
```

In [57]:
```python
#instanciation
aml = H2OAutoML(seed=100,nfolds=5,max_runtime_secs=180)
```

In [58]:
```python
#lancement des calculs
aml.train(x=x, y=y, training_frame=cardioTrain)
```

```
AutoML progress: |
20:23:05.31: AutoML: XGBoost is not available; skipping it.

20:23:27.531: GBM_1_AutoML_20210128_202305 [GBM def_1] failed: water.exceptions.H2OModelBuilderIllegalArgumentException: Illega
l argument(s) for GBM model: GBM_1_AutoML_20210128_202305_cv_1.  Details: ERRR on field: _ntrees: The tree model will not fit i
n the driver node's memory ( 770  B per tree x 10000 > 2,0 MB) - try decreasing ntrees and/or max_depth or increasing min_rows!

20:23:31.242: GBM_2_AutoML_20210128_202305 [GBM def_2] failed: water.exceptions.H2OModelBuilderIllegalArgumentException: Illega
l argument(s) for GBM model: GBM_2_AutoML_20210128_202305_cv_1.  Details: ERRR on field: _ntrees: The tree model will not fit i
n the driver node's memory (1,2 KB per tree x 10000 > 11,7 MB) - try decreasing ntrees and/or max_depth or increasing min_rows!

20:23:33.742: GBM_3_AutoML_20210128_202305 [GBM def_3] failed: water.exceptions.H2OModelBuilderIllegalArgumentException: Illega
l argument(s) for GBM model: GBM_3_AutoML_20210128_202305_cv_1.  Details: ERRR on field: _ntrees: The tree model will not fit i
n the driver node's memory (2,1 KB per tree x 10000 > 13,7 MB) - try decreasing ntrees and/or max_depth or increasing min_rows!
```

```
20:23:36.576: GBM_4_AutoML_20210128_202305 [GBM def_4] failed: water.exceptions.H2OModelBuilderIllegalArgumentException: Illegal argument(s) for GBM model: GBM_4_AutoML_20210128_202305_cv_1.  Details: ERRR on field: _ntrees: The tree model will not fit in the driver node's memory (4,9 KB per tree x 10000 > 28,5 MB) - try decreasing ntrees and/or max_depth or increasing min_rows!

| 100%
```

In [59]:
```python
#récupérer le tableau des modèles
lb = aml.leaderboard
```

In [60]:
```python
#nombre de modèles
print(lb.nrow) #35
```

```
5
```

In [61]:
```python
#afficher les modèles -- tri par défaut AUC pour le classement binaire
result = lb.head(rows=lb.nrow).as_data_frame()
result.loc[:,["model_id","auc"]]
```

Out[61]:

	model_id	auc
0	StackedEnsemble_BestOfFamily_AutoML_20210128_202305	0.783956
1	StackedEnsemble_AllModels_AutoML_20210128_202305	0.783908
2	GBM_5_AutoML_20210128_202305	0.765696
3	DRF_1_AutoML_20210128_202305	0.735211
4	GLM_1_AutoML_20210128_202305	0.712019

```
In [62]: #meilleur modèle
aml.leader
```

```
Model Details
=============
H2OStackedEnsembleEstimator :  Stacked Ensemble
Model Key:  StackedEnsemble_BestOfFamily_AutoML_20210128_202305

No model summary for this model

ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.15071948299201818
RMSE: 0.38822607201477105
LogLoss: 0.4751235241757313
Null degrees of freedom: 9939
Residual degrees of freedom: 9936
Null deviance: 13779.853592633595
Residual deviance: 9445.455660613537
AIC: 9453.455660613537
AUC: 0.8856432242227588
AUCPR: 0.8870851323966201
Gini: 0.7712864484455175

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.4635565176469616:
```

		0	1	Error	Rate
0	0	3934.0	1055.0	0.2115	(1055.0/4989.0)
1	1	872.0	4079.0	0.1761	(872.0/4951.0)
2	Total	4806.0	5134.0	0.1939	(1927.0/9940.0)

Maximum Metrics: Maximum metrics at their respective thresholds

	metric	threshold	value	idx
0	max f1	0.463557	0.808924	219.0
1	max f2	0.289765	0.869052	311.0
2	max f0point5	0.580201	0.828292	164.0
3	max accuracy	0.482159	0.806942	210.0
4	max precision	0.919440	1.000000	0.0
5	max recall	0.122218	1.000000	394.0
6	max specificity	0.919440	1.000000	0.0
7	max absolute_mcc	0.500040	0.613947	202.0
8	max min_per_class_accuracy	0.476800	0.805171	212.0
9	max mean_per_class_accuracy	0.482159	0.806924	210.0
10	max tns	0.919440	4989.000000	0.0
11	max fns	0.919440	4943.000000	0.0
12	max fps	0.103505	4989.000000	399.0
13	max tps	0.122218	4951.000000	394.0
14	max tnr	0.919440	1.000000	0.0
15	max fnr	0.919440	0.998384	0.0
16	max fpr	0.103505	1.000000	399.0
17	max tpr	0.122218	1.000000	394.0

```
In [63]: #prédiction sur le test
         predAml = aml.predict(cardioTest).as_data_frame()
         print(predAml.head(10))
```

```
stackedensemble prediction progress: |████████████████████████████████████████| 100%
   predict        p0        p1
0        1  0.380239  0.619761
1        0  0.621306  0.378694
2        0  0.694760  0.305240
3        1  0.561358  0.438642
4        1  0.299229  0.700771
5        1  0.252403  0.747597
6        1  0.380064  0.619936
7        1  0.523493  0.476507
8        1  0.595332  0.404668
9        1  0.511398  0.488602
```

```
In [64]: #F1-score
         print(metrics.f1_score(cardioTest.as_data_frame()["cardio"],predAml.predict,pos_label=1))
```

```
0.7310084825636193
```

ML Méthode - Cardio Dataset	F1-Score
Random Forest	0.7212
Gradient Boosting	0.7387
Naïve Bayes	0.6871
Simple Perceptron	0.7326
H2OAutoML	0.7310

13. KNN POUR TRAITE LE BIG DATA

une nouvelle méthode kNN pour traiter le Big Data.

D'abord à un clustering de k-means pour séparer l'ensemble de données en plusieurs parties.

- ❖ Processus d’apprentissage

  Algorithme de LSC

- ❖ Processus de test

  Algorithme de LC-KNN

IMPLEMENTATION DE L'ALGORITHME DE LSC

Subdivision en échantillons d'apprentissage et de test

In [14]:
```
#subdivision
cardioTrain,cardioTest = cardio.split_frame(ratios=[0.8],seed=1)
```

In [15]:
```
#vérification train
cardioTrain.shape
```

Out[15]:
```
(56007, 13)
```

In [16]:
```
#vérification test
cardioTest.shape
```

Out[16]:
```
(13993, 13)
```

IMPLEMENTATION DES ALGORITHMES

ALGORITHME DE LSC

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics.pairwise import pairwise_distances
import scipy


def find_p(X, start=1, end=10):
```

```
def find_p(X, start=1, end=10):

    min_size=[]
    number_of_clusters=[]
    for i in range(start,end+1):
        min_size.append(i)
        number_of_clusters.append(KMeans(n_clusters=i).fit(X).inertia_)
    _, ax=plt.subplots()

    ax.set(ylabel='Inertia', xlabel='Number of clusters', title='The elbow method')
    plt.xticks(np.arange(start,end, 1))
    plt.plot(min_size,number_of_clusters)
    plt.show()
```

```
def get_Landmarks(X, p, method="random"):
    if method=="random":
        N = len(X)
        perm= np.random.permutation(np.arange(N))
        print(perm)
        landmarks = X[perm[:p],:]
        return landmarks
    else:
        kmeans_model=KMeans(n_clusters=p).fit(X)
        return kmeans_model.cluster_centers_


def gaussian_kernel(dist_mat, bandwidth):
    return np.exp(-dist_mat / (2*bandwidth**2))

def compose_Sparse_ZHat_Matrix(X, landmarks, bandwidth, r):
    dist_mat=pairwise_distances(X,landmarks)
    sim_mat=gaussian_kernel(dist_mat, bandwidth)

    Zhat = np.zeros(sim_mat.shape)

    for i in range(Zhat.shape[0]):
        #may need j.sort.selectperm
        top_Landmarks_indices = np.argsort(-sim_mat[i,:])[:r]
        top_Landmarks_coefs = sim_mat[i,top_Landmarks_indices]
        top_Landmarks_coefs /= np.sum(top_Landmarks_coefs)
        Zhat[i, top_Landmarks_indices] = top_Landmarks_coefs
    #May be wrong
    diagm=np.sum(Zhat, axis=0)**(-1/2)
    return diagm*Zhat
```

```
def LSC_Clustering(X, n_clusters, n_landmarks, method, non_zero_landmark_weights, bandwidth):
    landmarks = get_Landmarks(X, n_landmarks, method)
    Zhat = compose_Sparse_ZHat_Matrix(X, landmarks, bandwidth, non_zero_landmark_weights)
    svd_result = np.linalg.svd(Zhat, full_matrices=False)[0]
    clustering_result = KMeans(n_clusters=n_clusters).fit(svd_result)
    return clustering_result
```

14. CONCLUSION

Merci pour votre attention

Questions ?