# Practical Machine Learning

H2O.ai

# H2O

- **Herramienta de Machine Learning en Java**

## Herramientas/librerías que pueden ayudar

- **Grandes conjuntos de datos**

**H2O:**

- Es una plataforma machine Learning open-source desarrollada en Java que ofrece un gran conjunto de algoritmos de machine learning con alta capacidad de procesamiento (cluster)
- Gracias a su forma de comprimir y almacenar los datos, H2O es capaz de trabajar con millones de registros en un único ordenador (emplea todos sus cores) o en un cluster de muchos ordenadores

# H2O

- **Instalación**

**H2O**

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from pandas.plotting import scatter_matrix
from sklearn.feature_selection import VarianceThreshold
from sklearn.metrics import accuracy_score, auc, confusion_matrix, f1_score, precision_score, recall_score, roc_curve
from sklearn.feature_selection import SelectKBest
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import cross_val_score
```

```
!pip install h2o
```

In [33]:
```
import h2o
from h2o.estimators import H2ORandomForestEstimator
```

# H2O

- **Importar modelo (RandomForest) y levantar un cluster**

In [33]:
```python
import h2o
from h2o.estimators import H2ORandomForestEstimator
```

In [8]:
```python
# Creación de un cluster local H2O
# ---------------------------------------------------------------------------
h2o.init(ip = "localhost",
         # -1 indica que se empleen todos los cores disponibles.
         nthreads = -1,
         # Máxima memoria disponible para el cluster.
         max_mem_size = "4g")
```

# H2O

- **Borramos datos iniciales, cargamos datos y los pasamos a format H2O**

```
In [9]:  # Se eliminan los datos del cluster por si ya había sido iniciado.
         h2o.remove_all()

In [10]: df_bank=pd.read_csv('../bank-full.csv', sep=';')

In [11]: df_bank_h2o = h2o.H2OFrame(python_obj = df_bank, destination_frame = "df_bank_h2o")

         Parse progress: |████████████████████████████████████████████████| (done) 100%

In [12]: df_bank_h2o.head()
```

# H2O

- **datos.head()**

```
In [12]: df_bank_h2o.head()
```

Out[12]:

| age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pdays | previous | poutcome | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | -1 | 0 | unknown | no |
| 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | -1 | 0 | unknown | no |
| 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | -1 | 0 | unknown | no |
| 47 | blue-collar | married | unknown | no | 1506 | yes | no | unknown | 5 | may | 92 | 1 | -1 | 0 | unknown | no |
| 33 | unknown | single | unknown | no | 1 | no | no | unknown | 5 | may | 198 | 1 | -1 | 0 | unknown | no |
| 35 | management | married | tertiary | no | 231 | yes | no | unknown | 5 | may | 139 | 1 | -1 | 0 | unknown | no |
| 28 | management | single | tertiary | no | 447 | yes | yes | unknown | 5 | may | 217 | 1 | -1 | 0 | unknown | no |
| 42 | entrepreneur | divorced | tertiary | yes | 2 | yes | no | unknown | 5 | may | 380 | 1 | -1 | 0 | unknown | no |
| 58 | retired | married | primary | no | 121 | yes | no | unknown | 5 | may | 50 | 1 | -1 | 0 | unknown | no |
| 43 | technician | single | secondary | no | 593 | yes | no | unknown | 5 | may | 55 | 1 | -1 | 0 | unknown | no |

# H2O

- **datos.shape y col_names (no cols como en pandas)**

```
In [14]: df_bank_h2o.shape
Out[14]: (45211, 17)

In [16]: df_bank_h2o.col_names
Out[16]: ['age',
          'job',
          'marital',
          'education',
          'default',
          'balance',
          'housing',
          'loan',
          'contact',
          'day',
          'month',
          'duration',
          'campaign',
          'pdays',
          'previous',
          'poutcome',
          'y']
```

# H2O

- **datos.describe**

```
In [18]: df_bank_h2o.describe()
```

```
Rows:45211
Cols:17
```

| | age | job | marital | education | default | balance | housing | loan | contact | day | month | dur |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| type | int | enum | enum | enum | enum | int | enum | enum | enum | int | enum | |
| mins | 18.0 | | | | | -8019.0 | | | | 1.0 | | |
| mean | 40.93621021432809 | | | | | 1362.272057685082 | | | | 15.806418791886935 | | 258.163079781 |
| maxs | 95.0 | | | | | 102127.0 | | | | 31.0 | | 4 |
| sigma | 10.61876204097539 | | | | | 3044.7658291685234 | | | | 8.322476153044159 | | 257.527812265 |
| zeros | 0 | | | | | 3514 | | | | 0 | | |
| missing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 58.0 | management | married | tertiary | no | 2143.0 | yes | no | unknown | 5.0 | may | |
| 1 | 44.0 | technician | single | secondary | no | 29.0 | yes | no | unknown | 5.0 | may | |
| 2 | 33.0 | entrepreneur | married | secondary | no | 2.0 | yes | yes | unknown | 5.0 | may | |
| 3 | 47.0 | blue-collar | married | unknown | no | 1506.0 | yes | no | unknown | 5.0 | may | |
| 4 | 33.0 | unknown | single | unknown | no | 1.0 | no | no | unknown | 5.0 | may | |
| 5 | 35.0 | management | married | tertiary | no | 231.0 | yes | no | unknown | 5.0 | may | |
| 6 | 28.0 | management | single | tertiary | no | 447.0 | yes | yes | unknown | 5.0 | may | |
| 7 | 42.0 | entrepreneur | divorced | tertiary | yes | 2.0 | yes | no | unknown | 5.0 | may | |

# H2O

- **Contadores con .table()**

```
In [19]: df_bank_h2o['marital'].table()
Out[19]:
```

| marital | Count |
|---|---|
| divorced | 5207 |
| married | 27214 |
| single | 12790 |

```
[3 rows x 2 columns]
```

# H2O

- **.split_frame**

**Muestreo**

```
In [32]: df_bank_train_h2o, df_bank_test_h2o = df_bank_h2o.split_frame(ratios=[0.8],destination_frames= ["df_bank_train_h2o",
                                                    "df_bank_test_h2o"],
                                                    seed = 123)
```

```
H2OGradientBoostingEstimator(fold_assignment = assignment_type, nfolds = 5, seed = 1234)
```

```
In [29]: df_bank_train_h2o.shape

Out[29]: (36227, 17)
```

# H2O

- **Preparación de datos: .drop(‘y’)**

In [37]:
```
df_bank_train_h2o.drop('y')
```

Out[37]:

| age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | -1 | 0 | unknown |
| 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | -1 | 0 | unknown |
| 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | -1 | 0 | unknown |
| 47 | blue-collar | married | unknown | no | 1506 | yes | no | unknown | 5 | may | 92 | 1 | -1 | 0 | unknown |
| 33 | unknown | single | unknown | no | 1 | no | no | unknown | 5 | may | 198 | 1 | -1 | 0 | unknown |
| 35 | management | married | tertiary | no | 231 | yes | no | unknown | 5 | may | 139 | 1 | -1 | 0 | unknown |
| 28 | management | single | tertiary | no | 447 | yes | yes | unknown | 5 | may | 217 | 1 | -1 | 0 | unknown |
| 42 | entrepreneur | divorced | tertiary | yes | 2 | yes | no | unknown | 5 | may | 380 | 1 | -1 | 0 | unknown |
| 41 | admin. | divorced | secondary | no | 270 | yes | no | unknown | 5 | may | 222 | 1 | -1 | 0 | unknown |
| 29 | admin. | single | secondary | no | 390 | yes | no | unknown | 5 | may | 137 | 1 | -1 | 0 | unknown |

# H2O

- **Preparación de datos: .col_names**

```
In [46]: df_bank_train_h2o.drop('y').col_names
Out[46]: ['age',
          'job',
          'marital',
          'education',
          'default',
          'balance',
          'housing',
          'loan',
          'contact',
          'day',
          'month',
          'duration',
          'campaign',
          'pdays',
          'previous',
          'poutcome']
```

# H2O

- **Preparacion de datos: y**

**Modelos**

In [35]:
```
model_h2o = H2ORandomForestEstimator(ntrees=10,
                                     max_depth=5,
                                     min_rows=10,
                                     calibrate_model=True,
                                     calibration_frame=df_bank_test_h2o,
                                     binomial_double_trees=True)
```

In [45]:
```
df_bank_train_h2o['y'].asfactor()
```

Out[45]:

| y |
|---|
| no |
| no |
| no |
| no |

# H2O

- **Modelo y validación**

```
In [47]: model_h2o.train(x=df_bank_train_h2o.drop('y').col_names,
                         y='y',
                         training_frame=df_bank_train_h2o,
                         validation_frame=df_bank_test_h2o)

         drf Model Build progress: |████████████████████████████████████████████████| (done) 100%

Out[47]:
         Model Details
         =============
         H2ORandomForestEstimator : Distributed Random Forest
         Model Key: DRF_model_python_1675340086299_1
```

# H2O

- **Modelo y validación**

```
In [47]: model_h2o.train(x=df_bank_train_h2o.drop('y').col_names,
                         y='y',
                         training_frame=df_bank_train_h2o,
                         validation_frame=df_bank_test_h2o)

         drf Model Build progress: |██████████████████████████████████████████████████| (done) 100%

Out[47]:
         Model Details
         =============
         H2ORandomForestEstimator : Distributed Random Forest
         Model Key: DRF_model_python_1675340086299_1
```

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 |
|---|---|---|---|---|---|---|---|---|
| 10.0 | 20.0 | 8613.0 | 5.0 | 5.0 | 5.0 | 26.0 | 32.0 | 29.65 |

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

MSE: 0.07143322908577195
RMSE: 0.2672699554491151
LogLoss: 0.24150543139577066
Mean Per-Class Error: 0.19078149158071428
AUC: 0.8939062495083637
AUCPR: 0.5586624659180423
Gini: 0.7878124990167275
```

# H2O

- **Matriz de confusion y métricas**

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.19247724905076902

| | no | yes | Error | Rate |
|---|---|---|---|---|
| no | 28354.0 | 3277.0 | 0.1036 | (3277.0/31631.0) |
| yes | 1173.0 | 3047.0 | 0.278 | (1173.0/4220.0) |
| Total | 29527.0 | 6324.0 | 0.1241 | (4450.0/35851.0) |

Maximum Metrics: Maximum metrics at their respective thresholds

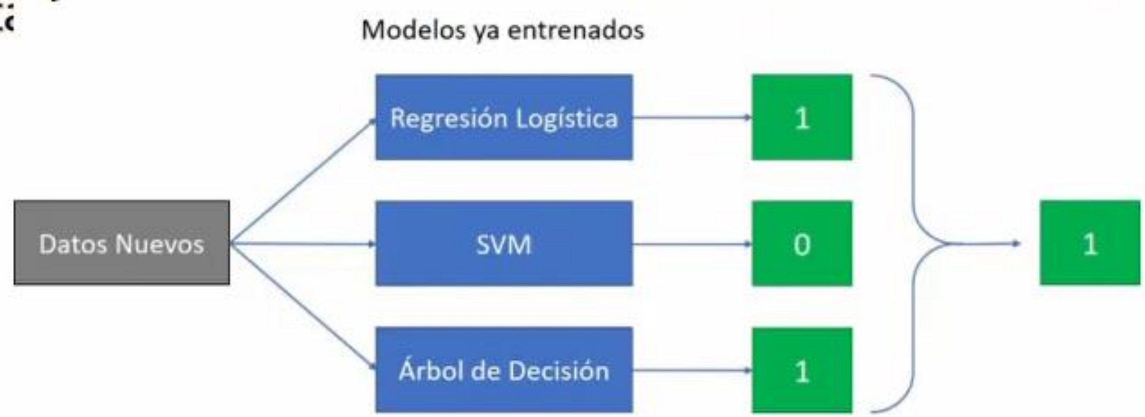
| metric | threshold | value | idx |
|---|---|---|---|
| max f1 | 0.1924772 | 0.5779590 | 231.0 |
| max f2 | 0.1330235 | 0.6755505 | 264.0 |
| max f0point5 | 0.4030910 | 0.5611235 | 137.0 |
| max accuracy | 0.4138724 | 0.9004491 | 132.0 |
| max precision | 1.0 | 1.0 | 0.0 |
| max recall | 0.0064442 | 1.0 | 397.0 |
| max specificity | 1.0 | 1.0 | 0.0 |
| max absolute_mcc | 0.1840600 | 0.5230680 | 235.0 |
| max min_per_class_accuracy | 0.1113503 | 0.8248445 | 278.0 |
| max mean_per_class_accuracy | 0.1068303 | 0.8252925 | 281.0 |
| max tns | 1.0 | 31631.0 | 0.0 |

# Ensemble Voting

## Ensemble models

Se tratan de métodos combinados que utilizan múltiples algoritmos de machine learning para obtener un mejor rendimiento predictivo.

- Voting: permite entrenar varios modelos con los mismos datos. Cada modelo tiene asociado un voto. La predicción final se obtiene como la más vot



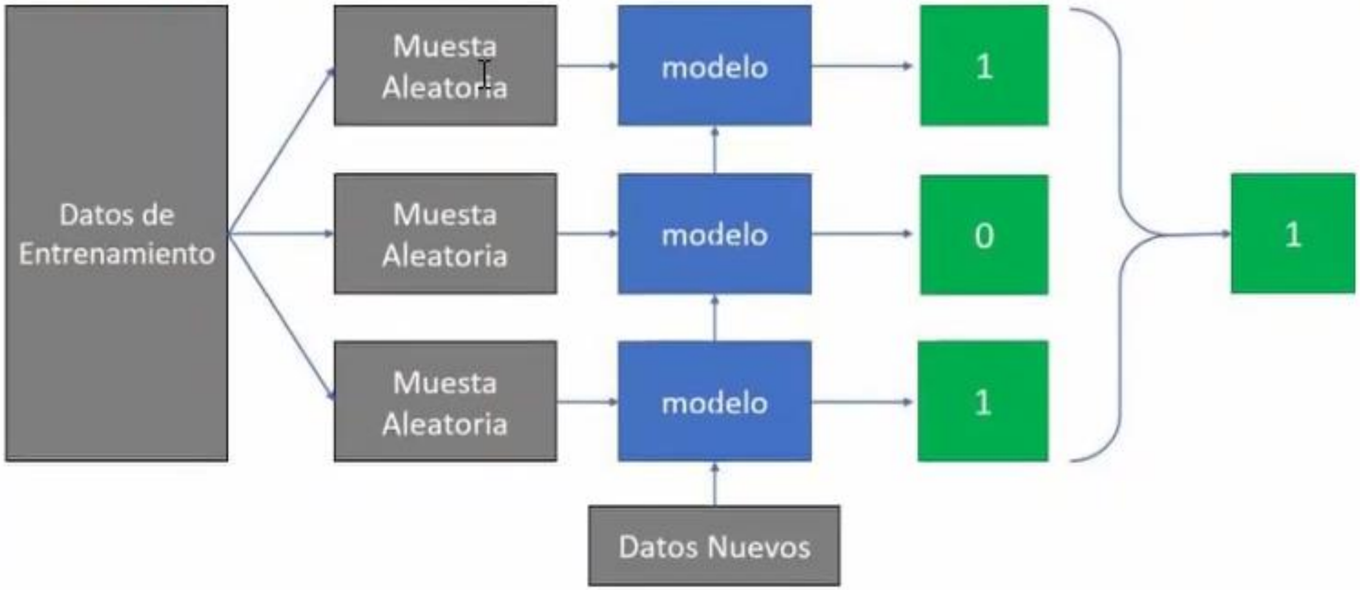

- Stacking: se trata de apilar modelos, es decir, usar la salida de un modelo como entrada de otro.

# Ensemble Models: Bagging

## Ensemble models

- Bagging: es un meta-algoritmo que consigue combinaciones de modelos a través de una familia inicial, provocando un menos overfitting y varianza. Consigue que los errores se compensen entre sí, entrenando cada modelo con subconjuntos del dataset original. El resultado es una combinación.

Ejemplo: Random Forest (bagging de decission tree)

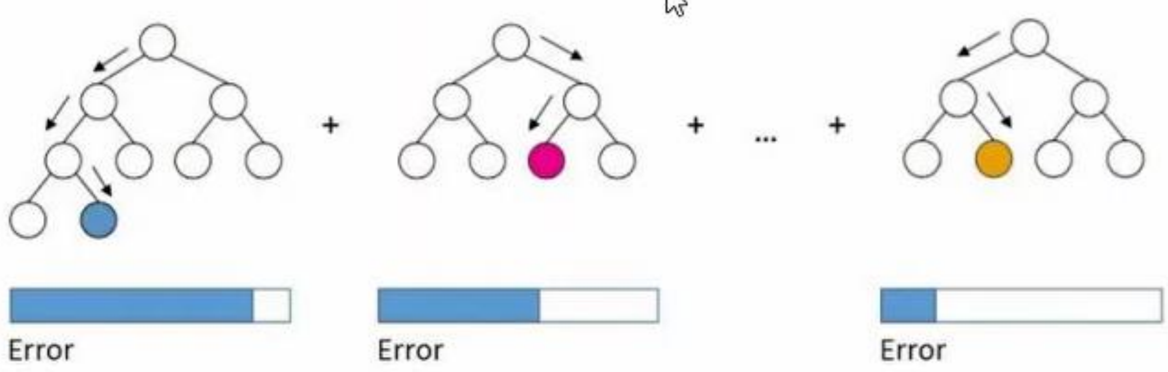


# Ensemble Models: Boosting

## Ensemble methods

- Boosting: en este caso, cada modelo intenta arreglar los errores de los modelos anteriores. Tras la primera clasificación, se dará más peso a las muestras mal clasificadas. En el caso de regresión se da más peso al error cuadrático medio para el siguiente modelo.

- Ejemplos: Xgboost, CatBoost, Lightgbm, AdaBoost




Intentar aplicar un modelo boosting a uno de los modelos realizados anteriormente (ejemplo Clasificación)

# Ensemble Models: XGBoost

```
In [36]: model2 = XGBClassifier().fit(X_train, y_train)
         y_pred = model2.predict(X_test)

C:\Python38\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and
will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when cons
tructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[16:08:52] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.

In [37]: saca_metricas(y_test, y_pred)

matriz de confusión
[[7692  293]
 [ 544  514]]
accuracy
0.9074422205020458
precision
0.6369268897149938
recall
0.48582230623818523
f1
```

# Ensemble Models: XGBoost

- **Ventajas:**
  - **Mejor precisión**
- **Inconvenientes:**
  - **No explicables**
  - **Mayor complejidad computacional**