

# Amine Ait Laamim

## Importation des bibliothèques

```
In [57]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
import joblib
```

## Chargement des données

```
In [58]: df = pd.read_csv("synthetic_heart_disease_dataset.csv")

Out[58]:
```

## Exploration des données

```
In [59]: df.shape

Out[59]: (58000, 21)
```

```
In [60]: df.describe()

Out[60]:
```

	Age	Weight	Height	BMI	Hypertension	Diabetes	Hyperlipidemia	Family_History	Previous_Heart_Attack	Systolic_BP	Diastolic_BP	Heart_Rate	Blood_Sugar_Fasting	Cholesterol_Total	Heart_Disease
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	54.46406	84.547520	174.460000	26.984284	0.299620	0.199260	0.251660	0.400500	0.099280	139.299580	89.528000	84.449560	124.493200	224.556360	0.463460
std	14.43709	20.213257	14.40379	6.367494	0.458096	0.399448	0.433971	0.490005	0.299041	23.083544	17.259063	14.491325	31.691507	43.157467	0.498668
min	30.0000	50.000000	150.000000	18.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.000000	60.000000	60.000000	70.000000	150.000000	0.000000
25%	42.00000	67.000000	162.000000	23.500000	0.000000	0.000000	0.000000	0.000000	0.000000	119.000000	75.000000	72.000000	97.000000	187.000000	0.000000
50%	54.00000	85.000000	174.000000	29.000000	0.000000	0.000000	0.000000	0.000000	0.000000	139.000000	90.000000	85.000000	125.000000	225.000000	0.000000
75%	67.00000	102.000000	187.000000	34.500000	1.000000	0.000000	1.000000	1.000000	0.000000	159.000000	104.000000	97.000000	152.000000	262.000000	1.000000
max	79.00000	119.000000	199.000000	40.000000	1.000000	1.000000	1.000000	1.000000	1.000000	179.000000	119.000000	109.000000	179.000000	299.000000	1.000000

```
In [61]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58000 entries, 0 to 49999
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   Age                   58000 non-null   int64   
 1   Gender                58000 non-null   object  
 2   Weight                58000 non-null   int64   
 3   Height                58000 non-null   int64   
 4   BMI                   58000 non-null   float64  
 5   Smoking               58000 non-null   object  
 6   Alcohol_Intake         29891 non-null   object  
 7   Physical_Activity       58000 non-null   object  
 8   Diet                  58000 non-null   object  
 9   Stress_Level           58000 non-null   object  
10  Hypertension           58000 non-null   int64   
11  Diabetes                58000 non-null   int64   
12  Hyperlipidemia          58000 non-null   int64   
13  Family_History          58000 non-null   int64   
14  Previous_Heart_Attack   58000 non-null   int64   
15  Systolic_BP             58000 non-null   int64   
16  Diastolic_BP            58000 non-null   int64   
17  Heart_Rate              58000 non-null   int64   
18  Blood_Sugar_Fasting     58000 non-null   int64   
19  Cholesterol_Total        58000 non-null   int64   
20  Heart_Disease           58000 non-null   int64   
dtypes: float64(1), int64(14), object(6)
memory usage: 8.0+ MB

In [62]: df.head()

Out[62]:
```

	Age	Gender	Weight	Height	BMI	Smoking	Alcohol_Intake	Physical_Activity	Diet	Stress_Level	...	Diabetes	Hyperlipidemia	Family_History	Previous_Heart_Attack	Systolic_BP	Diastolic_BP	Heart_Rate	Blood_Sugar_Fasting	Cholesterol_Total	Heart_Disease
0	48	Male	78	157	26.4	Never	NaN	Sedentary	Healthy	Medium	...	0	1	1	0	104	99	71	165	200	0
1	35	Female	73	163	33.0	Never	Low	Active	Average	High	...	0	1	1	0	111	72	60	145	206	0
2	79	Female	88	152	32.3	Never	NaN	Moderate	Average	Medium	...	0	0	1	0	116	102	78	148	208	0
3	75	Male	106	171	37.4	Never	Moderate	Moderate	Average	Low	...	0	1	0	0	171	92	109	105	230	0
4	34	Female	65	191	18.5	Current	NaN	Sedentary	Healthy	Low	...	1	0	0	0	164	67	108	116	220	0

5 rows × 21 columns

```
In [63]: df.columns

Out[63]: Index(['Age', 'Gender', 'Weight', 'Height', 'BMI', 'Smoking', 'Alcohol_Intake', 'Physical_Activity', 'Diet', 'Stress_Level', 'Hypertension', 'Diabetes', 'Hyperlipidemia', 'Family_History', 'Previous_Heart_Attack', 'Systolic_BP', 'Diastolic_BP', 'Heart_Rate', 'Blood_Sugar_Fasting', 'Cholesterol_Total', 'Heart_Disease'],
          dtype='object')

In [64]: df['Heart_Disease'].value_counts()

Out[64]:
Heart_Disease
0    26827
1    23173
Name: count, dtype: int64

In [65]: df.isna().sum()

Out[65]:
Age                0
Gender              0
Weight             0
Height             0
BMI                0
Smoking            0
Alcohol_Intake     21009
Physical_Activity   0
Diet               0
Stress_Level       0
Hypertension       0
Diabetes            0
Hyperlipidemia     0
Family_History     0
Previous_Heart_Attack 0
Systolic_BP        0
Diastolic_BP       0
Heart_Rate         0
Blood_Sugar_Fasting 0
Cholesterol_Total  0
Heart_Disease      0
dtype: int64
```

```
In [66]: df[df['Alcohol_Intake'].isna()]

Out[66]:
```

	Age	Gender	Weight	Height	BMI	Smoking	Alcohol_Intake	Physical_Activity	Diet	Stress_Level	...	Diabetes	Hyperlipidemia	Family_History	Previous_Heart_Attack	Systolic_BP	Diastolic_BP	Heart_Rate	Blood_Sugar_Fasting	Cholesterol_Total	Heart_Disease
0	48	Male	78	157	26.4	Never	NaN	Sedentary	Healthy	Medium	...	0	1	1	0	104	99	71	165	200	0
2	79	Female	88	152	32.3	Never	NaN	Moderate	Average	Medium	...	0	0	1	0	116	102	78	148	208	0
4	34	Female	65	191	18.5	Current	NaN	Sedentary	Healthy	Low	...	1	0	0	0	164	67	108	116	220	0
5	50	Male	116	186	25.3	Current	NaN	Sedentary	Average	Medium	...	0	1	0	0	171	91	106	97	225	0
7	51	Male	75	176	18.2	Former	NaN	Active	Average	Medium	...	0	0	1	0	117	63	89	143	154	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49985	54	Male	113	190	19.4	Current	NaN	Moderate	Average	Low	...	1	0	1	0	113	90	108	122	225	0
49986	46	Female	54	167	36.2	Never	NaN	Moderate	Average	Medium	...	0	1	0	0	108	108	70	83	291	0
49989	37	Male	117	178	30.4	Never	NaN	Moderate	Healthy	Low	...	0	1	0	0	138	89	102	99	230	0
49994	62	Male	91	197	36.8	Never	NaN	Active	Unhealthy	Low	...	0	0	1	0	117	80	106	97	270	0
49995	74	Male	104	155	29.9	Current	NaN	Active	Average	Medium	...	0	0	0	0	127	80	83	174	248	0

20109 rows × 21 columns

## Nettoyage des données

```
In [67]: df = df.drop(columns="Alcohol_Intake")

In [68]: df.isna().sum()

Out[68]:
Age                0
Gender              0
Weight             0
Height             0
BMI                0
Smoking            0
Physical_Activity   0
Diet               0
Stress_Level       0
Hypertension       0
Diabetes            0
Hyperlipidemia     0
Family_History     0
Previous_Heart_Attack 0
Systolic_BP        0
Diastolic_BP       0
Heart_Rate         0
Blood_Sugar_Fasting 0
Cholesterol_Total  0
Heart_Disease      0
dtype: int64

In [69]: cate_cols = df.select_dtypes(include=['object']).columns
cate_cols

Out[69]: Index(['Gender', 'Smoking', 'Physical_Activity', 'Diet', 'Stress_Level'], dtype='object')

In [70]: for col in cate_cols:
print(f'{col} : {df[col].unique()}')

Gender : ('Male', 'Female')
Smoking : ('Never', 'Current', 'Former')
Physical_Activity : ('Sedentary', 'Active', 'Moderate')
Diet : ('Healthy', 'Moderate', 'Unhealthy')
Stress_Level : ('Medium', 'High', 'Low')

In [71]: for col in cate_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    mapping = dict(zip(le.classes_, range(len(le.classes_))))
    print(f'{col} : {mapping}')

df.info()

Gender : ('Female': 0, 'Male': 1)
Smoking : ('Current': 0, 'Former': 1, 'Never': 2)
Physical_Activity : ('Active': 0, 'Moderate': 1, 'Sedentary': 2)
Diet : ('Average': 0, 'Healthy': 1, 'Unhealthy': 2)
Stress_Level : ('High': 0, 'Low': 1, 'Medium': 2)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58000 entries, 0 to 49999
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   Age                   58000 non-null   int64   
 1   Gender                58000 non-null   int64   
 2   Weight                58000 non-null   int64   
 3   Height                58000 non-null   int64   
 4   BMI                   58000 non-null   float64  
 5   Smoking               58000 non-null   int64   
 6   Physical_Activity       58000 non-null   int64   
 7   Diet                  58000 non-null   int64   
 8   Stress_Level           58000 non-null   int64   
 9   Hypertension           58000 non-null   int64   
10  Diabetes                58000 non-null   int64   
11  Hyperlipidemia          58000 non-null   int64   
12  Family_History          58000 non-null   int64   
13  Previous_Heart_Attack   58000 non-null   int64   
14  Systolic_BP             58000 non-null   int64   
15  Diastolic_BP            58000 non-null   int64   
16  Heart_Rate              58000 non-null   int64   
17  Blood_Sugar_Fasting     58000 non-null   int64   
18  Cholesterol_Total        58000 non-null   int64   
19  Heart_Disease           58000 non-null   int64   
dtypes: float64(1), int64(19)
memory usage: 7.0+ MB
```

## Préparation des données pour l'entraînement

```
In [72]: x = df.drop(columns="Heart_Disease")
y = df["Heart_Disease"]

In [73]: y.value_counts()

Out[73]:
Heart_Disease
0    26827
1    23173
Name: count, dtype: int64

In [74]: def equilibrage(X, y):
    import random
    x = X.values
    y = y.values
    idx_zeroes = [i for i, label in enumerate(y) if label == 1]
    idx_ones = [i for i, label in enumerate(y) if label == 0]

    if len(idx_ones) > len(idx_zeroes):
        majority = idx_zeroes
        minority = idx_zeroes
    else:
        majority = idx_ones
        minority = idx_ones

    random.shuffle(majority)
    majority = majority[:len(minority)]

    indices = majority + minority
    random.shuffle(indices)

    X_final = [X[i] for i in indices]
    y_final = [y[i] for i in indices]

    X_final = np.array(X_final)
    y_final = np.array(y_final).reshape(-1, 1)

    return X_final, y_final

In [75]: X, y = equilibrage(X, y)

In [76]: (y == 1).sum() == (y == 0).sum()

Out[76]: np.True_

In [77]: X.shape

Out[77]: (46346, 19)
```

```
In [78]: x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Normalisation des données

```
In [79]: scaler = StandardScaler()
x_train = scaler.fit_transform(X_train)
x_test = scaler.transform(X_test)
```

## Entraînement du modèle

```
In [80]: estimators = [10, 50, 100]
max_depths = [5, 10, 20]
for
    print(f"=====pour n_estimators = {e} =====")
    for d in max_depths:
        print(f"=====pour max_depth = {d} =====")
        RFC = RandomForestClassifier(max_depth=d, n_estimators=e, min_samples_leaf=2000, max_features='sqrt', random_state=42)
        RFC.fit(X_train, y_train)
        y_pred = RFC.predict(X_test)

        print(f"Accuracy : ", accuracy_score(y_test, y_pred))
        print(f"Precision : ", precision_score(y_test, y_pred))
        print(f"Recall : ", recall_score(y_test, y_pred))
        print(f"F1 score : ", f1_score(y_test, y_pred))
        print(f"feature importances : {RFC.feature_importances_}")

=====pour n_estimators = 10 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.924487594398087
Precision : 0.869997186212116
Recall : 0.8444444444444444
F1-score : 0.926885314931481
feature importances : [1.67602944e-01 1.7414541e-05 6.48189163e-04 1.78491283e-04 1.81269518e-04 3.6832188e-04 1.17158188e-04 3.73280782e-05 5.68723788e-05 3.28721943e-01 1.71795899e-01 2.1243928e-04 0.00000000e+0 0.0754200e-02 4.2862621e-04 2.50786985e-04 4.86783135e-04 5.46822812e-04 2.54117943e-01]
=====pour n_estimators = 50 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.916181229734628
Precision : 0.895669485308315
Recall : 0.94465738097914
F1-score : 0.919566880690363
feature importances : [1.67602944e-01 1.7414541e-05 5.92721811e-04 1.78491283e-04 1.82454823e-04 3.67642864e-04 9.75845705e-05 3.73280782e-05 5.68723788e-05 3.28721943e-01 1.71795899e-01 2.1243928e-04 0.00000000e+0 0.0754200e-02 4.2862621e-04 2.50786985e-04 4.86783135e-04 5.46822812e-04 2.54117943e-01]
=====pour n_estimators = 20 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.916181229734628
Precision : 0.895669485308315
Recall : 0.94465738097914
F1-score : 0.919566880690363
feature importances : [1.67602944e-01 1.7414541e-05 5.92721811e-04 1.78491283e-04 1.82454823e-04 3.67642864e-04 9.75845705e-05 3.73280782e-05 5.68723788e-05 3.28721943e-01 1.71795899e-01 2.1243928e-04 0.00000000e+0 0.0754200e-02 4.2862621e-04 2.50786985e-04 4.86783135e-04 5.46822812e-04 2.54117943e-01]
=====pour n_estimators = 100 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.934196322545846
Precision : 1.0
Recall : 0.870157138356747
F1-score : 0.938571365328933
feature importances : [1.56315851e-01 3.94566956e-05 5.8285129e-04 3.48013218e-04 3.4013717e-04 4.45488523e-04 1.15099511e-05 2.2455124e-04 5.6888884e-05 2.18137473e-01 9.51638723e-02 2.9924453e-05 1.41808976e-05 4.42857571e-02 3.37693521e-04 1.9752403e-04 6.4648704e-04 4.4366065e-04 1.17519582e-01]
=====pour n_estimators = 200 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.95129813872798
Precision : 1.0
Recall : 0.967831284767986
F1-score : 0.951698282755088
feature importances : [1.12119797e-01 6.76586225e-05 1.68538284e-03 2.67848606e-03 2.74537398e-04 9.7984842e-05 8.87948464e-05 1.28716815e-03 0.04013345e-05 2.10579800e-01 0.00130555e-01 4.49812127e-05 1.53748827e-05 1.36339858e-02 1.34839926e-03 1.35726045e-03 3.73454176e-03 8.8986208e-04 3.32669959e-01]
=====pour n_estimators = 500 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.95129813872798
Precision : 1.0
Recall : 0.967831284767986
F1-score : 0.951698282755088
feature importances : [1.12119797e-01 6.76586225e-05 1.68538284e-03 2.67848606e-03 2.74537398e-04 9.7984842e-05 8.87948464e-05 1.28716815e-03 0.04013345e-05 2.10579800e-01 0.75267931e-02 1.49582168e-05 1.46628158e-05 1.25971069e-02 1.35231493e-03 1.35722907e-03 3.7543431e-03 1.0384272e-04 1.35438210e-01]
=====pour n_estimators = 1000 =====
c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  return self._fit(X, y, dtype=DTYPE, sample_weight=sample_weight)
Accuracy : 0.95129813872798
Precision : 1.0
Recall : 0.967831284767986
F1-score : 0.951698282755088
feature importances : [1.12119797e-01 6.76586225e-05 1.68538284e-03 2.67848606e-03 2.74537398e-04 9.7984842e-05 8.87948464e-05 1.28716815e-03 0.04013345e-05 2.10579800e-01 0.75267931e-02 1.49582168e-05 1.46628158e-05 1.25971069e-02 1.35231493e-03 1.35722907e-03 3.7543431e-03 1.0384272e-04 1.35438210e-01]
```

## Enregistrer les modèles entraînés

```
In [87]: joblib.dump(RFC, "rf_classification.pkl")
joblib.dump(scaler, "ScalerRF_classification.pkl")

Out[87]: ["ScalerRF_classification.pkl"]
```

## Bagging

```
bagging avec estimator = DecisionTreeClassifier()

n_estimators=50

In [82]: bagging = BaggingClassifier(estimator=DecisionTreeClassifier(min_samples_leaf=2000), n_estimators=50,
max_samples=0.8, bootstrap=True, random_state=77)

bagging.fit(X_train, y_train)
y_pred = bagging.predict(X_test)

print("Accuracy : ", accuracy_score(y_test, y_pred))
print("Precision : ", precision_score(y_test, y_pred))
print("Recall : ", recall_score(y_test, y_pred))
print("F1-score : ", f1_score(y_test, y_pred))

c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\ensemble\bagging.py:930: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  (y = column_or_1d(y, warn=True))
Accuracy : 0.827723840451995
Precision : 0.8664975587927774
Recall : 0.787782834984717
F1-score : 0.8225358373152573
n_estimators=10

In [83]: bagging = BaggingClassifier(estimator=DecisionTreeClassifier(min_samples_leaf=2000), n_estimators=5,
max_samples=0.8, bootstrap=True, random_state=54)

bagging.fit(X_train, y_train)
y_pred = bagging.predict(X_test)

print("Accuracy : ", accuracy_score(y_test, y_pred))
print("Precision : ", precision_score(y_test, y_pred))
print("Recall : ", recall_score(y_test, y_pred))
print("F1-score : ", f1_score(y_test, y_pred))

c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\ensemble\bagging.py:930: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  (y = column_or_1d(y, warn=True))
Accuracy : 0.827723840451995
Precision : 0.8664975587927774
Recall : 0.787782834984717
F1-score : 0.8225358373152573
```

```
In [84]: from sklearn.linear_model import LogisticRegression
bagging = BaggingClassifier(estimator=LogisticRegression(),
n_estimators=50, max_samples=0.8, bootstrap=True, random_state=42)

bagging.fit(X_train, y_train)
y_pred = bagging.predict(X_test)

print("Accuracy : ", accuracy_score(y_test, y_pred))
print("Precision : ", precision_score(y_test, y_pred))
print("Recall : ", recall_score(y_test, y_pred))
print("F1-score : ", f1_score(y_test, y_pred))

c:\Users\Lapto\anaconda3\envs\global_env\Lib\site-packages\sklearn\ensemble\bagging.py:930: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
  (y = column_or_1d(y, warn=True))
Accuracy : 0.92597982826969
Precision : 0.92593831846498
Recall : 0.9263516389953171
F1-score : 0
```