```
import pandas                   as pd
from   sklearn.preprocessing   import StandardScaler
from   sklearn.model_selection import train_test_split
from   sklearn.svm             import SVC
from   sklearn.metrics         import accuracy_score
```

```
df = pd.read_csv("parkinson_disease.csv")
df.shape
```

```
(195, 24)
```

```
# Voir toutes les colonnes
pd.set_option('display.max_columns', 25)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   name              195 non-null    object 
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
```

```
df.isnull().sum()
```

```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

Il n'y a pas de valeur manquante.
Que des nombres sauf name.
Name n'a pas d'intérêt pour ce projet.
Pas de valeur catégorielle.

```
df.drop(columns='name', inplace=True)
```

```
# Diviser en feature et label
y = df['status']
X = df.drop(columns='status')
```

```
# Quelles sont les valeur des colonnes de fetures
X['NHR'].min(), X['NHR'].max()
X['MDVP:Flo(Hz)'].min(), X['MDVP:Flo(Hz)'].max()
```

```
(65.476, 239.17)
```

Aucune variables de features n'est normalisées.
Il est impossible d'alimenter un algo de ML.

Normaliser avec une échelle strandard est une option.

```
scaler = StandardScaler()
X = scaler.fit_transform(X)
X
```

```
array([[-0.82929965, -0.43616456, -0.95203729, ...,  0.48047686,
    -0.21053082,  0.86888575],
   [-0.77097169, -0.53097409, -0.05772056, ...,  1.31118546,
     0.27507712,  1.80360503],
   [-0.90947638, -0.7231683 , -0.10987483, ...,  1.01768236,
    -0.10362861,  1.40266141],
   ...,
   [ 0.49557839,  0.47010361, -0.96839309, ..., -0.81807931,
     0.78033848, -0.83241014],
   [ 1.07876114,  2.19004398, -0.95417967, ..., -0.22906571,
    -0.63700298, -0.92610456],
   [ 1.45481664,  0.69224632, -0.88348115, ..., -0.43085284,
     0.45480231, -0.64505466]])
```

Après la normalisation de df devient un tableau NumPy.
Le type array NumPy n'est pas un souci pour alimenter le modèle de ML.

Je vais tout de même le convertir en dataframe.

```
X_columns = df.drop(columns='status').columns
pd.DataFrame(X, columns=X_columns)
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDV
0	-0.829300	-0.436165	-0.952037	0.334914	0.749759	0.132963	0.760800	0.131755	0.74
1	-0.770972	-0.530974	-0.057721	0.715418	1.037674	0.453892	1.276809	0.452684	1.68
2	-0.909476	-0.723168	-0.109875	0.884991	1.325589	0.720770	1.585687	0.721813	1.20
3	-0.909622	-0.649092	-0.114229	0.775389	1.325589	0.578885	1.284076	0.577677	1.34
4	-0.925657	-0.606245	-0.130608	1.368893	1.901418	1.095750	2.047187	1.096793	1.83
...	...	...	...	...	...	...	...	...	...
190	0.483467	0.371185	-0.508265	-0.337173	-0.401899	-0.228505	-0.311189	-0.227459	0.59
191	1.339202	0.612690	-0.618218	-0.120037	-0.401899	0.001213	-0.191272	0.002258	-0.11
192	0.495578	0.470104	-0.968393	1.526058	1.037674	0.991026	0.797139	0.992069	-0.3
193	1.078761	2.190044	-0.954180	0.243924	-0.113985	0.132963	0.164847	0.131755	-0.3
194	1.454817	0.692246	-0.883481	-0.113833	-0.401899	-0.120403	-0.100425	-0.120483	-0.5

195 rows × 22 columns

```
# diviser les features et label en ensembles d'entraînement et en ensembles de test
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, strati-
fy=y, random_state=34)
x_train.shape, x_test.shape
```
```
((156, 22), (39, 22))
```

```
y_train.shape, y_test.shape
```
```
((156,), (39,))
```

```
#
svc = SVC()
svc.fit(x_train, y_train)
```

```
svc_prediction = svc.predict(x_test)

svc_score = accuracy_score(y_test, svc_prediction)
svc_score
```
```
0.8974358974358975
```