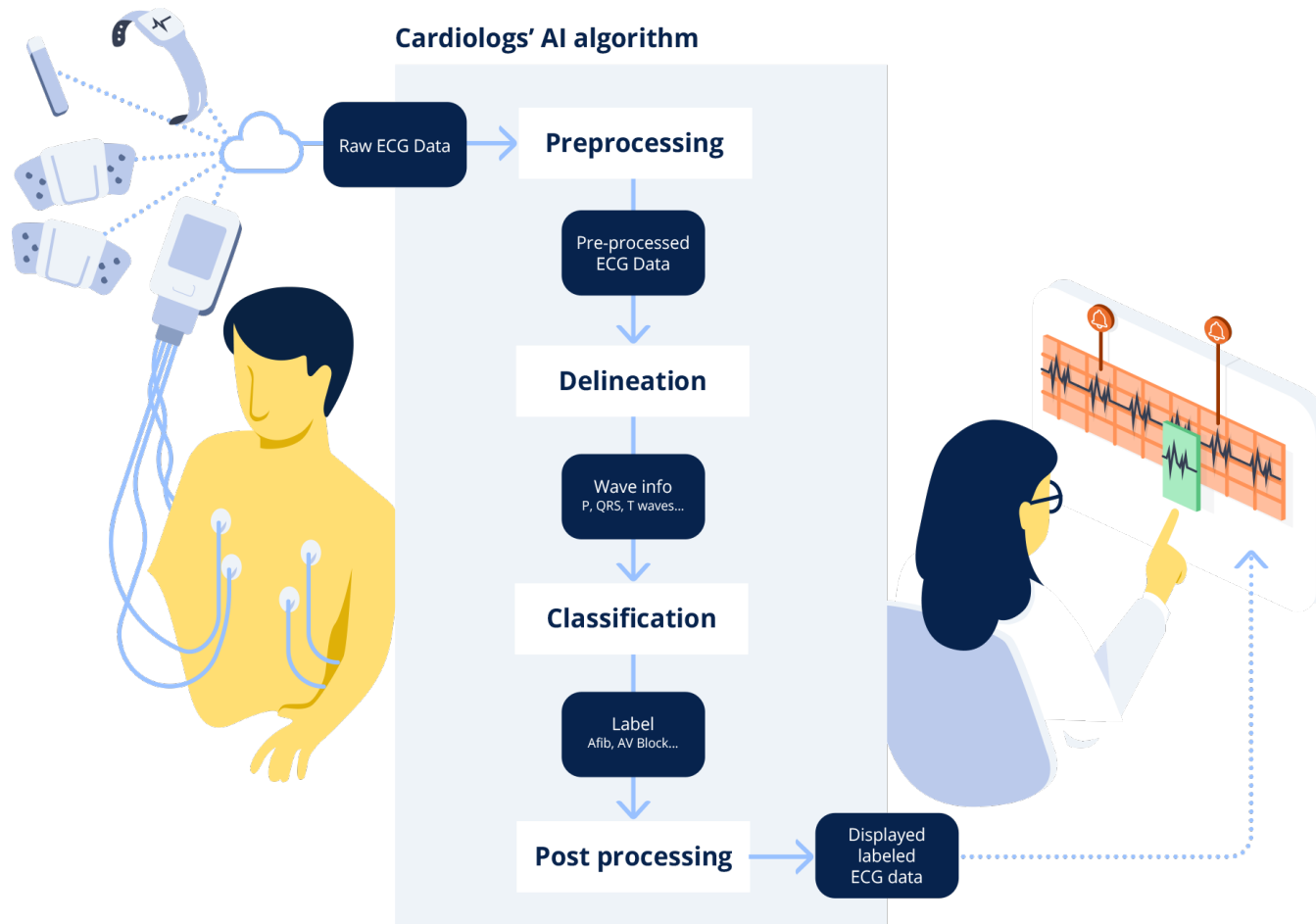


Machine learning : Algorithme de comparaison et de diagnostic d'ECG

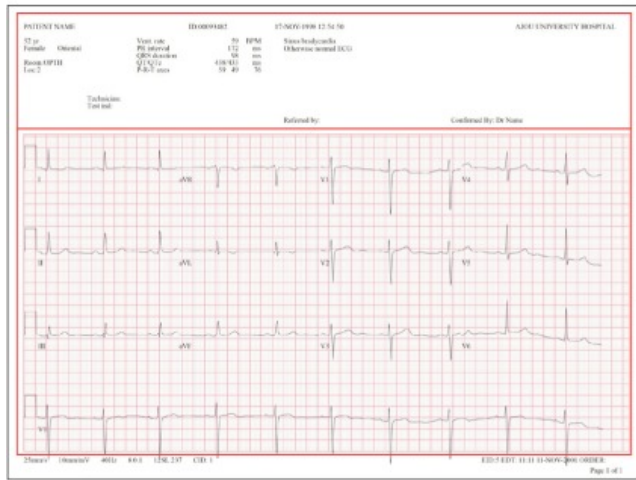


<https://cardiologs.com/blog/ai-systems-powered-by-deep-learning/>

- 1- Établir une base de données d'ECG:
 - a – Définition
 - b – Justification
 - c – Recherche de la base et justification du choix
 - d – Présentation de la base de données
- 2- Comparer les différentes méthodes de placements d'électrodes:
 - a – Définition
 - b – Présentation des méthodes
 - c – Lien avec la base de données
- 3- Développer un algorithme de machine learning utilisant la base de données et les résultats comparateurs de méthodes:
 - a – Définition
 - b – Régression logistique
 - c – Arbre de décision
 - d – Machines à vecteurs de support
 - e – Programme 1
 - f – Programme 2
 - g – Conclusion

Annexe

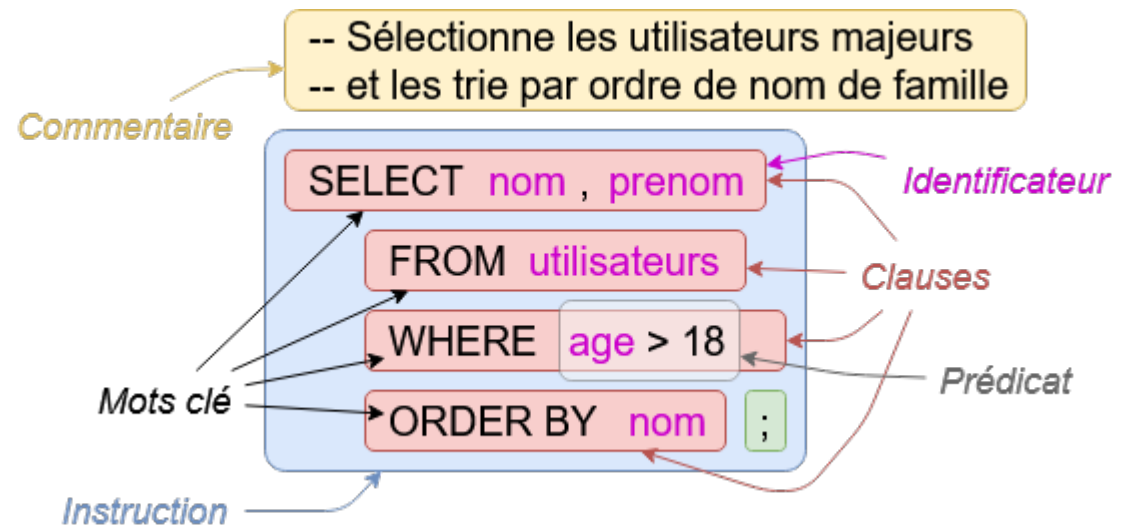
1- Établir une base de données d'ECG



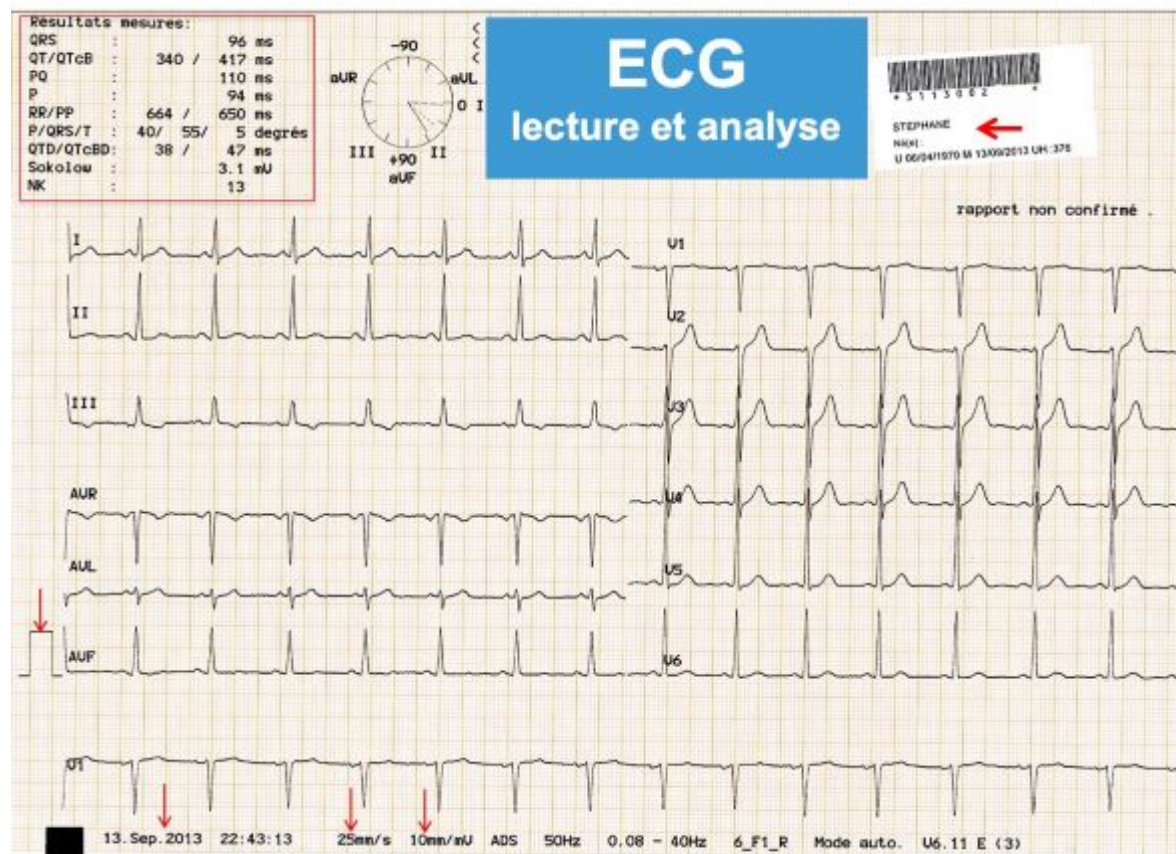
- Alphanumeric value
 - Demographic information
 - Patient ID
 - Date
 - Ethnicity
 - ECG parameter values
 - Ventricular rate
 - PR interval
 - QRS duration
 - QT/QTc
 - P-R-T axes
 - Interpretation
-
- Waveform data
 - 12 lead waveform data
 - I, II, III
 - aVR, aVL, aVF
 - V1, V2, V3, V4, V5, V6

Exemple de catégorisation d'un ECG et exemple langage SQL

<https://e-hir.org/journal/Figure.php?xn=hir-24-242.xml&id=>



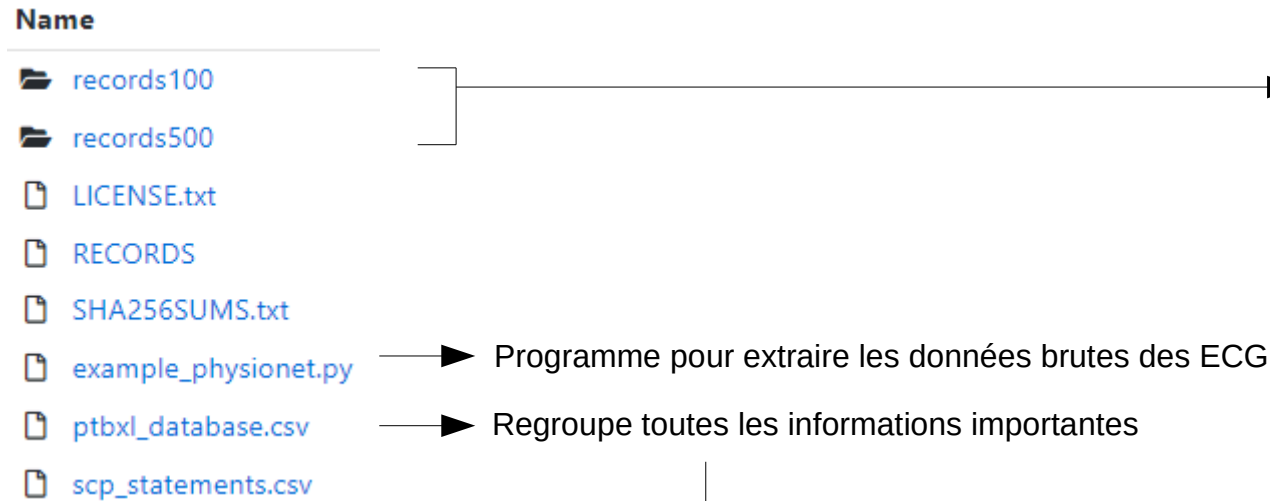
<https://info.blaisepascal.fr/wp-content/uploads/2020/10/Structure-dune-instruction-SQL.png>



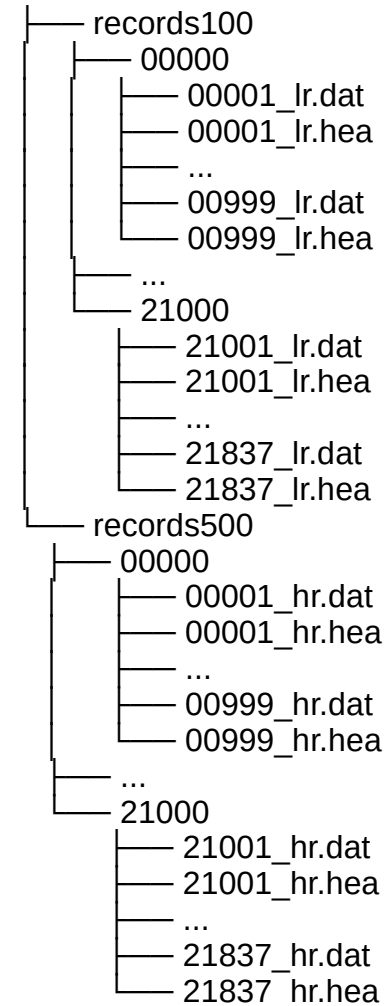
<https://www.e-cardiogram.com/ecg-lecture-et-analyse/>

	patient_id	age	sex	height	weight	nurse	site	device	recording_date	report	...	validated_by_human
0	15709.0	56.0	1	NaN	63.0	2.0	0.0	CS-12 E	1984-11-09 09:17:34	sinusrhythmus periphere niederspannung	...	True
1	13243.0	19.0	0	NaN	70.0	2.0	0.0	CS-12 E	1984-11-14 12:55:37	sinusbradykardie sonst normales ekg	...	True
2	20372.0	37.0	1	NaN	69.0	2.0	0.0	CS-12 E	1984-11-15 12:49:10	sinusrhythmus normales ekg	...	True
3	17014.0	24.0	0	NaN	82.0	2.0	0.0	CS-12 E	1984-11-15 13:44:57	sinusrhythmus normales ekg	...	True

Base de données PTB-XL



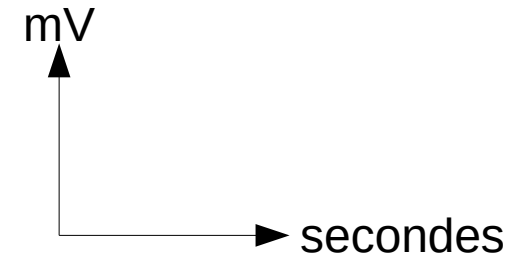
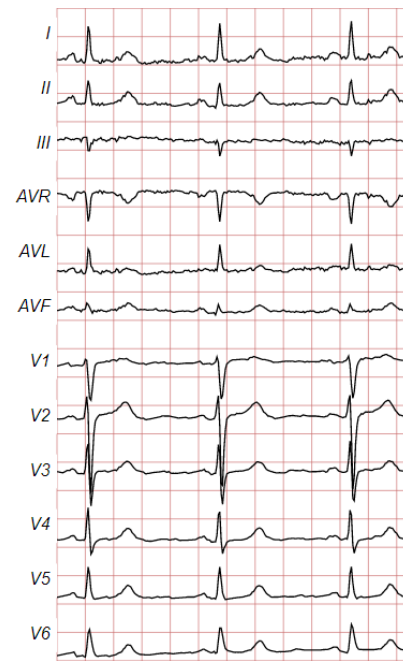
	patient_id	age	sex	height	weight	nurse	site	device	recording_date	report	...	validated_by_human
0	15709.0	56.0	1	NaN	63.0	2.0	0.0	CS-12 E	1984-11-09 09:17:34	sinusrhythmus periphere niederspannung	...	True
1	13243.0	19.0	0	NaN	70.0	2.0	0.0	CS-12 E	1984-11-14 12:55:37	sinusbradykardie sonst normales ekg	...	True
2	20372.0	37.0	1	NaN	69.0	2.0	0.0	CS-12 E	1984-11-15 12:49:10	sinusrhythmus normales ekg	...	True
3	17014.0	24.0	0	NaN	82.0	2.0	0.0	CS-12 E	1984-11-15 13:44:57	sinusrhythmus normales ekg	...	True



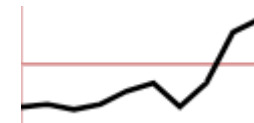
Données d'ECG regroupées par paquet de 1000 pour une fréquence d'échantillonnage De 500 ou 100Hz.

Exemple du premier ECG

	0	1	2	3	4	...
0	-0.119	-0.116	-0.120	-0.117	-0.103	
1	-0.055	-0.051	-0.044	-0.038	-0.031	
2	0.064	0.065	0.076	0.080	0.072	
3	0.086	0.083	0.082	0.077	0.066	
4	-0.091	-0.090	-0.098	-0.098	-0.087	
5	0.004	0.006	0.016	0.021	0.021	
6	-0.069	-0.064	-0.058	-0.050	-0.045	
7	-0.031	-0.036	-0.034	-0.030	-0.027	
8	0.000	-0.003	-0.010	-0.015	-0.020	
9	-0.026	-0.031	-0.028	-0.023	-0.019	
10	-0.039	-0.034	-0.029	-0.022	-0.018	
11	-0.079	-0.074	-0.069	-0.064	-0.058	



	0	1	2	3	4	5	6	7	8	9
0	-0.119	-0.116	-0.120	-0.117	-0.103	-0.097	-0.119	-0.096	-0.048	-0.037



Citation:

Wagner P., Strodthoff N., Bousseljot R., Samek W., & Schaeffter T. (2020). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.1). PhysioNet. <https://doi.org/10.13026/x4td-x982>.

Wagner P., Strodthoff N., Bousseljot R.-D., Kreiseler D., Lunze F.I., Samek W., Schaeffter T. (2020), PTB-XL: A Large Publicly Available ECG Dataset. Scientific Data. <https://doi.org/10.1038/s41597-020-0495-6>

Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., ... & Stanley H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

2- Comparer les différentes méthodes de placements d'électrodes

Exemples d'électrodes



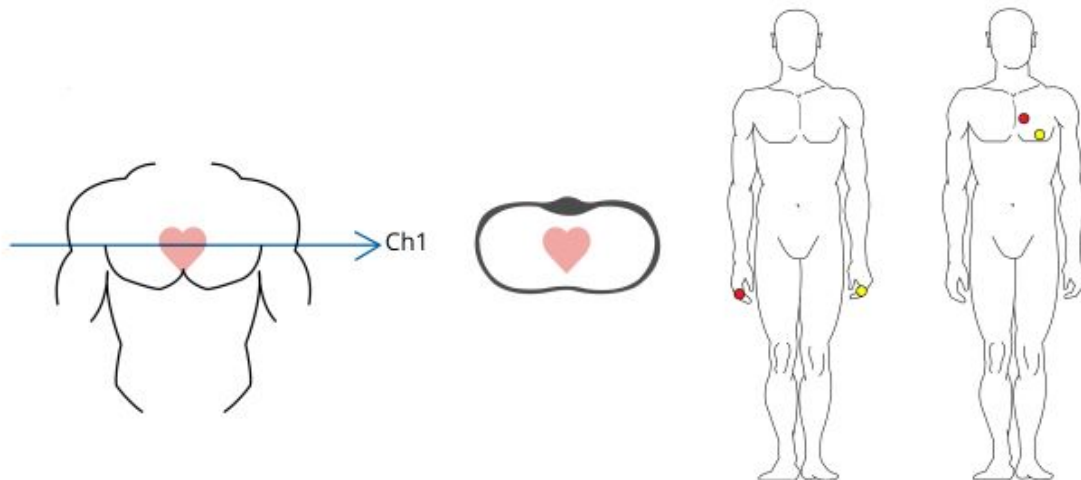
Électrode à usage unique, mousse, 1 cm
Electrode chimique Ag/AgCl recouverte sur le plastique

<https://www.medicalexpo.fr/prod/intco-medical/product-118592-936155.html>



Électrode adhésive à usage unique, mousse, 3 cm
Connexion par bouton pression

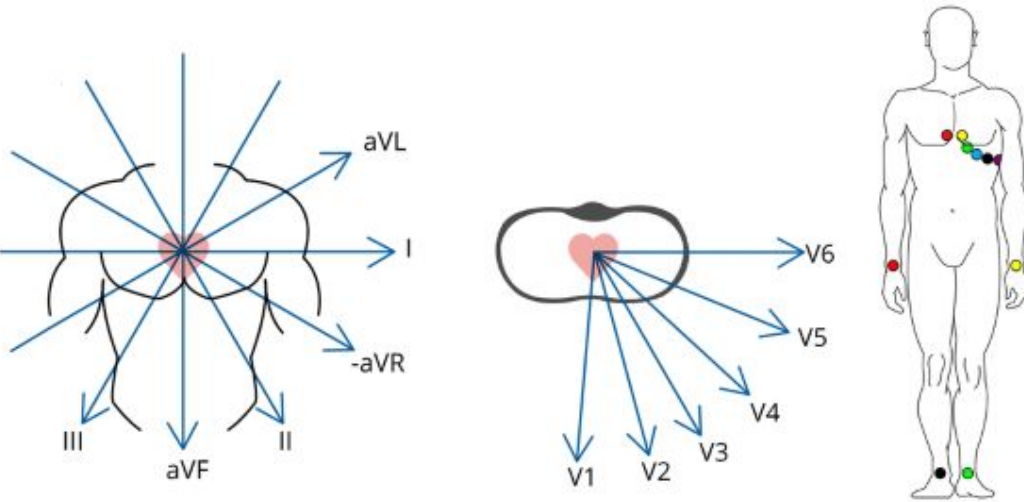
<https://www.cnsac-medshop.com/fr/produit/electrodes-ecg-a-usage-unique-mousse-3-cm-%E2%88%85-30-pieces/>



ECG à 1 dérivation

<https://www.cardiosecur.com/fr/magazine/articles-specialises/systemes-de-derivation-ecg>

ECG à 12 dérivations et signaux obtenus



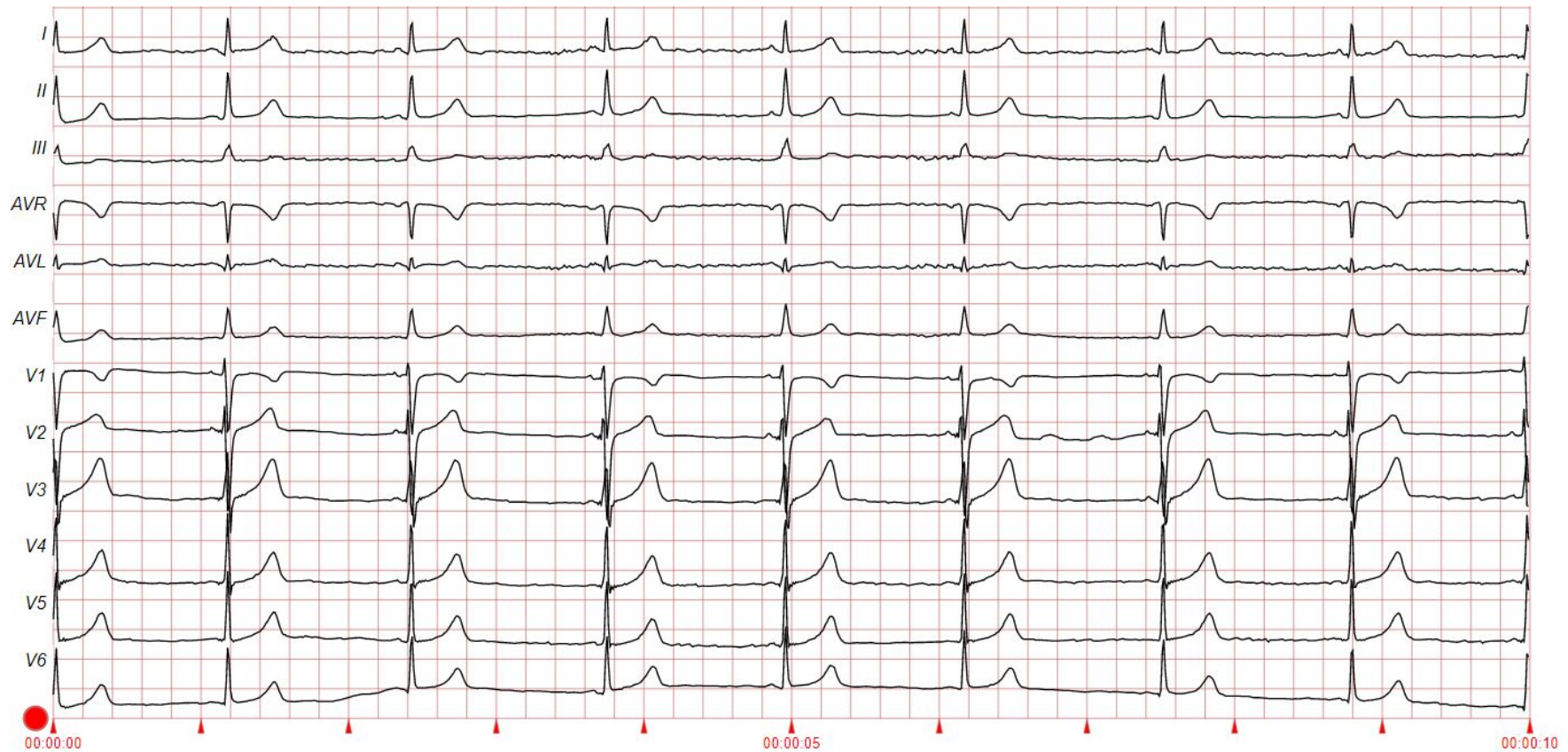
<https://www.cardiosecur.com/fr/magazine/articles-specialises/systemes-de-derivation-ecg>

Citation:

Wagner P., Strodthoff N., Bousseljot R., Samek W., & Schaeffter T. (2020). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.1). PhysioNet. <https://doi.org/10.13026/x4td-x982>.

Wagner P., Strodthoff N., Bousseljot R.-D., Kreiseler D., Lunze F.I. Samek W., Schaeffter T. (2020). PTB-XL: A Large Publicly Available ECG Dataset. Scientific Data. <https://doi.org/10.1038/s41597-020-0495-6>

Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., ... & Stanley H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.



Tension
(0,25 mV
par carré)

Temps (0,2 secondes par carré)

Visualisation de signaux via physionet



Citation:

Wagner P., Strodthoff N., Bousseljot R., Samek W., & Schaeffter T. (2020). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.1). PhysioNet. <https://doi.org/10.13026/x4td-x982>.

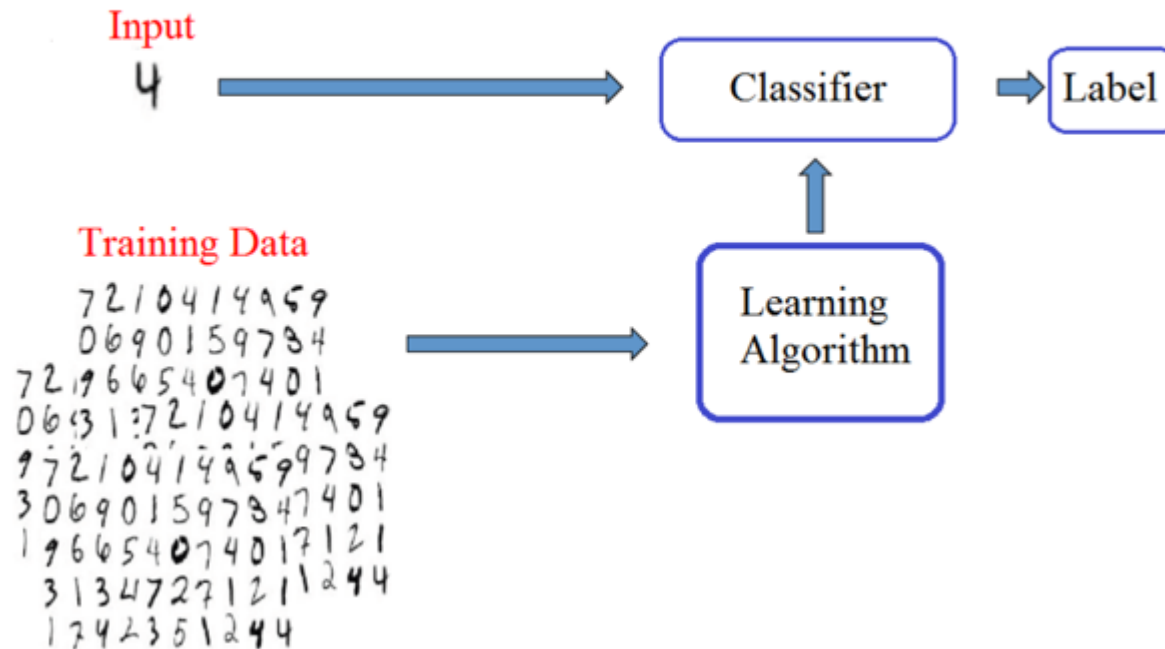
Wagner P., Strodthoff N., Bousseljot R.-D., Kreiseler D., Lunze F.I. Samek W., Schaeffter T. (2020), PTB-XL: A Large Publicly Available ECG Dataset. Scientific Data. <https://doi.org/10.1038/s41597-020-0495-6>

Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., ... & Stanley H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

3- Développer un algorithme de machine learning utilisant la base de données et les résultats comparateurs de méthodes

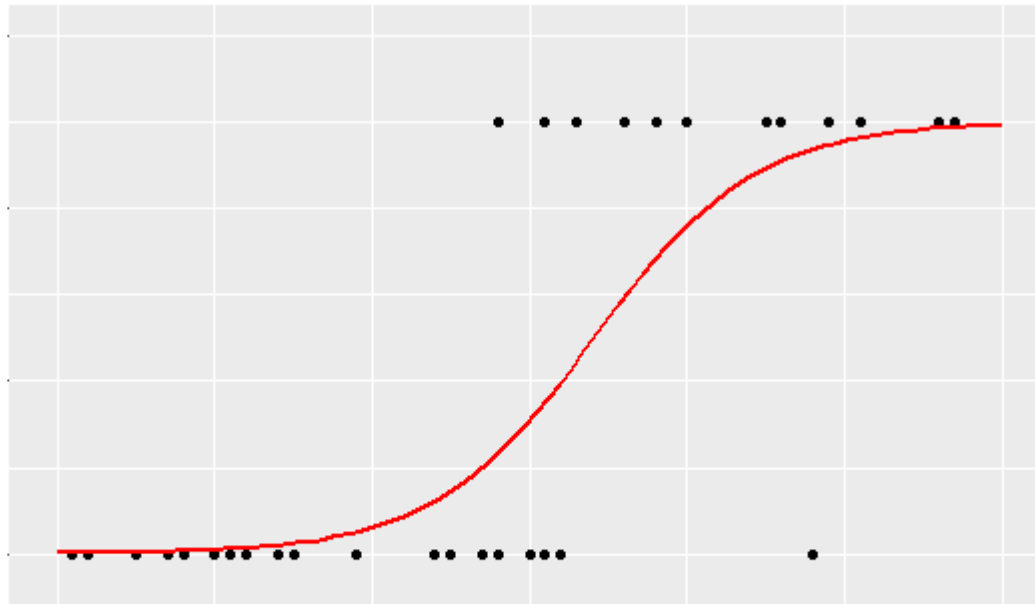
Fonctionnement de l'apprentissage supervisé

Supervisé



<https://www.itrain.fr/post/apprentissage-supervis%C3%A9-et-non-supervis%C3%A9>

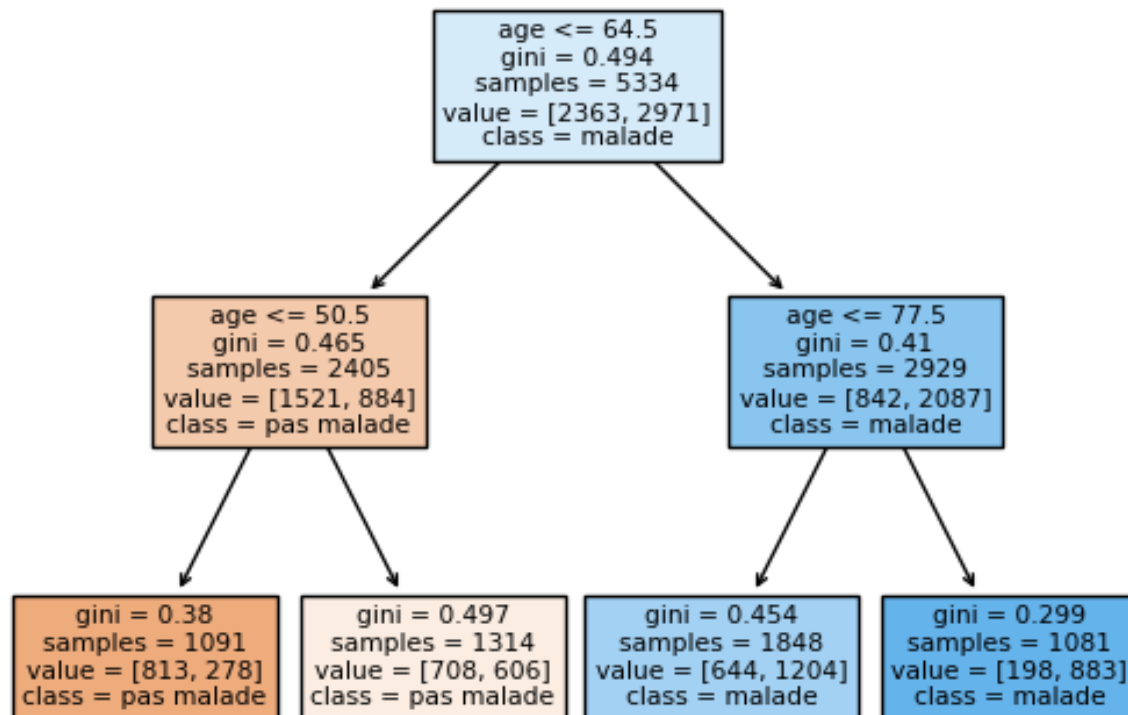
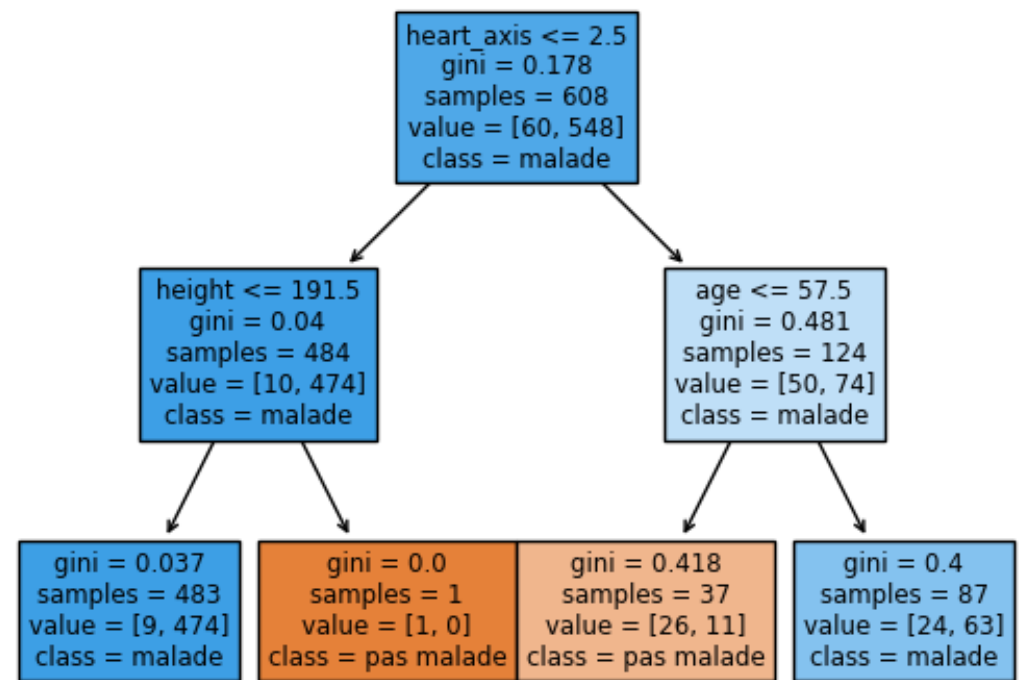
La fonction de régression logistique



<https://delladata.fr/regression-logistique/>

$$P(X) = \frac{\exp(\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n)}{1 + \exp(\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n)} = \frac{\exp(\sum \beta X)}{1 + \exp(\sum \beta X)}$$

Les 2 arbres de décisions des programmes

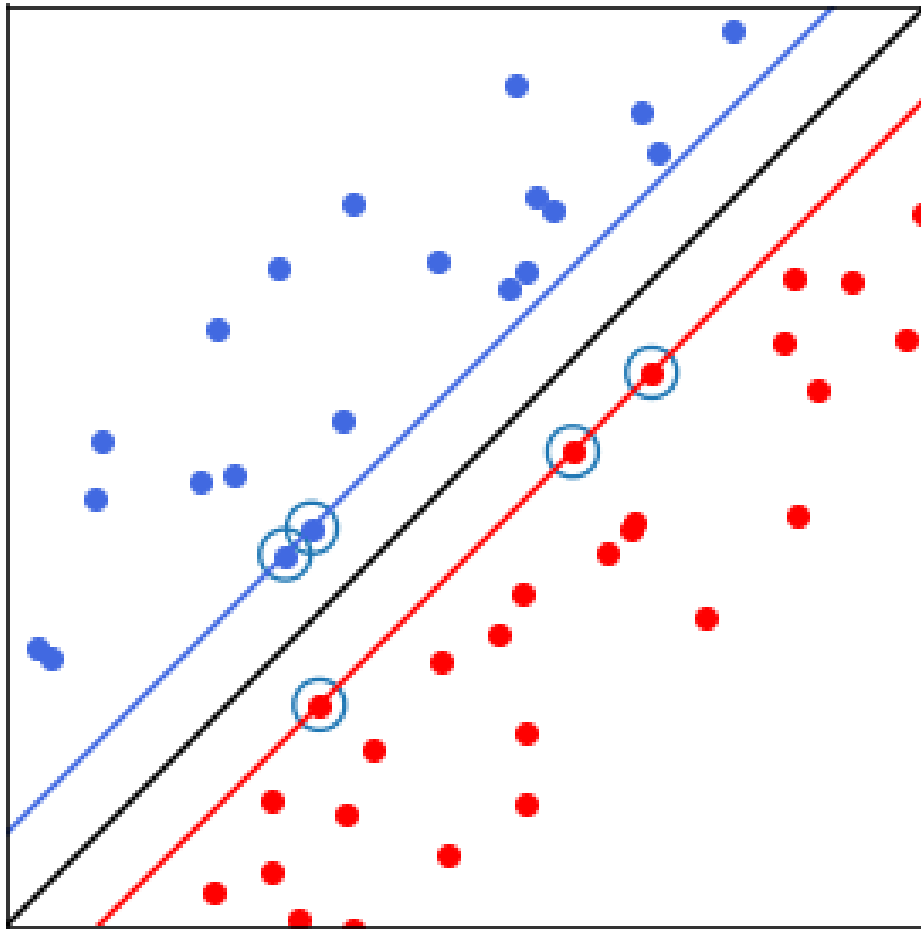







Le score Gini :

$$Gini = \sum (pk \times (1 - pk))$$

<https://kongakura.fr/article/arbre-de-decision-python>

Représentation de la classification SVM



-   : Vecteurs de support
-  : Hyperplan
-   : Limite des marges maximales

<https://dataanalyticspost.com/Lexique/svm/>

1er programme

Y, le tableau initial

	patient_id	age	sex	height	weight	nurse	site	device	recording_date	report	...	baseline_drift	static_noise	burst_noise
0	15709.0	56.0	1	NaN	63.0	2.0	0.0	CS-12 E	1984-11-09 09:17:34	sinusrhythmus periphere niederspannung	...	NaN	, I-V1,	NaN
1	13243.0	19.0	0	NaN	70.0	2.0	0.0	CS-12 E	1984-11-14 12:55:37	sinusbradykardie sonst normales ekg	...	NaN	NaN	NaN
2	20372.0	37.0	1	NaN	69.0	2.0	0.0	CS-12 E	1984-11-15 12:49:10	sinusrhythmus normales ekg	...	NaN	NaN	NaN
3	17014.0	24.0	0	NaN	82.0	2.0	0.0	CS-12 E	1984-11-15 13:44:57	sinusrhythmus normales ekg	...	, II,III,AVF	NaN	NaN
electrodes_problems			extra_beats		pacemaker		strat_fold		filename_lr		filename_hr			
NaN			NaN		NaN		3		records100/00000/00001_lr		records500/00000/00001_hr			
NaN			NaN		NaN		2		records100/00000/00002_lr		records500/00000/00002_hr			
NaN			NaN		NaN		5		records100/00000/00003_lr		records500/00000/00003_hr			
NaN			NaN		NaN		3		records100/00000/00004_lr		records500/00000/00004_hr			

Colonne des diagnostics
Rajoutées à Y

0	[NORM]
1	[NORM]
2	[NORM]
3	[NORM]
4	[NORM]
...	
21832	[STTC]
21833	[NORM]
21834	[STTC]
21835	[NORM]
21836	[NORM]
Name: diagnostic_superclass	

- On enlève les colonnes avec la fonction `.drop()` :

```
Y = Y.drop(['patient_id', 'nurse', 'recording_date', 'report', 'scp_codes', 'heart_axis',
'infarction_stadium1', 'infarction_stadium2', 'validated_by', 'baseline_drift',
'static_noise', 'burst_noise', 'electrodes_problems', 'extra_beats',
'pacemaker', 'filename_lr', 'filename_hr'], axis=1)
```

- Changement des éléments de la colonne diagnostic en entiers

0	[NORM]	→	0	0
1	[NORM]		1	0
2	[NORM]		2	0
3	[NORM]		3	0
4	[NORM]		4	0
...				
21832	[STTC]		21832	1
21833	[NORM]		21833	0
21834	[STTC]		21834	1
21835	[NORM]		21835	0
21836	[NORM]		21836	0
Name: diagnostic_superclass			Name: diagnostic_superclass	

- Changement des éléments de la colonne device en entiers

```
L=['CS-12 E', 'AT-6 C 5.0', 'AT-6 C', 'CS-12', 'AT-6 C 5.5', 'AT-6 C 5.8',
'AT-6 C 5.6', 'AT-6 6', 'AT-6 C 5.3', 'AT-60 3', 'CS100 3']
```

0	CS-12 E	→	0	0
1	CS-12 E		1	0
2	CS-12 E		2	0
3	CS-12 E		3	0
4	CS-12 E		4	0
...				
21832	AT-60 3		21832	9
21833	AT-60 3		21833	9
21834	AT-60 3		21834	9
21835	AT-60 3		21835	9
21836	AT-60 3		21836	9
Name: device			Name: device	

- Changement des éléments de la colonne second_opinion en entiers

0	False
1	False
2	False
3	False
4	False
...	
21832	False
21833	False
21834	True
21835	False
21836	False
Name: second_opinion	



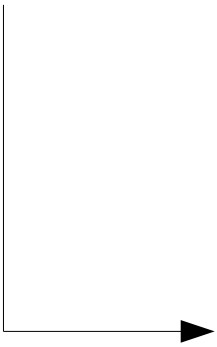
0	0
1	0
2	0
3	0
4	0
...	
21832	0
21833	0
21834	1
21835	0
21836	0
Name: second_opinion	

La fonction .dropna() a supprimée toutes les lignes contenant une case vide

	age	sex	height	weight	site	device	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold	diagnostic_superclass
100	48.0	1	172.0	72.0	0.0	0	0	0	1	4	0
115	49.0	1	174.0	74.0	0.0	0	0	0	1	10	1
135	52.0	0	167.0	72.0	3.0	0	0	0	1	5	0
138	58.0	0	181.0	69.0	3.0	0	0	0	1	1	1
139	27.0	1	164.0	54.0	3.0	0	0	0	1	9	1
...
20942	52.0	0	174.0	80.0	2.0	3	0	0	1	10	0
20969	58.0	0	168.0	74.0	2.0	9	0	1	1	6	1
21039	58.0	0	168.0	74.0	2.0	9	0	1	1	6	1
21105	47.0	1	160.0	50.0	2.0	9	0	1	1	1	0
21343	57.0	1	168.0	97.0	2.0	3	0	0	1	1	0

Tableau de 6668 lignes X 11 colonnes

Pour obtenir un index plus clair, on crée un nouveau tableau avec nos valeurs



age	
0	48
1	49
2	52
3	58
4	27
...	...
6663	52
6664	58
6665	58
6666	47
6667	57

Set de diagnostic test,
ce que le modèle est censé prédire à partir du set de variables x_test

4648 0
2388 1
770 0
467 0
4338 1

3822 1
2460 1
3818 1
1250 1
705 0

Name: diagnostic_superclass, Length: 1667

	age	sex	height	weight	site	device	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold
4648	53	1	177	76	1	7	0	0	1	8
2388	72	0	183	78	28	4	0	0	1	10
770	58	1	149	60	1	4	0	0	1	2
467	38	1	150	50	1	4	0	0	1	6
4338	70	0	178	70	1	4	0	0	1	7
...

1 - Chargement des différents modèles prédictifs

```
LR = LogisticRegression(solver="liblinear")
modele = LR.fit(x_entrainement,y_entrainement)
```

```
DT = DecisionTreeClassifier(max_depth=2)
modele = DT.fit(x_entrainement,y_entrainement)
```

```
sv = svm.SVC()
modele = sv.fit(x_entrainement,y_entrainement)
```

2 – Prédiction des diagnostics par rapport au set x_test

► y_pred = modele.predict(x_test)

3 – Taux de précision obtenus

► Régression logistique : 68,4%

► Arbre de décision : 60,8%

► Machine à vecteurs de Support : 68,1%

2ème programme

Nombre de cases vides parmi chaque colonnes : [0, 89, 0, 14854, 12408, 1509, 18, 0, 0, 0, 0, 8505, 16211, 21734, 9411, 0, 0, 0, 20230, 18575, 21224, 21807, 19883, 21544, 0, 0, 0]

Les nouvelles colonnes non enlevés

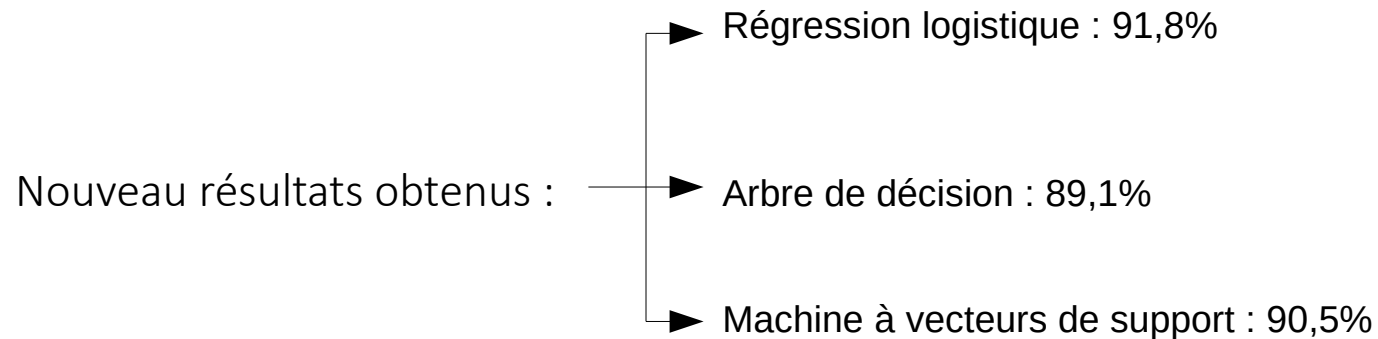
0	NaN	0	2.0	0	NaN	0	NaN
1	NaN	1	2.0	1	NaN	1	NaN
2	NaN	2	2.0	2	NaN	2	NaN
3	NaN	3	2.0	3	NaN	3	NaN
4	NaN	4	2.0	4	NaN	4	NaN
...							
21832	1.0	21832	1.0	21832	LAD	21832	0
21833	1.0	21833	1.0	21833	MID	21833	4
21834	1.0	21834	1.0	21834	MID	21834	4
21835	1.0	21835	1.0	21835	LAD	21835	0
21836	1.0	21836	1.0	21836	MID	21836	4
Name: validated_by		Name: nurse		Name: heart_axis		Name: heart_axis	

Modification des éléments en entiers

	age	sex	height	weight	nurse	site	device	heart_axis	validated_by	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold
0	48	1	155	45	11	1	4	1	7	0	0	1	7
1	81	1	160	75	1	1	4	0	0	0	0	1	9
2	87	1	166	70	8	1	5	0	0	0	0	1	8
3	67	1	152	60	11	1	4	0	0	0	0	1	10
4	70	0	174	63	10	1	4	0	0	0	0	1	2
...
756	52	0	174	80	1	2	3	4	1	0	0	1	10
757	58	0	168	74	1	2	9	0	1	0	1	1	6
758	58	0	168	74	1	2	9	0	1	0	1	1	6
759	47	1	160	50	1	2	9	4	1	0	1	1	1
760	57	1	168	97	1	2	3	4	1	0	0	1	1

Tableau de 761 lignes X 14 colonnes

Nouveau tableau obtenu après les modifications du 2ème programme et on a retiré la colonne des diagnostics



Annexe :

Citation de la base de donnée PTB-XL:

Wagner P., Strodthoff N., Bousseljot R., Samek W., & Schaeffter T. (2020). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.1). PhysioNet. <https://doi.org/10.13026/x4td-x982>.

Wagner P., Strodthoff N., Bousseljot R.-D., Kreiseler D., Lunze F.I., Samek W., Schaeffter T. (2020), PTB-XL: A Large Publicly Available ECG Dataset. Scientific Data. <https://doi.org/10.1038/s41597-020-0495-6>

Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., ... & Stanley H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

Programme 1

```
## Importation des modules
import pandas as pd
import seaborn as sns
import ast

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn import metrics, svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

## acquisition du fichier csv et incorporation d'une colonne de diagnostics
# Chemin à écrire devant le nom des fichiers dépendant d'où ils se situent
chemin="D:\\\\"

# On charge la base de données dans Y
Y = pd.read_csv(chemin+'ptbtl_database.csv')
Y=Y.drop(['ecg_id'],axis=1)
Y.scp_codes = Y.scp_codes.apply(lambda x: ast.literal_eval(x))

# On charge le fichier scp_statements pour en garder les diagnostics
tab_diag = pd.read_csv(chemin+'scp_statements.csv', index_col=0)
tab_diag = tab_diag[tab_diag.diagnostic == 1]

def aggregate_diagnostic(y_dic):
    tmp = []
    for key in y_dic.keys():
        if key in tab_diag.index:
            tmp.append(tab_diag.loc[key].diagnostic_class)
    return list(set(tmp))

# On ajoute la colonne des diagnostics
Y['diagnostic_superclass'] = Y.scp_codes.apply(aggregate_diagnostic)

## On manipule les colonnes et lignes pour obtenir un DataFrame correct (
## =comporte uniquement des entiers/flottants et aucune 'case' vide

# On retire les colonnes inutiles ou trop incomplètes
Y = Y.drop(['patient_id', 'nurse', 'recording_date', 'report', 'scp_codes',
            'heart_axis', 'infarction_stadium1', 'infarction_stadium2', 'validated_by',
            'baseline_drift', 'static_noise', 'burst_noise', 'electrodes_problems',
            'extra_beats', 'pacemaker', 'filename_lr', 'filename_hr'], axis=1)
```

```
# On remplace tout ce qui n'est pas norm en 0 et tout ce qui est norm en 1
for i in range(len(Y['diagnostic_superclass'])):
    if Y.at[i, 'diagnostic_superclass'] == ['NORM']:
        Y.at[i, 'diagnostic_superclass'] = 0 # ecg normal
    else:
        Y.at[i, 'diagnostic_superclass'] = 1 # ecg malade

# On remplace les False par 0 et les True par 1
L=['second_opinion', 'initial_autogenerated_report', 'validated_by_human']
for e in L:
    for i in range(len(Y[e])):
        if Y.at[i, e] == False:
            Y.at[i, e] = 0
        else:
            Y.at[i, e] = 1

# On remplace les appareils utilisés par des nombres entre 0 et 10
L=['CS-12 E', 'AT-6 C 5.0', 'AT-6 C', 'CS-12', 'AT-6 C 5.5', 'AT-6 C 5.8',
  'AT-6 C 5.6', 'AT-6 6', 'AT-6 C 5.3', 'AT-60 3', 'CS100 3']

for i in range(len(Y['device'])):
    for j in range(len(L)):
        if Y.at[i, 'device'] == L[j]:
            Y.at[i, 'device'] = j

# On retire les lignes qui comportent une case vide puis on envoie le
# DataFrame dans un nouveau fichier csv pour faire correspondre les numéros
# de ligne
Y=Y.dropna()
Y.to_csv(chemin+'ptbtl_database_2.csv')
Y=pd.read_csv(chemin+'ptbtl_database_2.csv')
Y=Y.drop(['Unnamed: 0'], axis=1)

# On rend les valeurs entières (on ne travaille pas avec des flottants)
Y=Y.astype('int')

## Machine learning
# on scinde Y en 2 matrices
x = Y.drop('diagnostic_superclass', axis=1)
y = Y['diagnostic_superclass']

x_entrainement, x_test, y_entrainement, y_test = train_test_split(
    x, y, test_size=0.2)
```

```

DT = DecisionTreeClassifier(max_depth=2)
LR = LogisticRegression(solver="liblinear")
sv = svm.SVC()
modele1 = sv.fit(x_entrainement,y_entrainement)
modele2 = LR.fit(x_entrainement,y_entrainement)
modele3 = DT.fit(x_entrainement,y_entrainement)

y_pred = modele1.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))
y_pred = modele2.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))
y_pred = modele3.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))

# On crée un arbre de décision
plt.close()
plot_tree(DT, feature_names=['age', 'sex', 'height', 'weight', 'nurse', 'site', 'device',
    'heart_axis', 'validated_by', 'second_opinion',
    'initial_autogenerated_report', 'validated_by_human', 'strat_fold'], class_names=['pas malade','malade'],filled=True)
plt.show()

## Moyenne
L=[modele1,modele2,modele3]
for e in L:
    k=0
    for i in range(100):
        X_entrainement, X_test, Y_entrainement, Y_test =train_test_split(x,y,test_size= 0.25)
        Y_pred = e.predict(X_test)
        k=k+metrics.accuracy_score(Y_test,Y_pred)
    print((100*k)/100)

```

Programme 2

```
## Importation des modules
import pandas as pd
import seaborn as sns
import ast

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn import metrics, svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

## acquisition du fichier csv et incorporation d'une colonne de diagnostics
# Chemin à écrire devant le nom des fichiers dépendant d'où ils se situent
chemin="D:\\\\"

# On charge la base de données dans Y
Y = pd.read_csv(chemin+'ptbxi_database.csv')
Y=Y.drop(['ecg_id'],axis=1)
Y.scp_codes = Y.scp_codes.apply(lambda x: ast.literal_eval(x))

# On charge le fichier scp_statements pour en garder les diagnostics
tab_diag = pd.read_csv(chemin+'scp_statements.csv', index_col=0)
tab_diag = tab_diag[tab_diag.diagnostic == 1]

def aggregate_diagnostic(y_dic):
    tmp = []
    for key in y_dic.keys():
        if key in tab_diag.index:
            tmp.append(tab_diag.loc[key].diagnostic_class)
    return list(set(tmp))

# On ajoute la colonne des diagnostics
Y['diagnostic_superclass'] = Y.scp_codes.apply(aggregate_diagnostic)

## On manipule les colonnes et lignes pour obtenir un DataFrame correct
# (= comporte uniquement des entiers/flottants et aucune 'case' vide

# On retire les colonnes inutiles ou trop incomplètes
Y = Y.drop(['patient_id', 'recording_date', 'report', 'scp_codes',
'infarction_stadium1', 'infarction_stadium2', 'baseline_drift', 'static_noise',
'burst_noise', 'electrodes_problems', 'extra_beats', 'pacemaker', 'filename_lr',
'filename_hr'],axis=1)
```

```
# On remplace les False par 0 et les True par 1
L=['second_opinion','initial_autogenerated_report','validated_by_human']
for e in L:
    for i in range(len(Y['second_opinion'])):
        if Y.at[i,e] == False:
            Y.at[i,e]=0
        else:
            Y.at[i,e]=1

# on remplace les valeurs dans les listes par des entiers consécutifs pour
#pouvoir utiliser ces colonnes dans la prédiction

# éléments de la colonne device
L1=['CS-12  E', 'AT-6 C 5.0', 'AT-6 C', 'CS-12', 'AT-6 C 5.5', 'AT-6 C 5.8', 'AT-6 C

# éléments de la colonne heart_axis
L2=['LAD', 'ALAD', 'RAD', 'AXR', 'MID', 'ARAD', 'AXL', 'SAG']

for i in range(len(Y['device'])):
    for j in range(len(L1)):
        if Y.at[i,'device']==L1[j]:
            Y.at[i,'device']=j

for i in range(len(Y['heart_axis'])):
    for j in range(len(L2)):
        if Y.at[i,'heart_axis']==L2[j]:
            Y.at[i,'heart_axis']=j

# On remplace tout ce qui n'est pas norm en 0 et tout ce qui est norm en 1
for i in range(len(Y['diagnostic_superclass'])):
    if Y.at[i,'diagnostic_superclass']=='NORM':
        Y.at[i,'diagnostic_superclass']=0
    else:
        Y.at[i,'diagnostic_superclass']=1

# On retire les lignes qui comportent une case vide puis on envoie le
#DataFrame dans un nouveau fichier csv pour faire correspondre les
#numéros de ligne
Y=Y.dropna()
Y.to_csv(chemin+'ptbxi_database_2.csv')
Y=pd.read_csv(chemin+'ptbxi_database_2.csv')
Y=Y.drop(['Unnamed: 0'],axis=1)
```



```
# On rend les valeurs entières (on ne travaille pas avec des flottants)
Y=Y.astype('int')
```

```
## Machine learning
# on scinde Y en 2 matrices
x = Y.drop('diagnostic_superclass', axis=1)
y = Y['diagnostic_superclass']

x_entrainement, x_test, y_entrainement, y_test = train_test_split(x,y,test_size= 0.3)
DT = DecisionTreeClassifier()
LR = LogisticRegression(solver="liblinear")
sv = svm.SVC()
modele1 = sv.fit(x_entrainement,y_entrainement)
modele2 = LR.fit(x_entrainement,y_entrainement)
modele3 = DT.fit(x_entrainement,y_entrainement)
```

```
y_pred = modele1.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))
y_pred = modele2.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))
y_pred = modele3.predict(x_test)
print("Précision:",metrics.accuracy_score(y_test, y_pred))
```

```
# On crée un arbre de décision
plt.close()
plot_tree(DT, feature_names=['age', 'sex', 'height', 'weight', 'nurse', 'site', 'device',
    'heart_axis', 'validated_by', 'second_opinion',
    'initial_autogenerated_report', 'validated_by_human', 'strat_fold'], class_names=['pas malade', 'malade'],filled=True)
plt.show()
## Moyenne
i,j,k=0,0,0
for p in range(100):
    X_entrainement, X_test, Y_entrainement, Y_test =train_test_split(x,y,test_size= 0.25)
    modele1 = sv.fit(X_entrainement,Y_entrainement)
    modele2 = LR.fit(X_entrainement,Y_entrainement)
    modele3 = DT.fit(X_entrainement,Y_entrainement)
    Y_pred1 = modele1.predict(X_test)
    Y_pred2 = modele2.predict(X_test)
    Y_pred3 = modele3.predict(X_test)
    i=i+metrics.accuracy_score(Y_test,Y_pred1)
    j=j+metrics.accuracy_score(Y_test,Y_pred2)
    k=k+metrics.accuracy_score(Y_test,Y_pred3)
print((100*i)/100)
print((100*j)/100)
print((100*k)/100)
```