

# Prédiction des décès dus au COVID-19 grâce au Machine Learning

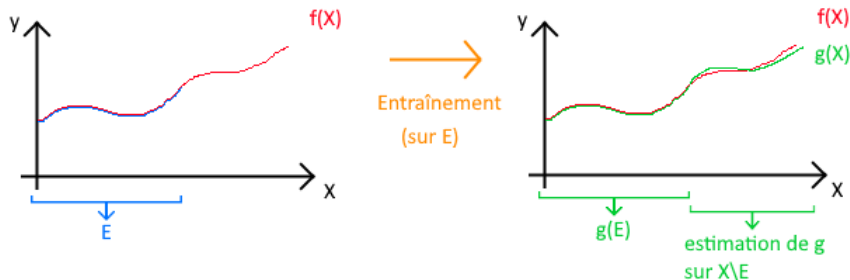
Enzo DE CARVALHO

Numéro d'inscription : 29448  
2020-2021

# Sommaire

- 1 Premières approches : simples regressions
- 2 Approches multivariées
- 3 Conclusion

# Principe de la démarche



↔ le modèle  $\hat{g}$  généralise les données connues  $E$  fournies

# Première approche : regression linéaire

En utilisant **ElasticNet**

Modèle :

$$\hat{g}_{deces}(\omega, t) = \omega_0 + \omega_1 t$$

$t$  le temps

$\omega = \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix}$  un paramètre à déterminer

# Première approche : regression linéaire

En utilisant **ElasticNet**

Modèle :

$$\hat{g}_{deces}(\omega, t) = \omega_0 + \omega_1 t$$

$t$  le temps

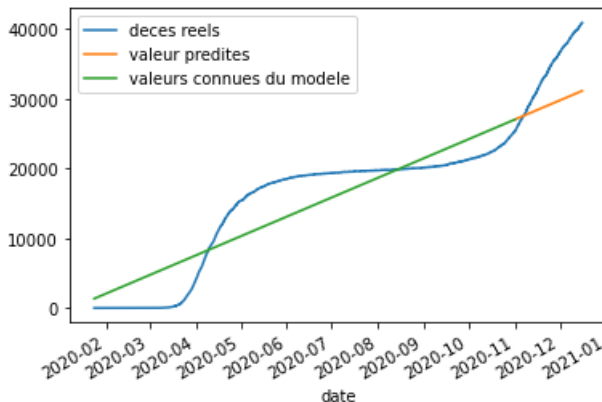
$$\omega = \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} \text{ un paramètre à déterminer}$$

Le modèle **ElasticNet** détermine alors  $\omega$

Le résultat dépend des hyperparamètres  $\alpha$  et  $\rho$

# Première application

Prédictions entre le 01/11/2020 et 16/12/2020



$$\rho = 0.9$$

$$\alpha = 0.1$$

Figure – Résultats peu satisfaisants...

# SVR

Modèle **SVR** ( **Régresseur à Support Vectoriel** )

Hyperparamètres :

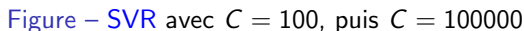
- $C$  le paramètre de régularisation

- $\epsilon$  la taille du tube de « non-pénalité »

- $\gamma$  paramètre du noyau (rbf ici)

---

$\gamma$  paramètre du noyau (rbf ici)





# Validation Croisée

Grille d'hyperparamètres :

```
params = {'svr__C' : [10, 100, ...],  
          'svr__eps' : [0.1, 0.01, ..], ... }
```

# Validation Croisée

Grille d'hyperparamètres :

```
params = {'svr__C' : [10, 100, ...],
          'svr__eps' : [0.1, 0.01, ..], ... }
```

Pour une combinaison d'hyperparamètres :

Données d'entraînement totale

Découpage 1	Découpage 2	Découpage 3	Découpage 4
----------------	----------------	----------------	----------------

Test 1			
	Test 2		
		Test 3	
			Test 4

→ Score 1

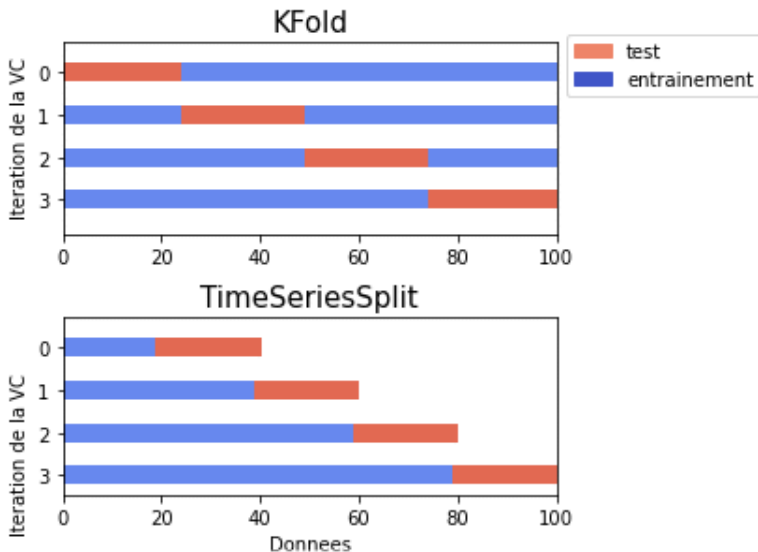
→ Score 2

→ Score 3

→ Score 4

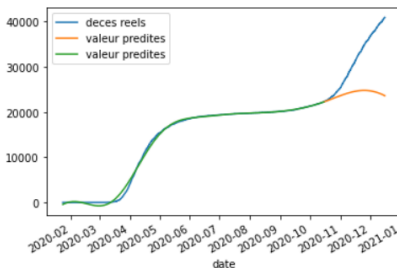
→ Score final

# Stratégie pour la Validation Croisée

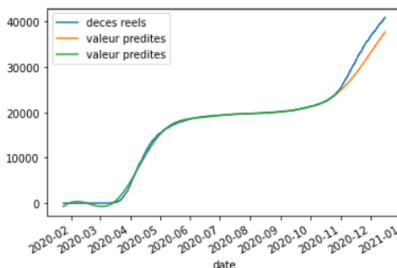


# Application avec SVR

```
params = {'svr__C' : [10**i for i in range(1,8)]}
```



{'svr\_\_C' : 100000}



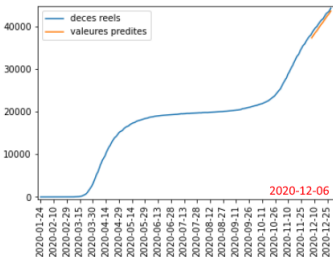
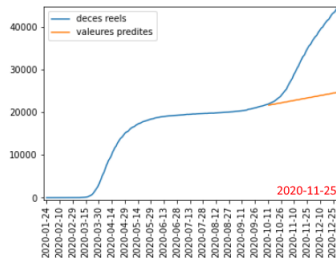
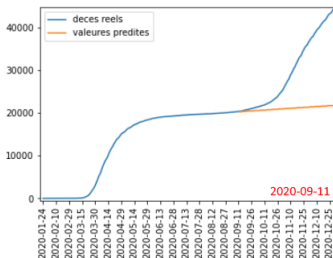
{'svr\_\_C' : 1000000}

Figure – à partir du 15/10/2020, puis du 01/11/2020

⇒ Échec de généralisation

# Prophet

## Approche avec le modèle Prophet de Facebook



# RegressorChain SVR

Approche multivariée avec **RegressorChain**

```
params = {
'svr__C' : [10**i for i in range(1,8)],
'svr__epsilon': [0.1,0.01,0.001]}
```

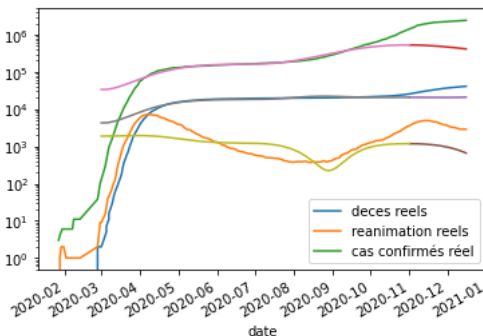
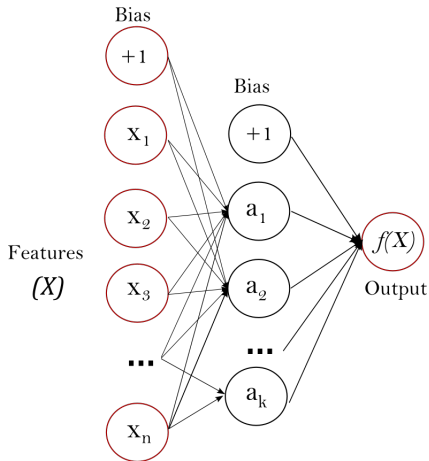


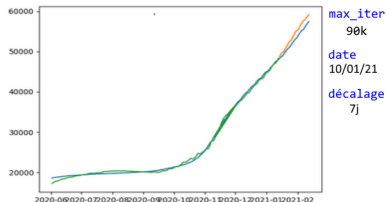
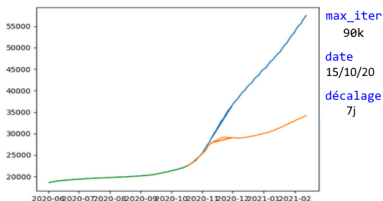
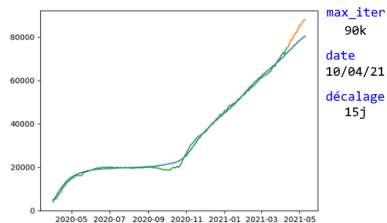
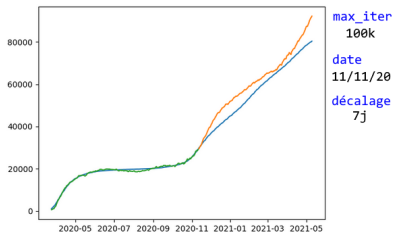
Figure – **RegressorChain** avec  $C = 10000$  et  $\epsilon = 0.001$

# Réseau neuronal

Approche avec des réseaux neuronaux



# Application avec réseau neuronal

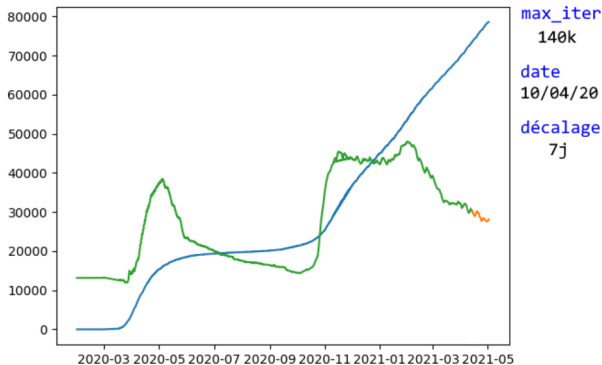




# Application avec réseau neuronal

Taux de corrélation entre

total\_cas\_confirmes et total\_deces\_hopital : 0.977939



⇒ échec du modèle sans la courbe des cas confirmés

# Conclusion

Plusieurs essais sur plusieurs modèles :

# Conclusion

Plusieurs essais sur plusieurs modèles :

⇒ Prédictions justes sur les périodes sans variations

# Conclusion

Plusieurs essais sur plusieurs modèles :

- ⇒ Prédictions justes sur les périodes sans variations
- ⇒ Difficulté de prédictions sans le point d'inflexion

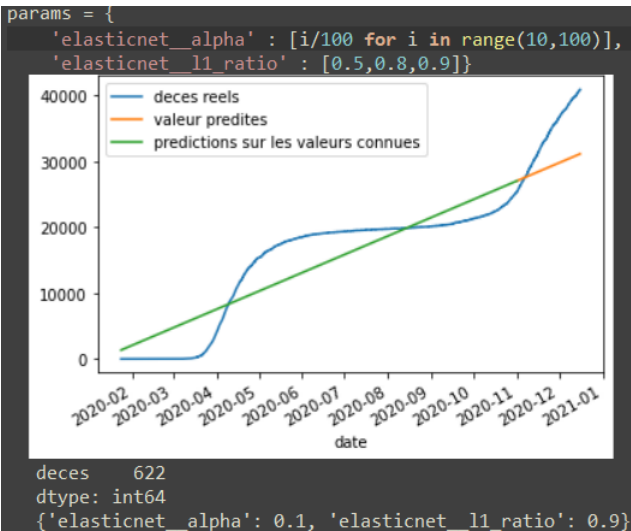
# fonction d'objectif d'ElasticNet

ElasticNet cherche  $\omega$  tel que :

$$\min_{\omega} \frac{1}{2n_{decès}} \|\hat{g}_{decès}(\omega, t) - f(t)\|_2^2 + \alpha \rho |\omega| + \frac{\alpha(1-\rho)}{2} \|\omega\|_2^2$$

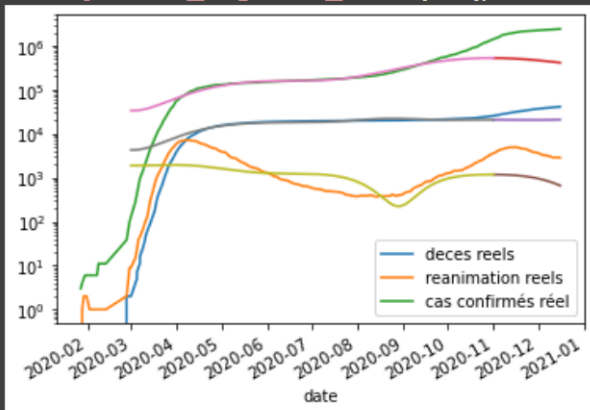
$\alpha$  et  $\rho$  les hyperparamètres définissant le modèle,  
 $f$  la courbe réelle des décès.

# Détails sur la regression lineaire



# Détails sur la regression multivariée

```
model = make_pipeline(StandardScaler(), RegressorChain(SVR(), order=[0,2,1])
params = {'regressorchain__base_estimator__C': [10**i for i in range(2,8)],
          'regressorchain__base_estimator__epsilon': [0.001,0.1,0.01],
          'regressorchain__base_estimator__kernel': ['rbf']}
```



```
{'regressorchain__base_estimator__C': 10000, 'regressorchain__base_estimator__epsilon': 0.001,
 'regressorchain__base_estimator__kernel': 'rbf'}
```

# Annexes

```
##Methode SVR##  
import matplotlib.pyplot as plt  
import pandas as pd  
  
from sklearn.pipeline import make_pipeline  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear_model import ElasticNet  
from sklearn.svm import SVR  
from sklearn.model_selection import GridSearchCV  
from sklearn.metrics import make_scorer  
from sklearn.metrics import mean_squared_error  
from sklearn.model_selection import TimeSeriesSplit
```



# Annexes

```
#####
svr = SVR()
params = {'svr_C' : [10**i for i in range(1,8)]}

#####Traitement de données#####
cdata = pd.read_csv('covid_numbers.csv',index_col='date',parse_dates=True)
cdata = cdata[cdata['granularite']=='pays']

for i in cdata:
    if not (cdata[i].name in ["deces"]):
        cdata.drop([i], axis=1, inplace = True)

cdata = cdata.dropna(axis=0)

print(cdata.count())
cdata

#####Rangement de données#####
```

# Annexes

```
date = '2020-10-15'

y = cdata[date]
X = pd.to_datetime(y.index)

size = len(X)

y = y.values.reshape(size,)
X = X.values.reshape(size,1)

X_train, y_train = X, y

model = make_pipeline(StandardScaler(), svr)
scorer = make_scorer(mean_squared_error, greater_is_better=False)
tscv = TimeSeriesSplit(n_splits=10)
grid = GridSearchCV(model, params, scorer, cv = tscv )

grid.fit(X_train, y_train)

x_prevu = pd.to_datetime(cdata[date:].index)
x_prevu = x_prevu.values.reshape(len(x_prevu),1)

cdata['deces'].plot(label="deces reels")
plt.plot(x_prevu,grid.predict(x_prevu),label="valeur predites")
plt.plot(X_train, grid.predict(X_train), label = "valeur predites")
plt.legend()
plt.show()
print(grid.best_params_)
```

# Annexes

```
### Multi regresseur avec le modele SVR ###  
  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.pipeline import make_pipeline  
from sklearn.preprocessing import StandardScaler  
from sklearn.svm import SVR  
from sklearn.model_selection import GridSearchCV  
from sklearn.multioutput import RegressorChain  
from sklearn.metrics import make_scorer  
from sklearn.metrics import mean_squared_error  
from sklearn.model_selection import TimeSeriesSplit
```

# Annexes

```
#####
tscv = TimeSeriesSplit(n_splits=15)
scorer = make_scorer(mean_squared_error, greater_is_better=False)
model = make_pipeline(StandardScaler(), RegressorChain(SVR(), order=[0,2,1]))
params = {'regressorchain__base_estimator__C': [10**i for i in range(2,8)],
          'regressorchain__base_estimator__epsilon': [0.001,0.1,0.01],
          'regressorchain__base_estimator__kernel': ['rbf']}

grid = GridSearchCV(model, params, cv = tscv, verbose = 1)

#####
cdata = pd.read_csv('covid_numbers.csv',index_col='date',parse_dates=True)

cdata = cdata[cdata['granularite']=='pays']
for i in cdata:
    if not (cdata[i].name in ["deces", "reanimation", "cas_confirmes"]):
        cdata.drop([i], axis=1, inplace = True)

cdata = cdata.dropna(axis=0)
```

# Annexes

```
date = '2020-11-01'

Y = cdata['2020-03-01':date]
x = pd.to_datetime(Y.index)

x = x.values.reshape(len(x),1)

grid.fit(x,Y)

x_futur = pd.to_datetime(cdata[date:].index)
x_futur = x_futur.values.reshape(len(x_futur),1)

cdata['deces'].plot(label="deces reels")
cdata['reanimation'].plot(label="reanimation reels")
cdata['cas_confirmes'].plot(label="cas confirmés réel")

plt.yscale('log')
plt.legend()
plt.plot(x_futur,grid.predict(x_futur),label="valeur predites")
plt.plot(x, grid.predict(x),label="valeur predites")

print(grid.best_params_)
```