Étude sur le propagation du Covid-19 avec machine learning

Pedro ALEXANDRINE

numero d'inscription : 41758

2020-2021

Sommaire

Utilisation faite de l'apprentissage automatique

Choix du modèle :
model

Utilisation faite de l'apprentissage automatique

Choix du modèle :
model

Entrainement sur données X :
model.fit(X)

Utilisation faite de l'apprentissage automatique

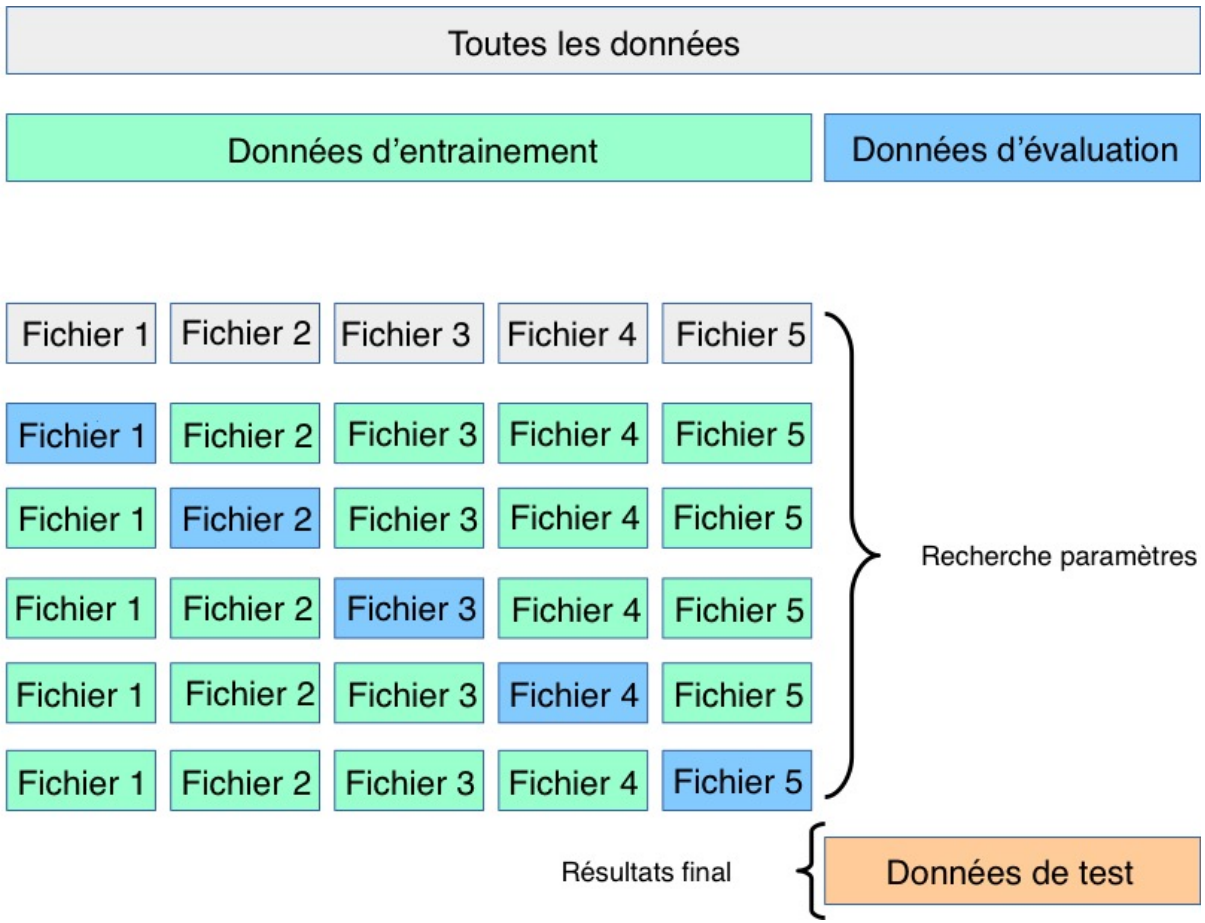
Choix du modèle :
model

Entrainement sur données X :
model.fit(X)

Utilisation du modèle :
model.predict(nouveauX)

Recherche du meilleur paramètre

Principe de la validation croisée:



SVR, premier résultat

Approche à l'aide du modèle SVR.
Noyau « rbf » $\rightarrow$ ajustement du paramètre C

— Modèle SVR (prédit les données entre le 02/12/20 et 16/12/20)

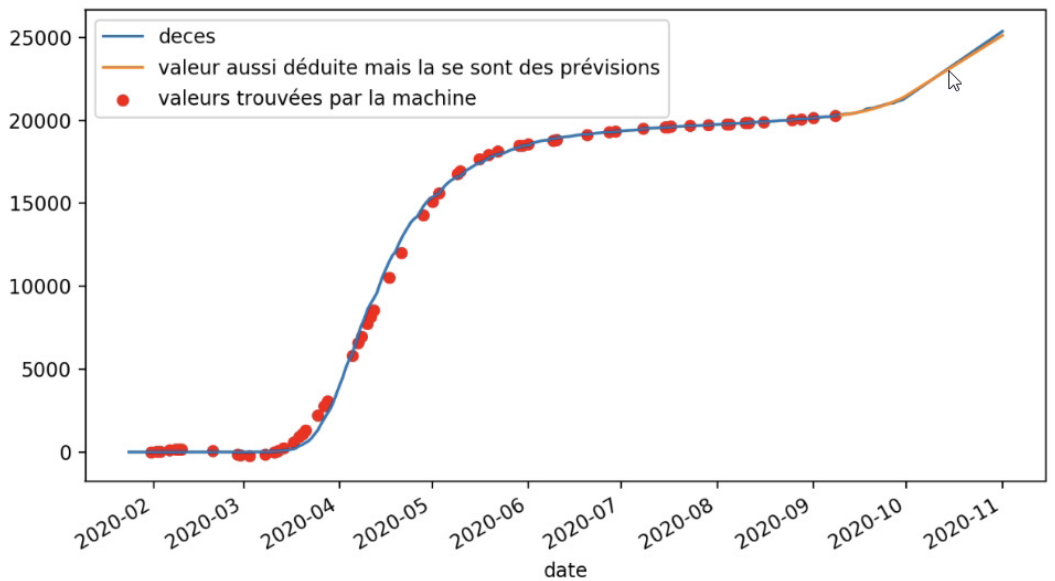


Figure: Premier résultat avec SVR et découpage inadapté, $C = 50k$

Découpage adapté pour la validation croisée

Remise en question de la méthode de découpage

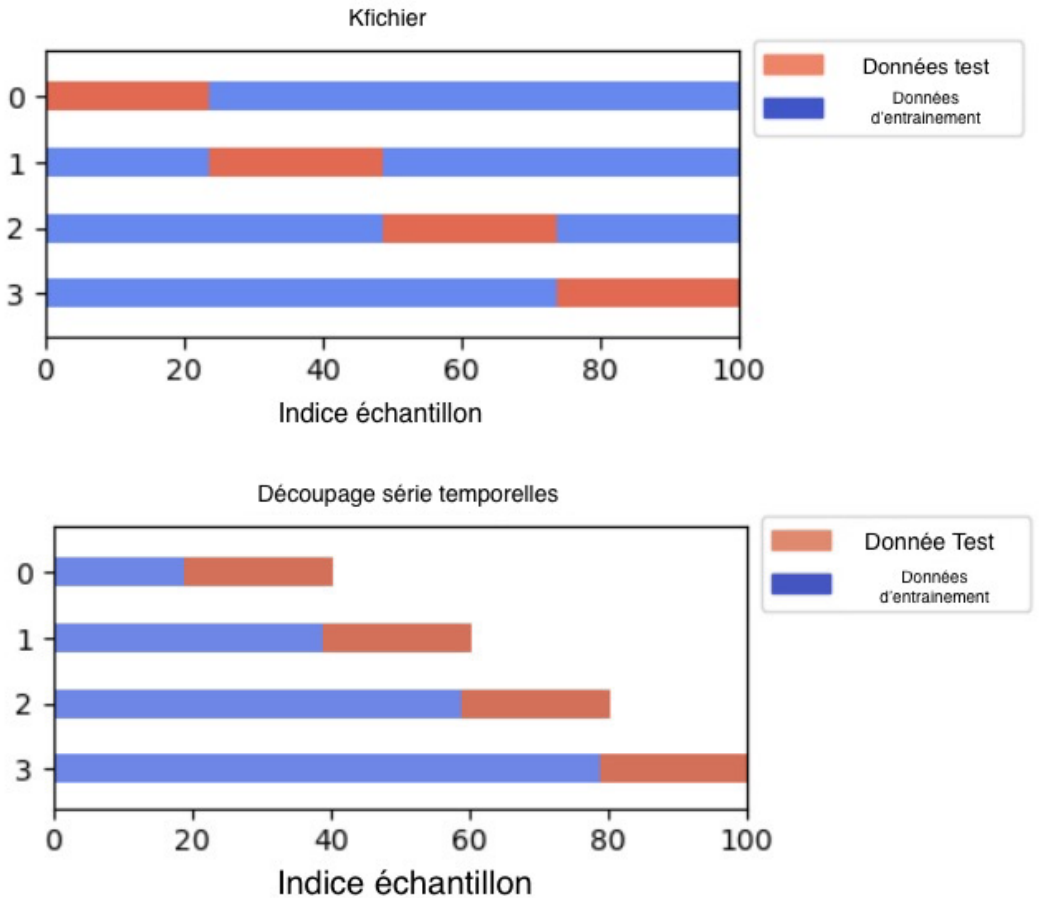


Figure: Comparaison des découpages pour la validation croisée

SVR

Avec $C = 10^5$

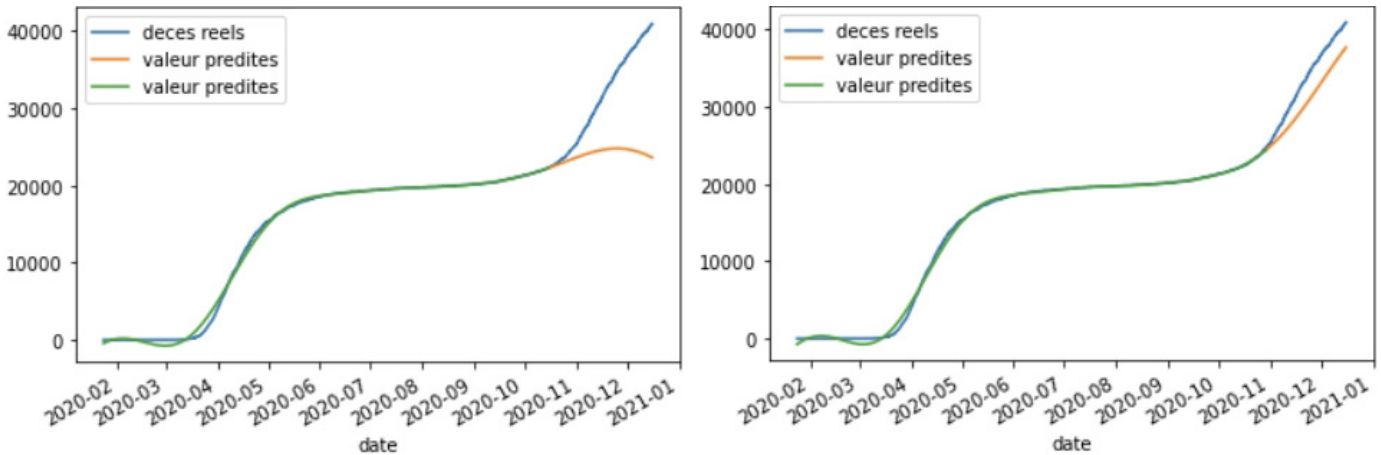


Figure: À gauche: prediction avant point d'inflexion; à droite: après.

$\Rightarrow$ Prédiction inefficace du point d'inflexion.

RegressorChain SVR

Multiregresseur RegressorChain
Corrélation : Cas confirmé → réanimation → décès

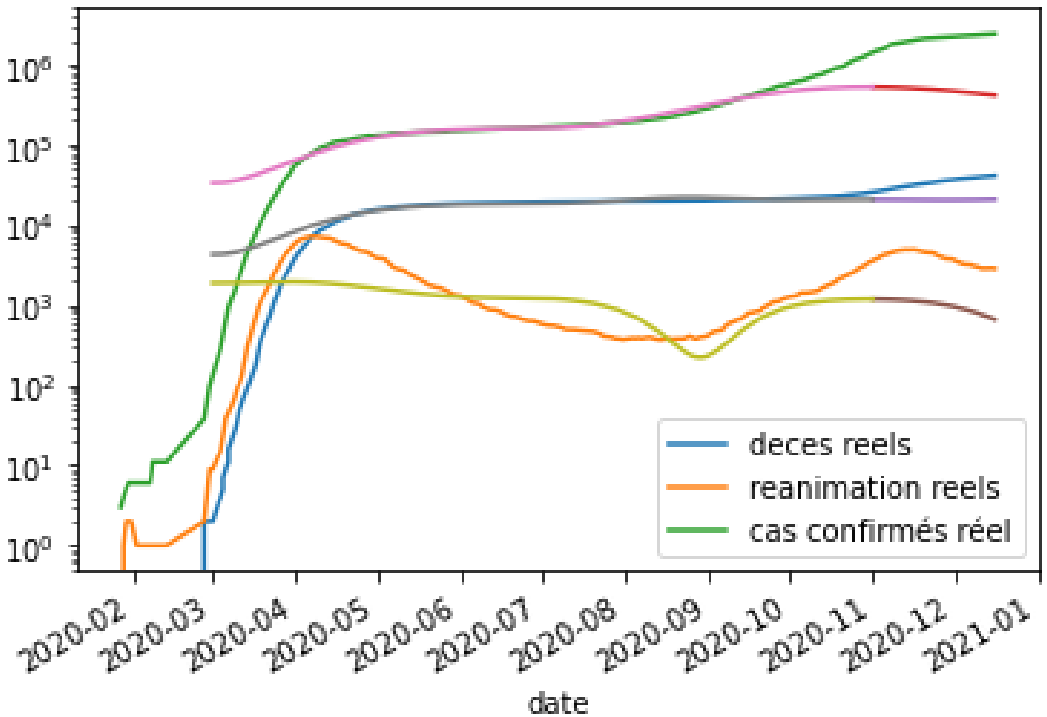


Figure: Résultat avec SVR insatisfaisant

RegressorChain TheilSenRegressor

Changement de régresseur : meilleurs résultats

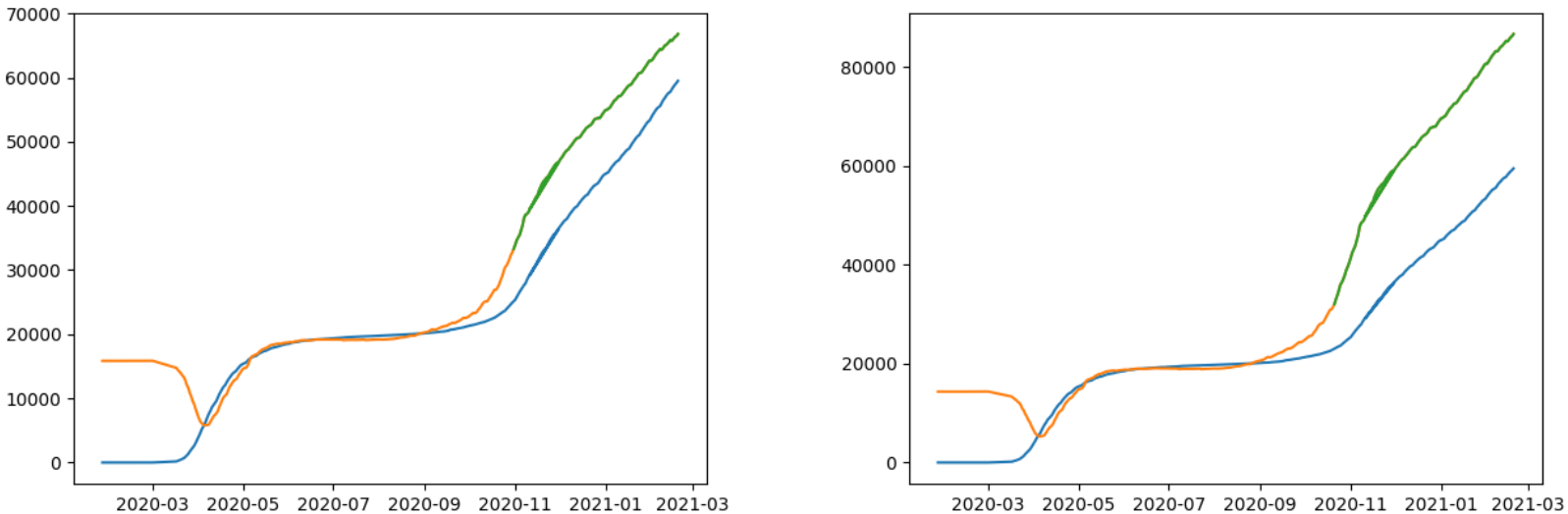


Figure: Prédiction avec et sans point d'inflexion

Principe du réseau neuronal

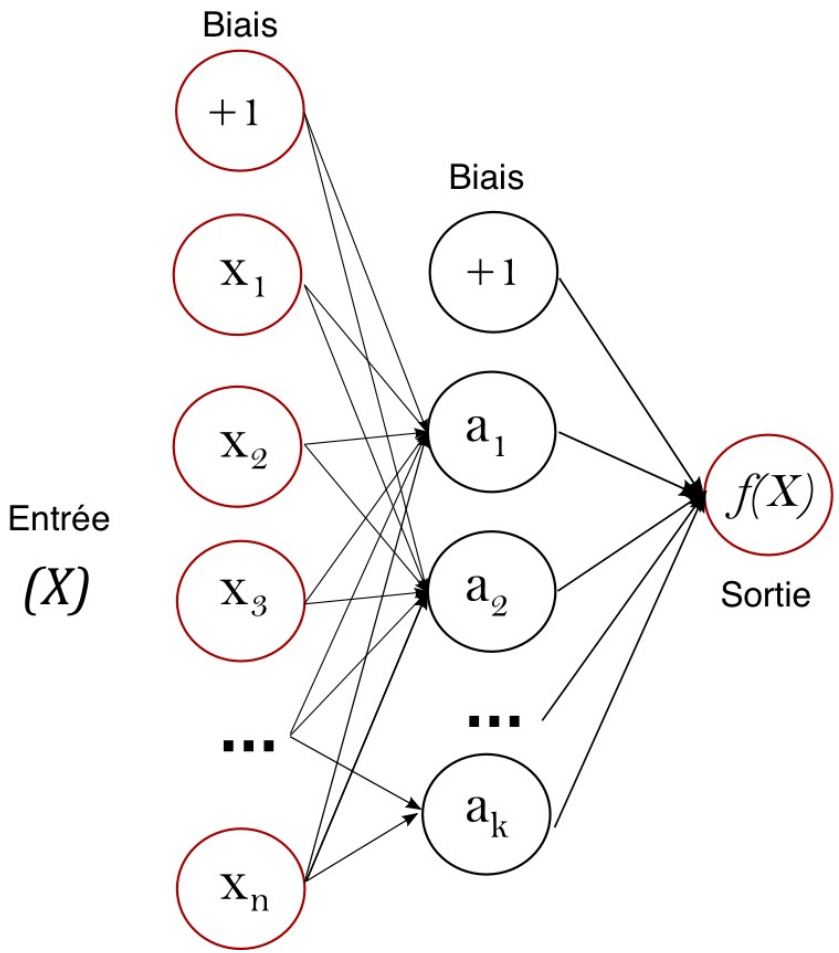
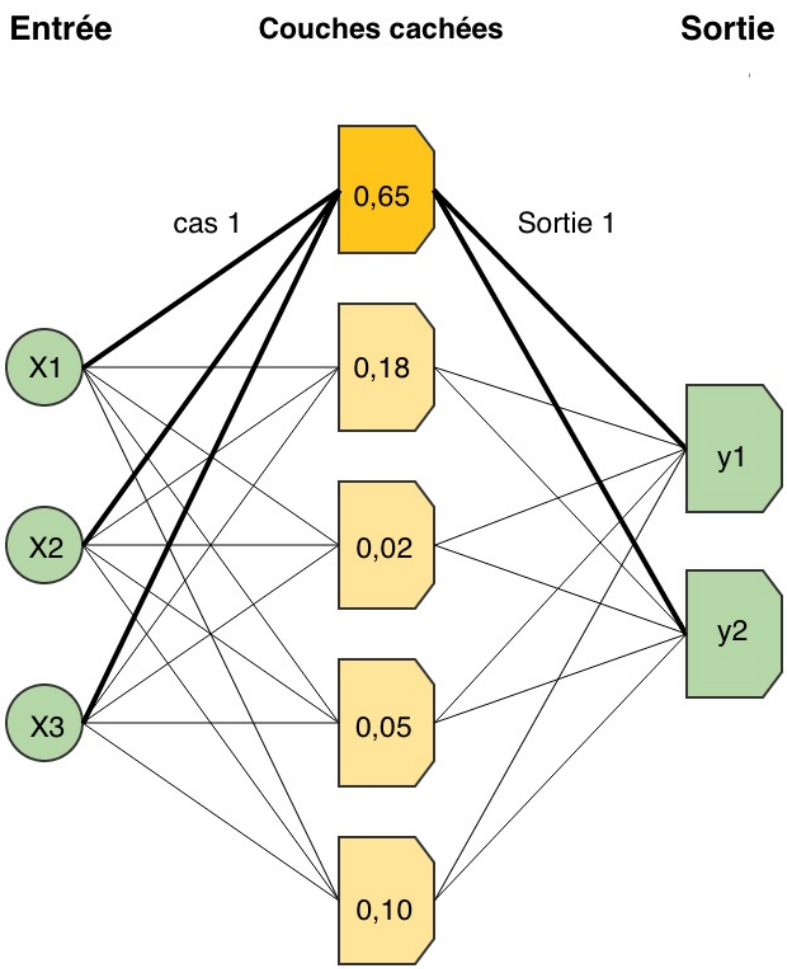


Illustration des idées

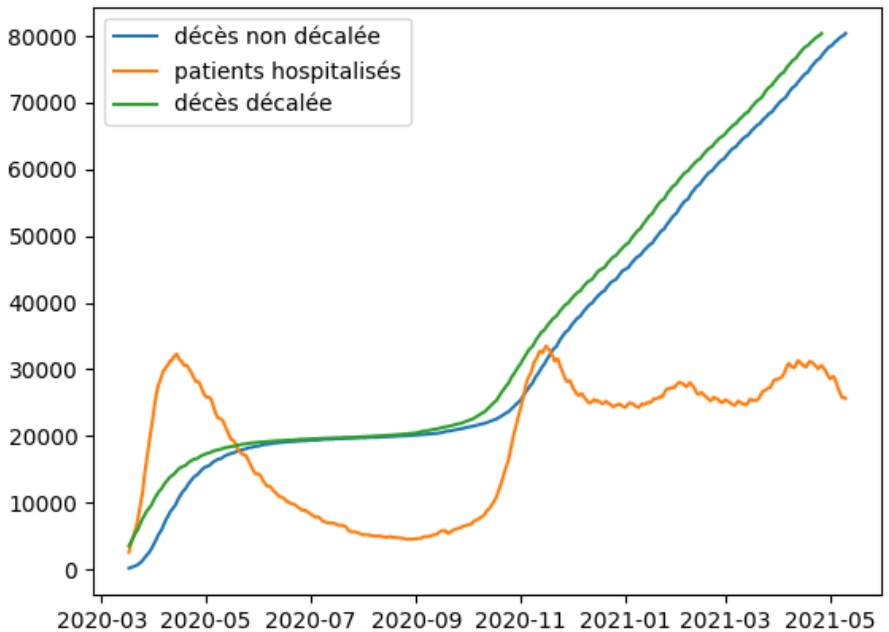


Décalage de courbes

Représentation avant et aprés décalage

date	
2020-03-17	175
2020-03-18	264
2020-03-19	372
2020-03-20	450
2020-03-21	562

date	
2020-03-17	1100
2020-03-18	1331
2020-03-19	1696
2020-03-20	1995
2020-03-21	2314



Corrélation et Premier résultat

Corrélation cas confirmés et décès : 0.978

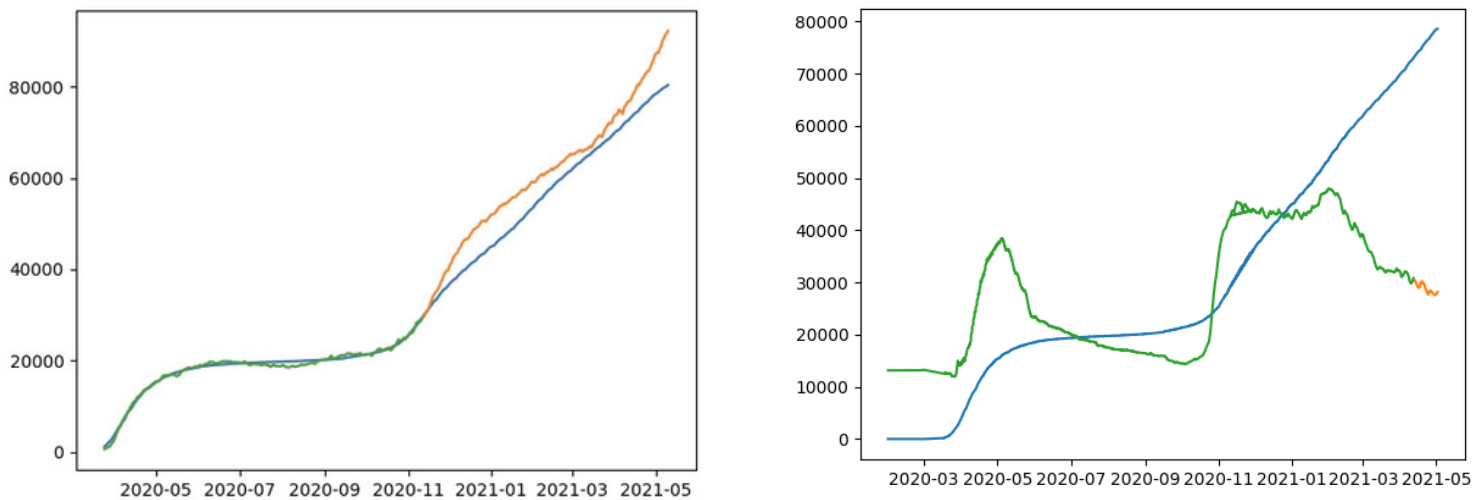
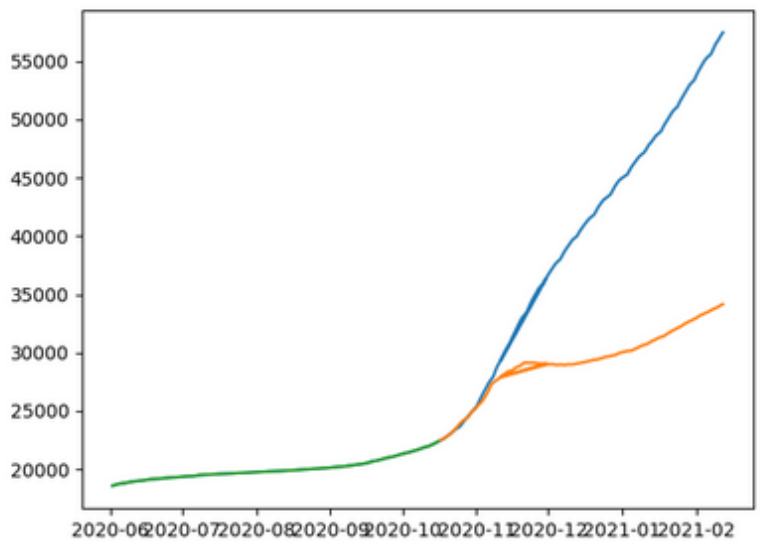


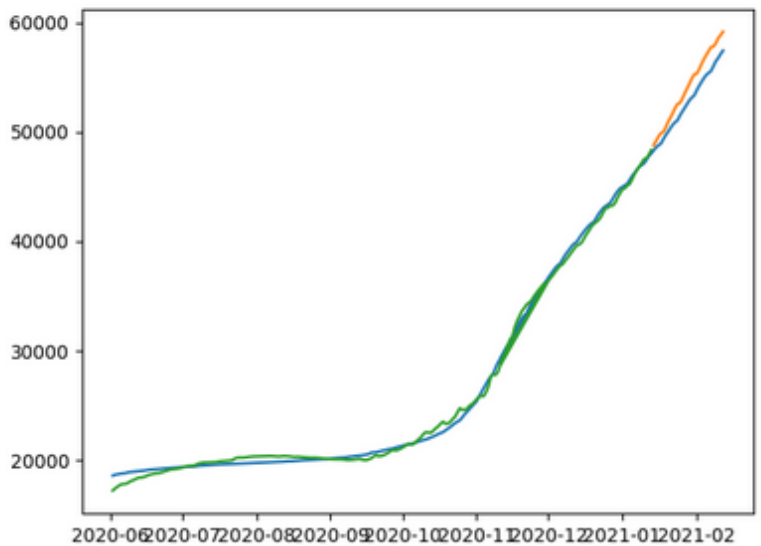
Figure: Tests avec et sans cas confirmés

Échec du modèle sans les cas confirmés → cohérent avec corrélation

Résultats

Réseaux neuronaux : Meilleurs paramètres





Neural network avec 7 jour de décalage; max_iter=90k

Augmentation du décalage

Résultats insatisfaisant

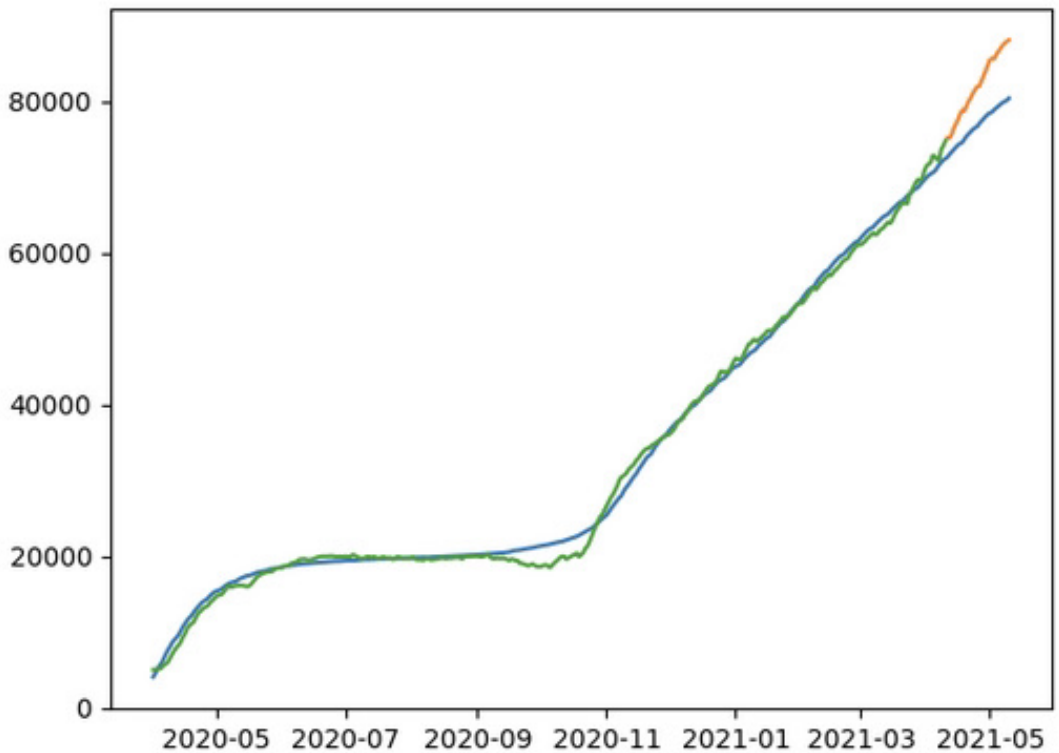


Figure: Tests avec 30 jours de décalage

Conclusion

Modèle peu fiable hors situation stabilisée ou début d'évolution

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.pipeline import make_pipeline
from sklearn.neural_network import MLPRegressor
from pandas.plotting import
    register_matplotlib_converters
register_matplotlib_converters()
from sklearn.metrics import make_scorer
from sklearn.metrics import mean_squared_error
scorer = make_scorer(mean_squared_error,
    greater_is_better=False)
from sklearn.preprocessing import
    StandardScaler
from sklearn.model_selection import
    TimeSeriesSplit
from sklearn.model_selection import
    GridSearchCV
```

```
def timeshift_day(data,n):
    data.index = data.index.shift(periods = n,
    freq ='D')


tscv = TimeSeriesSplit(n_splits=5)

covid = pd.read_csv('synthese-fra.csv',
    index_col='date',parse_dates=True)#,
    index_col='date',parse_dates=True)


for i in covid:
    if not (covid[i].name in ["
    total_deces_hopital","patients_reanimation',
                                    '
```

```
    patients_hospitalises','total_cas_confirmes'
    ]):
        covid.drop([i], axis=1, inplace = True)

jour_decale=7
covid = covid.dropna(axis=0)
timeshift_day(covid["total_deces_hopital"],-
    jour_decale)


full_size=covid.count()[0]
subject=['patients_reanimation',
         'patients_hospitalises','
    total_cas_confirmes']
result=["total_deces_hopital"]
#start:2020-03-17
predit_jour='2021-04-01'
```

```
X = covid[:predit_jour][subject]
y = covid[result][:predit_jour]
y = y.values.reshape(381,)
X_result=covid[predit_jour:][subject]
y_result=covid[result][predit_jour:]
y_result = y_result.values.reshape(40,)

params={
    'mlpregressor__max_iter': [10000],
    'mlpregressor__tol': [0.0001],
    'mlpregressor__n_iter_no_change': [2],
}

model=make_pipeline(StandardScaler(),
    MLPRegressor())


grid=GridSearchCV(model,param_grid=params,cv=
```

```
    tscv,scoring=scorer)


grid.fit(X,y)


timeshift_day(covid["total_deces_hopital"],
    jour_decale)
covid=covid[jour_decale:]


plt.figure()
plt.plot(covid.index,covid["total_deces_hopital
    "])
plt.plot(covid.index[374:],grid.predict(covid['
    2021-03-25':][subject]))
plt.plot(covid.index[:374],grid.predict(covid[:
    '2021-03-25'][subject]))
```

```
plt.show()
```

NN deces datagouv copie.py

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import
    train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import
    StandardScaler
from sklearn.svm import SVR
from sklearn.model_selection import
    GridSearchCV
from sklearn.linear_model import Lasso



covid = pd.read_excel('covid1.xlsx',index_col='
    date',parse_dates=True)
```

```
covid = covid[covid['granularite']=='pays']

covid.drop(['cas_ehpad', 'cas_confirmes_ehpad',
    'cas_possibles_ehpad','source_url',
            'deces_ehpad', 'reanimation', '
   gueris','source_nom',
            'nouvelles_hospitalisations', '
   nouvelles_reanimations','depistes',
            'maille_code', 'maille_nom','
   source_archive','source_type'],inplace=True,
   axis=1)

covid = covid.dropna(axis=0)

subject=['deces']

jour_prevu=75
```

```
size=covid.count()[0]-jour_prevu
X = covid.index[:size]
y = covid[:size][subject]



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

X_train, y_train=X, y

model=make_pipeline(StandardScaler(), SVR())
params={
    'svr__degree': [0],
    'svr__C': [600000],
}

grid = GridSearchCV(model,param_grid=params,cv
    =4)
```

```
grid.fit(X_train , y_train)



x_prevu = covid.index[size:]
x_prevu = x_prevu.values.reshape(len(x_prevu)
    ,1)
```

covid_deces copie.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import
    StandardScaler
from sklearn.model_selection import
    TimeSeriesSplit
```

```
from sklearn.svm import SVR
from sklearn.multioutput import RegressorChain
from sklearn.model_selection import
    GridSearchCV
from pandas.plotting import
    register_matplotlib_converters
register_matplotlib_converters()
from sklearn.metrics import make_scorer
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import
    RANSACRegressor
scorer = make_scorer(mean_squared_error,
    greater_is_better=False)
from sklearn.linear_model import
    TheilSenRegressor

tscv = TimeSeriesSplit(n_splits=10)
```

```
covid = pd.read_excel('chiffre_covid_14032021.
    xlsx',index_col='date',parse_dates=True)#,
    index_col='date',parse_dates=True)

covid = covid[covid['granularite']=='pays']

for i in covid:
    if not (covid[i].name in ["deces",'
    cas_confirmes','reanimation','hospitalises'
    ]):
        covid.drop([i], axis=1, inplace = True)
covid=covid[5:]

covid = covid.dropna(axis=0)

subject=['hospitalises','cas_confirmes','
    reanimation']
result=["deces"]
```

```
size=255

X = covid[:size][subject]
y = covid[:size][result]
X_result=covid[size:][subject]
y_result=covid[size:][result]

y = y.values.reshape(size,)
y_result = y_result.values.reshape(348-size,)

params={
    'theilsenregressor__tol': [0.001],
    'theilsenregressor__random_state': [i for i
    in range(45)],
    'theilsenregressor__max_iter':  [90000],

}
```

```
model=make_pipeline(StandardScaler(),
    TheilSenRegressor())


grid=GridSearchCV(model,param_grid=params,cv=
    tscv)

grid.fit(X,y)


p=covid[subject]

plt.figure()
plt.plot(covid.index,covid['deces'])
plt.plot(covid.index[size:],grid.predict(
    X_result))
```

```
plt.show()
```

multi regressor tipe copie.py