02_KNN

December 2, 2021

[1]:
```
# ZOOM de la SFGP - GTIAP
# 02 / 12 / 2021
# Roda Bounaceur
# LRGP - Nancy
# roda.bounaceur@univ-lorraine.fr
#
#
# Importation des bibliothèques de bases - Pandas et Numpy - pour manipuler les␣
↪data
#
import pandas as pd
import numpy as np
#
# Graphes
#
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.pyplot import plot
%matplotlib inline
#
# Packages des algorithmes de ML
#
import sklearn
```

[2]:
```
#
# Versions utilisées
#
print('Version de matplotlib = ',matplotlib.__version__)
print('Version de sklearn = ',sklearn.__version__)
!python -V
```

```
Version de matplotlib =  3.3.4
Version de sklearn =  0.24.1
Python 3.8.8
```

[3]:
```
#
# Importation du dataset
#
```

```
df =  pd.read_csv('Dataset_Complet.csv',sep = ',')
```

[4]:
```
#
# Affectation des features et des targets
#
X  = df[['Pressure_(bar)', 'Resident_Time_(s)', 'Temperature_(C)',␣
 ↪'Time_(sec)']]
y = df.drop(['Pressure_(bar)', 'Resident_Time_(s)', 'Temperature_(C)',␣
 ↪'Time_(sec)'],axis=1)
```

[5]:
```
#
# Répartition des valeurs "train" et "test"
#
from sklearn.model_selection import train_test_split  # méthode pour le train /␣
 ↪test
#
X_train , X_test , y_train , y_test = train_test_split( X , y , test_size=0.2 ,␣
 ↪shuffle=True , random_state=4 )
```

[6]:
```
#
# Méthode de ML : KNN
#
from sklearn.neighbors import KNeighborsRegressor # methode KNN regression
#
# Metrics -
#
from sklearn.metrics import mean_squared_error # erreur MSE
from sklearn.metrics import mean_absolute_error # MAE
```

[7]:
```
#
# Affectation de l'estimateur
#
Modele_KNN = KNeighborsRegressor()
```

[8]:
```
#
# Entrainement de la méthode avec les paramètres par défaut
#
Modele_KNN.fit( X_train , y_train )
```

[8]: KNeighborsRegressor()

[9]:
```
#
# Affichage des scores
#
print('R2_score')
print('KNN R2 train = ' ,  Modele_KNN.score(X_train,y_train))
print('KNN R2 test = ' ,  Modele_KNN.score(X_test,y_test))
```

```
R2_score
KNN R2 train =  0.9995255798604405
KNN R2 test =  0.9989076549162693
```

[10]:
```
#
# Affichage des scores
#
print('MAE')
print('KNN MAE train = ' ,  mean_absolute_error(y_train,Modele_KNN.
 ↪predict(X_train)))
print('KNN MAE test = ' ,  mean_absolute_error(y_test,Modele_KNN.
 ↪predict(X_test)))
#
# Affichage des scores
#
print('MSE')
print('KNN MSE train = ' ,  mean_squared_error(y_train,Modele_KNN.
 ↪predict(X_train)))
print('KNN MSE train = ' ,  mean_squared_error(y_train,Modele_KNN.
 ↪predict(X_train)))
```

```
MAE
KNN MAE train =  8.071257215842221e-05
KNN MAE test =  0.00013580976254289897
MSE
KNN MSE train =  4.95376100840807e-07
KNN MSE train =  4.95376100840807e-07
```

[11]:
```
#
# Affichage des hyper-paramètres disponible pour la méthode
#
Modele_KNN.get_params()
```

[11]:
```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'uniform'}
```

[12]:
```
#
# Analyse paramètrique sur un paramètre
#
n_neighbors = [x for x in range(1,15)]
#
for i in (n_neighbors):
```

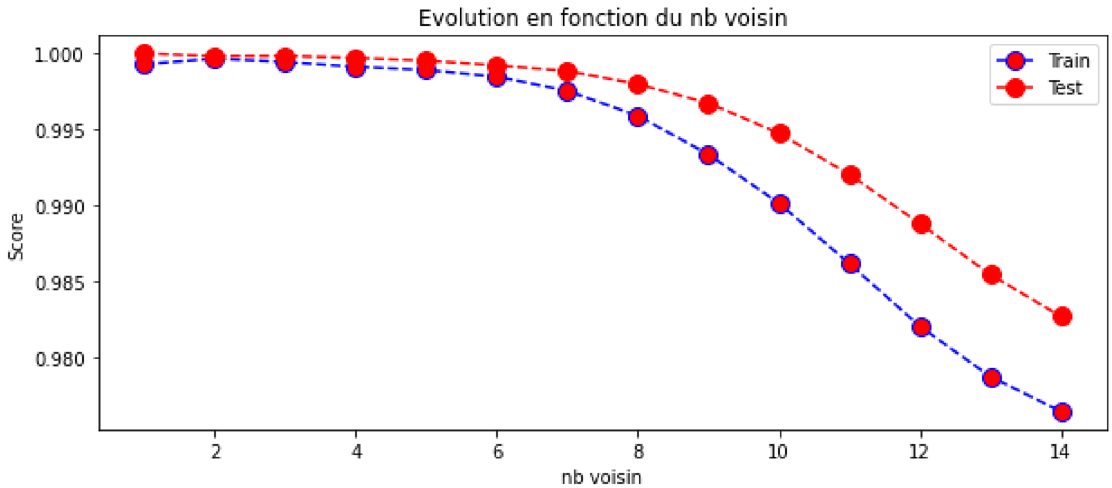
```
    Modele_KNN = KNeighborsRegressor(n_neighbors=i)
    Modele_KNN.fit( X_train , y_train )
    print('n_neighbors = ',i,'- KNN R2 train = ' ,  Modele_KNN.
↪score(X_train,y_train))
```

```
n_neighbors =  1 - KNN R2 train =  0.9999999990297944
n_neighbors =  2 - KNN R2 train =  0.9998172707584947
n_neighbors =  3 - KNN R2 train =  0.9998356948649295
n_neighbors =  4 - KNN R2 train =  0.9997015380413568
n_neighbors =  5 - KNN R2 train =  0.9995255798604405
n_neighbors =  6 - KNN R2 train =  0.9992164852623338
n_neighbors =  7 - KNN R2 train =  0.9988390997460975
n_neighbors =  8 - KNN R2 train =  0.997979909120591
n_neighbors =  9 - KNN R2 train =  0.9967337699160571
n_neighbors =  10 - KNN R2 train =  0.9947424937770087
n_neighbors =  11 - KNN R2 train =  0.9920331430350725
n_neighbors =  12 - KNN R2 train =  0.9887870666377916
n_neighbors =  13 - KNN R2 train =  0.9854397935890996
n_neighbors =  14 - KNN R2 train =  0.9826805579861378
```

[13]:
```
#
# Analyse paramètrique sur un paramètre - Résultat sous forme graphique
#
score_rate = []
score_train_rate = []
for i in range(1,15):
    Modele_KNN = KNeighborsRegressor(n_neighbors=i)
    Modele_KNN.fit( X_train , y_train )
    score_train_rate.append(Modele_KNN.score(X_train,y_train))
    score_rate.append(Modele_KNN.score(X_test,y_test))

plt.figure(figsize=(10,4))
plt.
↪plot(range(1,15),score_rate,color='blue',linestyle='dashed',marker='o',markerfacecolor='red
plt.
↪plot(range(1,15),score_train_rate,color='red',linestyle='dashed',marker='o',markerfacecolor
plt.title('Evolution en fonction du nb voisin')
plt.xlabel('nb voisin')
plt.ylabel('Score')
plt.legend()
```

[13]: <matplotlib.legend.Legend at 0x1ee84b9fbe0>



[ ]:
```
#
# Cross-Validation - Influence des hyper-paramètres
#
```

[14]:
```
#
# Affichage des hyper-paramètres disponible pour la méthode
#
Modele_KNN.get_params()
```

[14]:
```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 14,
 'p': 2,
 'weights': 'uniform'}
```

[15]:
```
#
# Analyse paramètrique sur plusieurs paramètres
#
n_neighbors = [x for x in range(1,15)]
weights = ['uniform','distance']
metric = ['minkowski' ,'euclidean' , 'manhattan']
#
# Dictionnaire
#
param_grid = {'n_neighbors': n_neighbors , 'weights': weights , 'metric':␣
 ↪metric}
```

[16]:
```python
#
# Cross-Validation - GridSearchCV
#
from sklearn.model_selection import GridSearchCV
#
Modele_KNN = KNeighborsRegressor()
#
grid = GridSearchCV( estimator=Modele_KNN , param_grid = param_grid , cv=3 ,␣
↪verbose=2 )
#
grid.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 84 candidates, totalling 252 fits
[CV] END …metric=minkowski, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=6, weights=distance; total time=   0.0s
```

```
[CV] END ..metric=minkowski, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=minkowski, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=minkowski, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END .metric=minkowski, n_neighbors=14, weights=distance; total time=   0.0s
```

```
[CV] END .metric=minkowski, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=8, weights=distance; total time=   0.0s
```

```
[CV] END ..metric=euclidean, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=euclidean, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=euclidean, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END .metric=euclidean, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=1, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=1, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=2, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=2, weights=distance; total time=   0.0s
```

```
[CV] END ..metric=manhattan, n_neighbors=2, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=3, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=4, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=4, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=5, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=5, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=6, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=6, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=7, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=7, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=8, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=8, weights=distance; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END …metric=manhattan, n_neighbors=9, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=9, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=10, weights=uniform; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=10, weights=distance; total time=   0.0s
```

```
[CV] END .metric=manhattan, n_neighbors=10, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=11, weights=uniform; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=11, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=12, weights=uniform; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=12, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=13, weights=uniform; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=13, weights=distance; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END ..metric=manhattan, n_neighbors=14, weights=uniform; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=14, weights=distance; total time=   0.0s
[CV] END .metric=manhattan, n_neighbors=14, weights=distance; total time=   0.0s
```

[16]:
```
GridSearchCV(cv=3, estimator=KNeighborsRegressor(),
             param_grid={'metric': ['minkowski', 'euclidean', 'manhattan'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                         13, 14],
                         'weights': ['uniform', 'distance']},
             verbose=2)
```

[17]:
```python
#
# Affiche les meilleurs hyperparametres trouvés
#
grid.best_params_
```

[17]:
```
{'metric': 'manhattan', 'n_neighbors': 2, 'weights': 'distance'}
```

[18]:
```python
#
# Avec ces hyperparamètres, affiche le meilleur score sur le train
#
grid.best_estimator_.score(X_train,y_train)
```

[18]:
```
0.9999999995148973
```

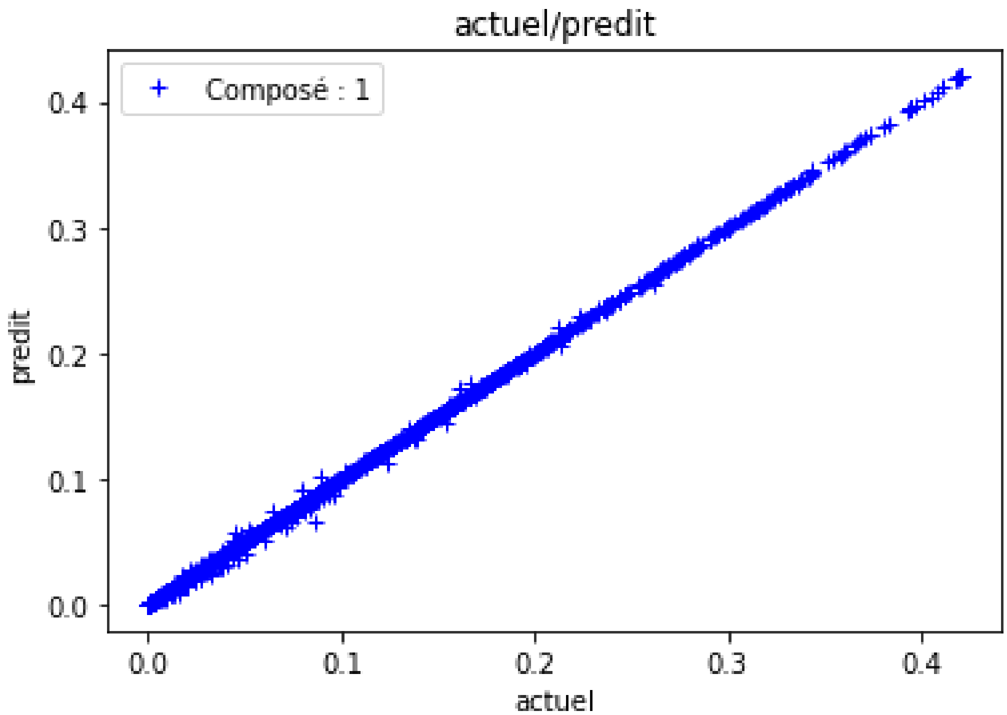
[19]:
```
#
# Avec ces hyperparamètres, affiche le meilleur score sur le test
#
grid.best_estimator_.score(X_test,y_test)
```

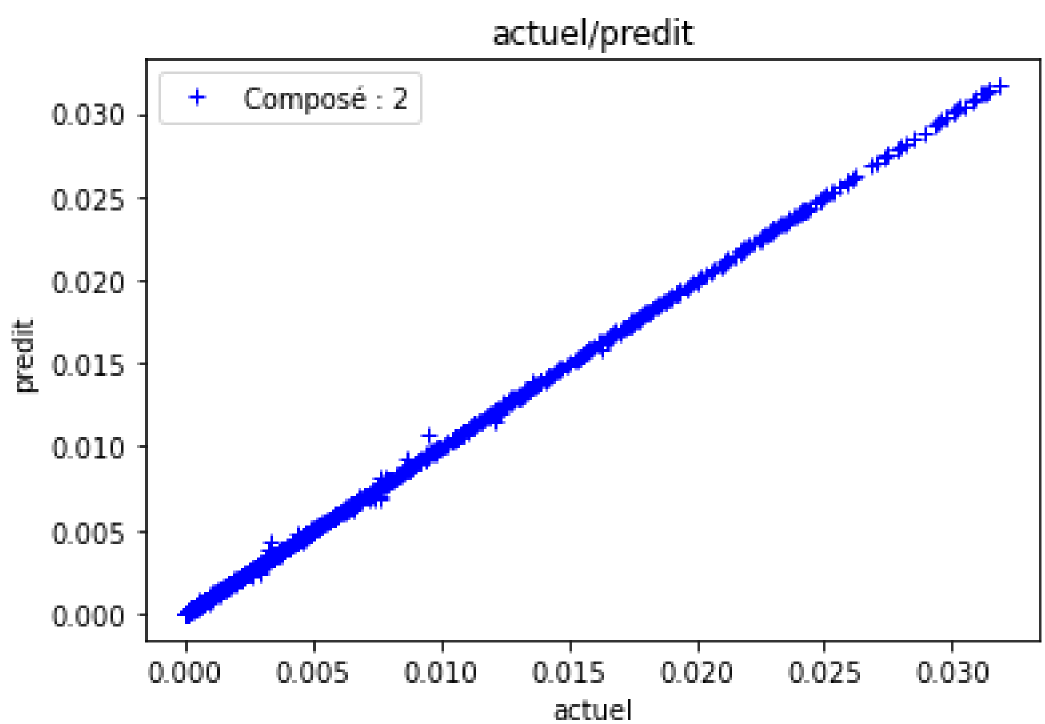
[19]:
```
0.9997595859006833
```

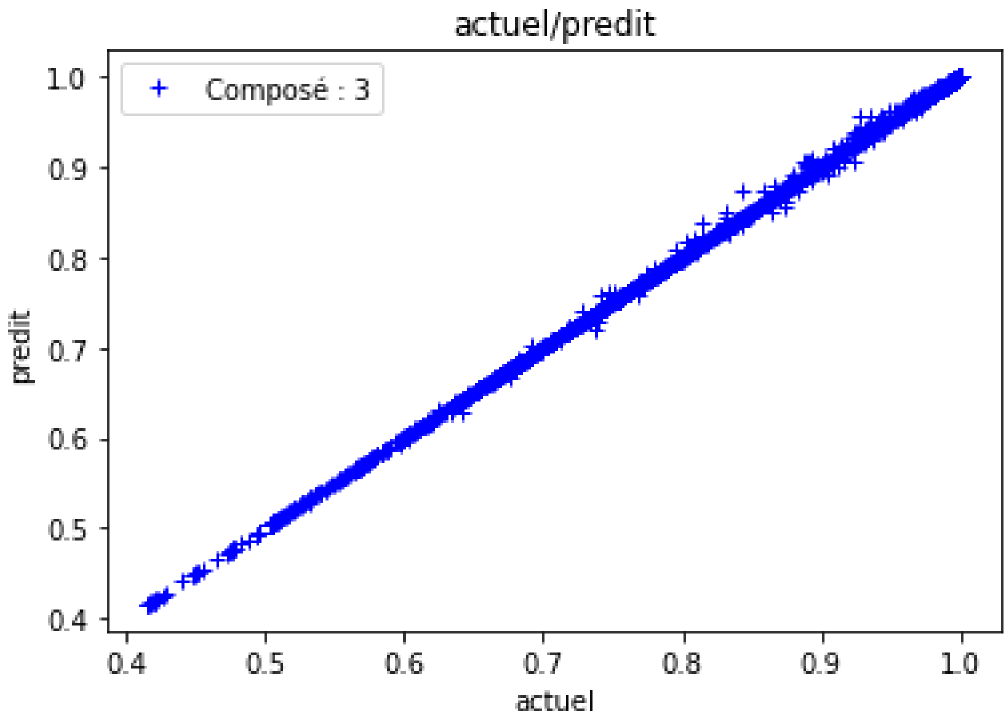
[20]:
```
#
# Génération d'une prédiction
#
grid.best_estimator_.predict([[0.003,.2,850,1.5]])
```

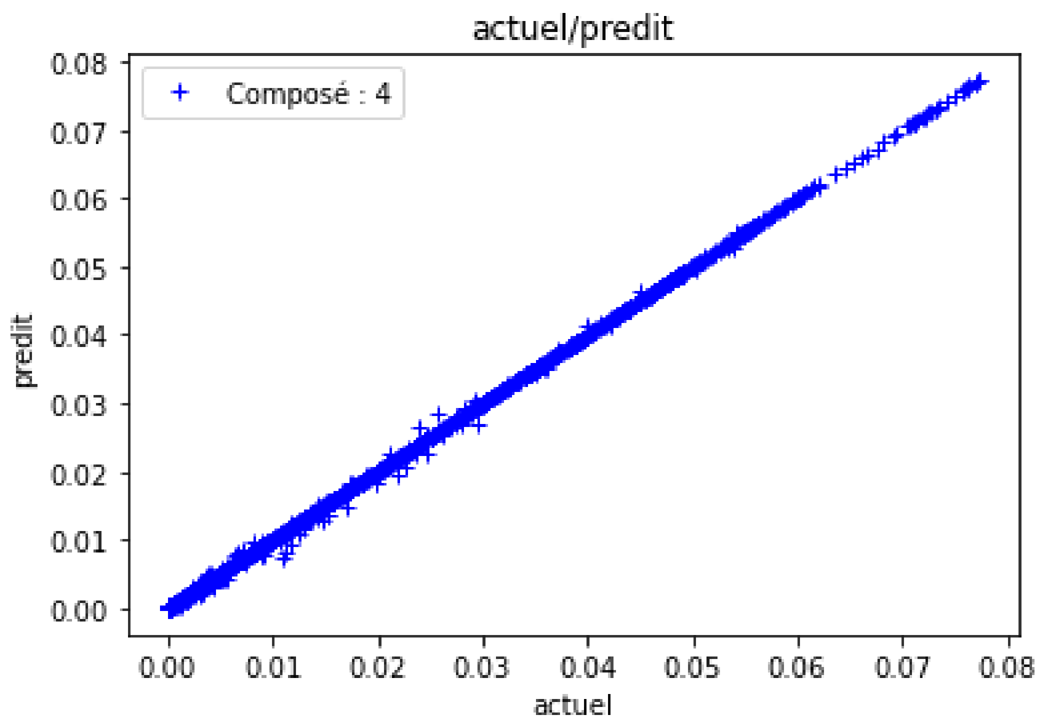
[20]:
```
array([[5.83000000e-03, 1.04000000e-06, 9.74000000e-01, 3.23000000e-04,
        1.07000000e-05, 1.56000000e-02, 1.63000000e-05, 3.11000000e-08,
        3.53000000e-03, 1.52000000e-04, 1.25000000e-07, 1.14000000e-04,
        5.58604167e-05, 2.62000000e-05, 1.20000000e-08, 4.78604167e-07,
        5.17604167e-08, 7.36604167e-06, 1.42000000e-09, 1.35604167e-08,
        2.17604167e-09, 3.10604167e-08, 2.89208333e-08, 1.38000000e-09,
        1.62000000e-09, 8.29416667e-10, 7.08229167e-12, 3.01208333e-12,
        1.15604167e-10, 2.34604167e-11, 3.02000000e-10]])
```

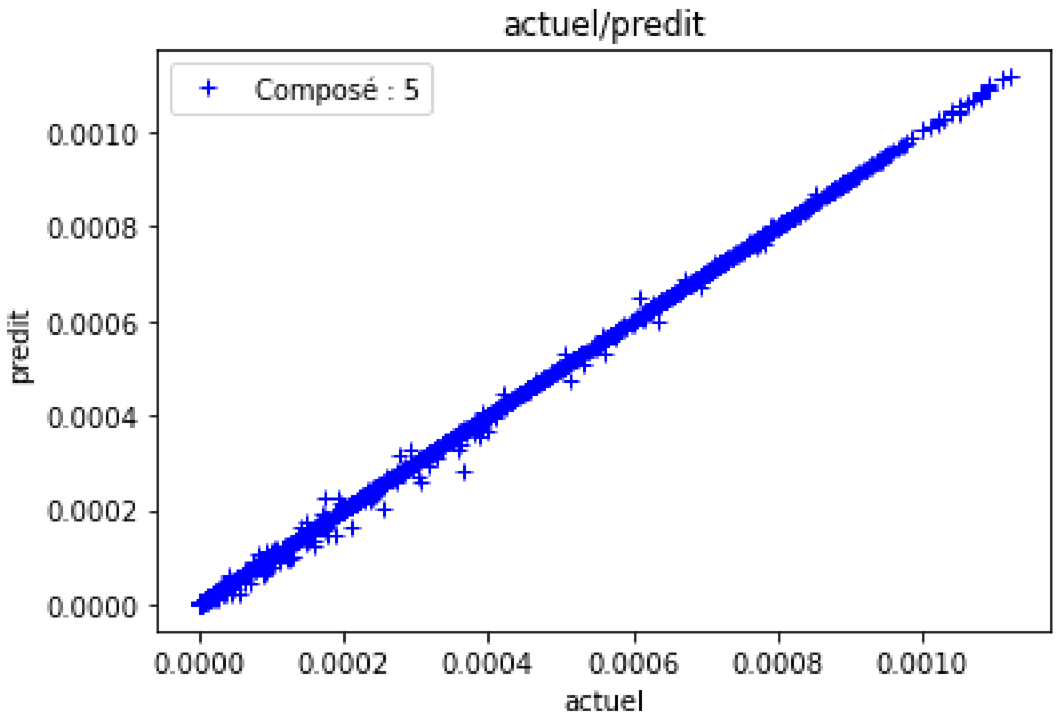
[21]:
```
#
# Courbes de comparaisons y_prédit vs y_test
#
from Ytest_Ypred_1 import Ytest_Ypred # Fonction personnelle
#
Ytest_Ypred( grid.best_estimator_, X_test , y_test )
```

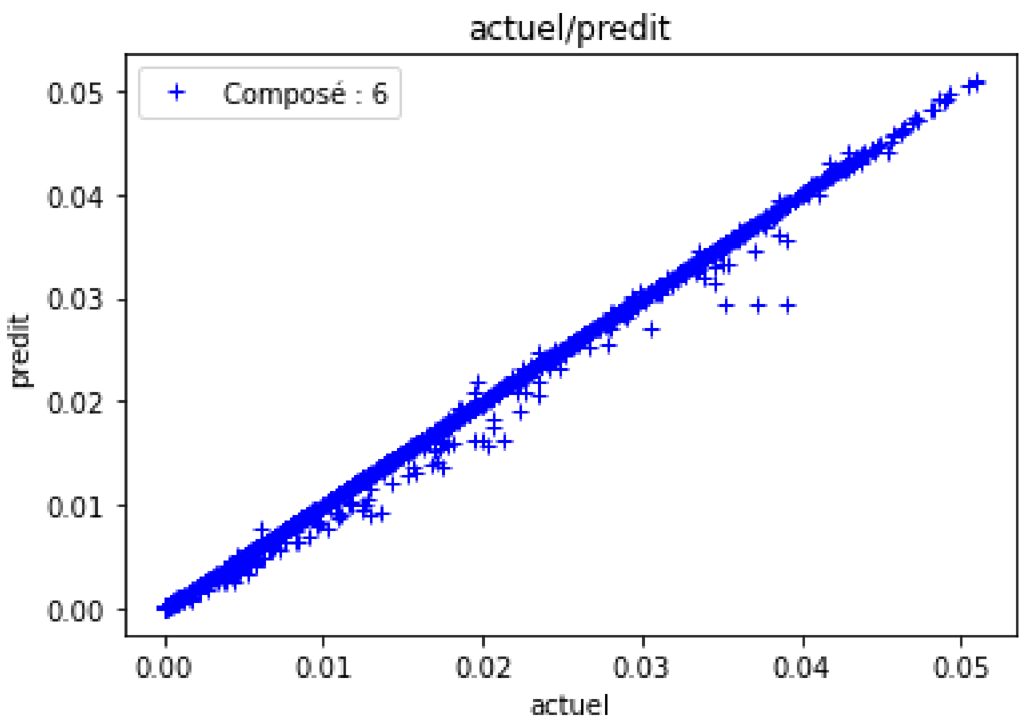

actuel/predit
Composé : 1
predit
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
actuel

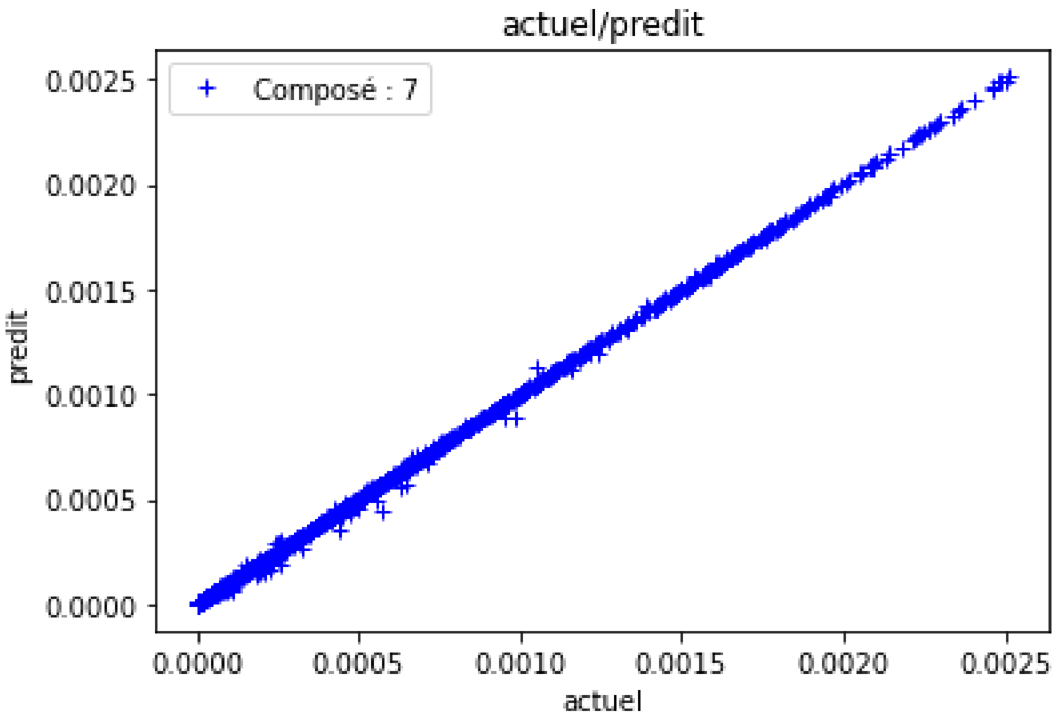

actuel/predit
Composé : 2
predit
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.000
0.005
0.010
0.015
0.020
0.025
0.030
actuel

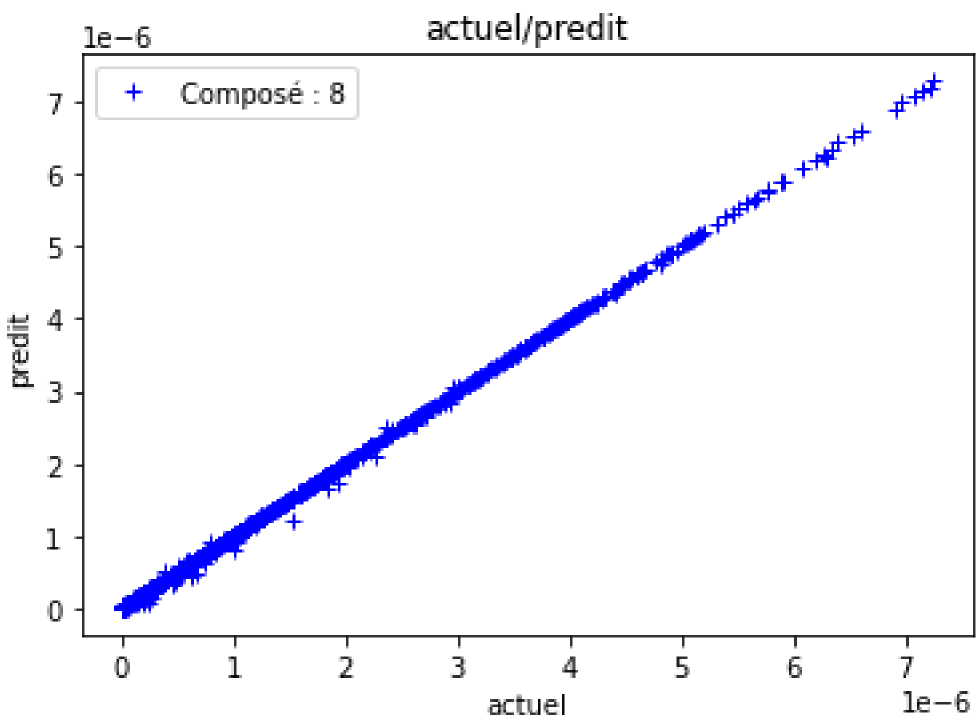

actuel/predit
Composé : 3
predit
actuel
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.4
0.5
0.6
0.7
0.8
0.9
1.0

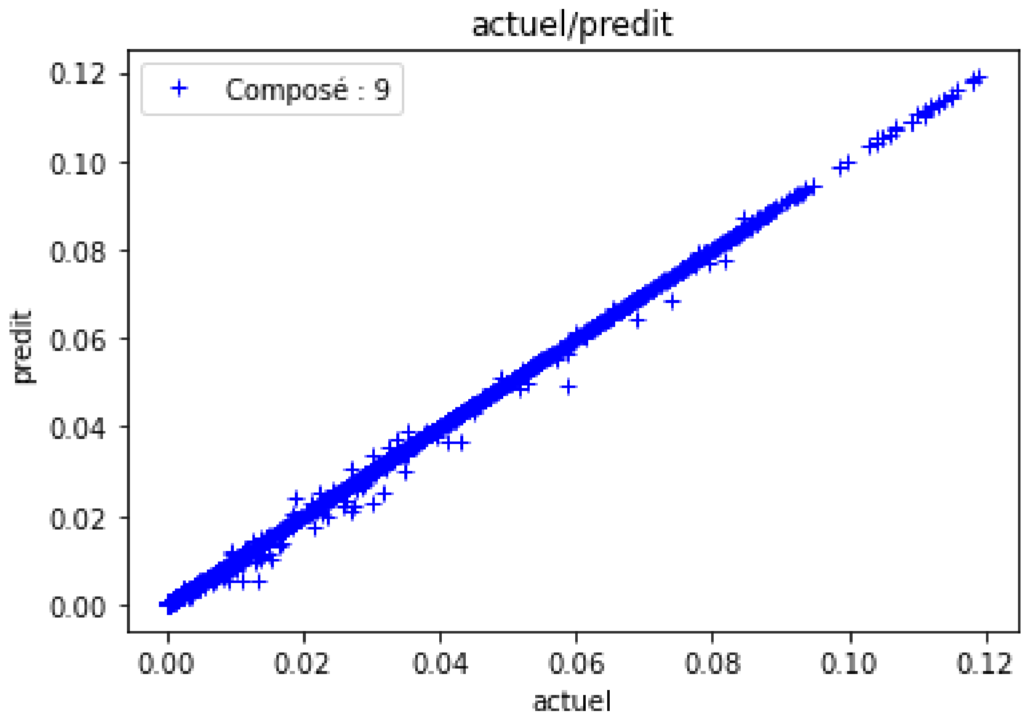

actuel/predit
Composé : 4
predit
actuel
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


actuel/predit
Composé : 5
predit
actuel
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010

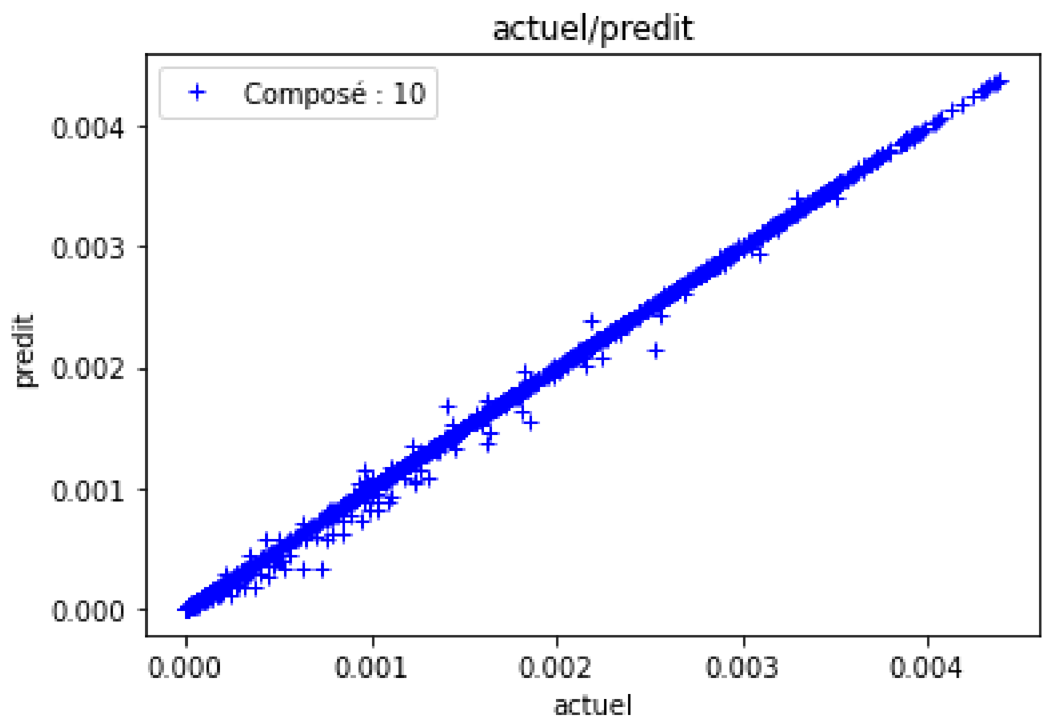

actuel/predit
Composé : 6
predit
actuel
0.00
0.01
0.02
0.03
0.04
0.05


actuel/predit
Composé : 7
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
predit
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
actuel

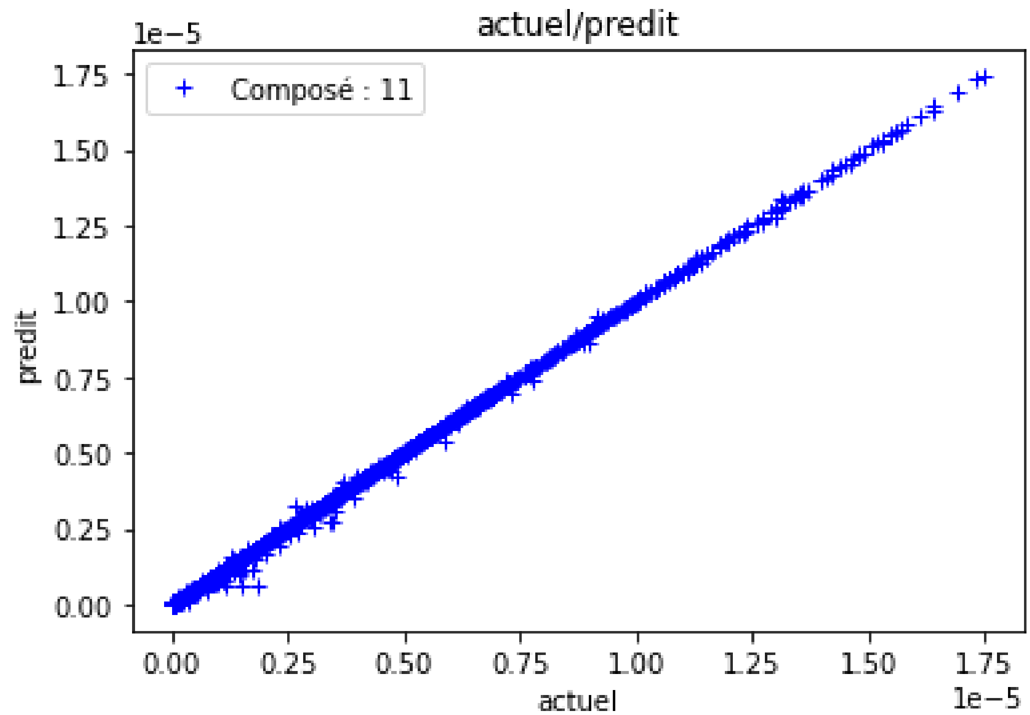

actuel/predit
1e−6
Composé : 8
7
6
5
4
3
2
1
0
predit
0
1
2
3
4
5
6
7
actuel
1e−6

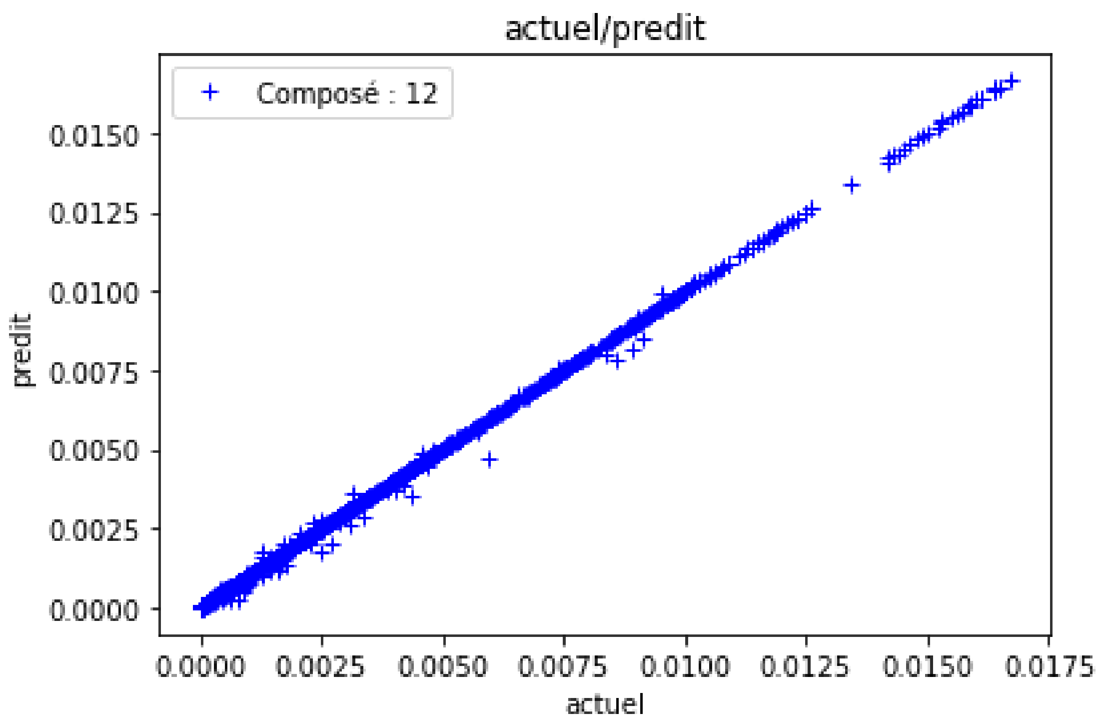


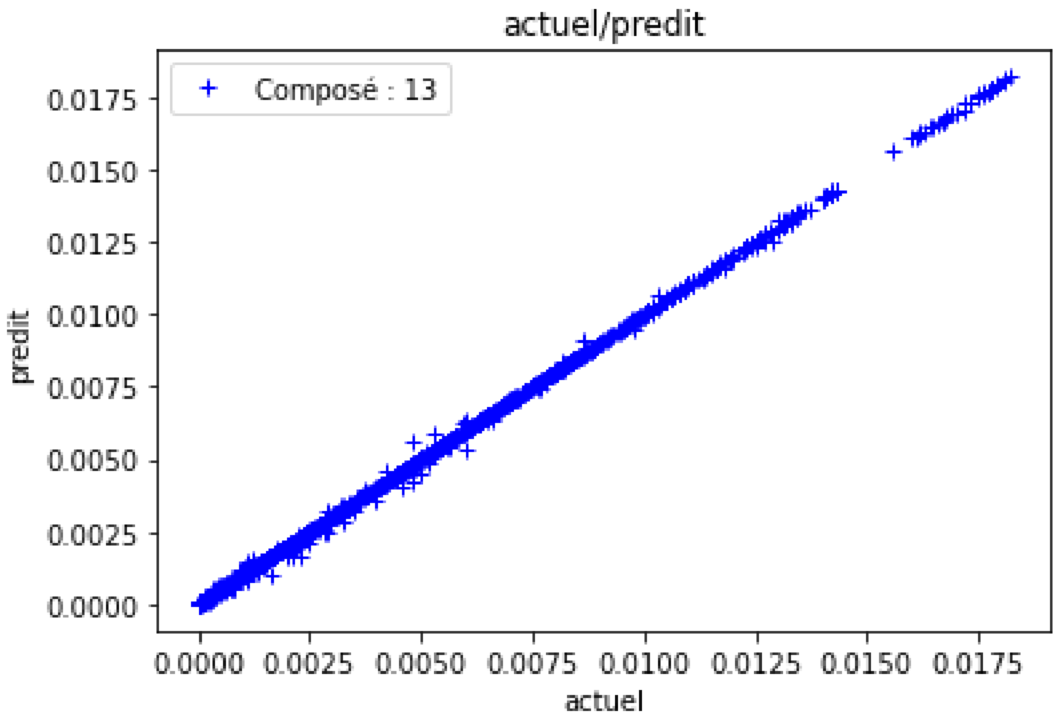
actuel/predit
Composé : 9
0.12
0.10
0.08
0.06
0.04
0.02
0.00
predit
0.00
0.02
0.04
0.06
0.08
0.10
0.12
actuel

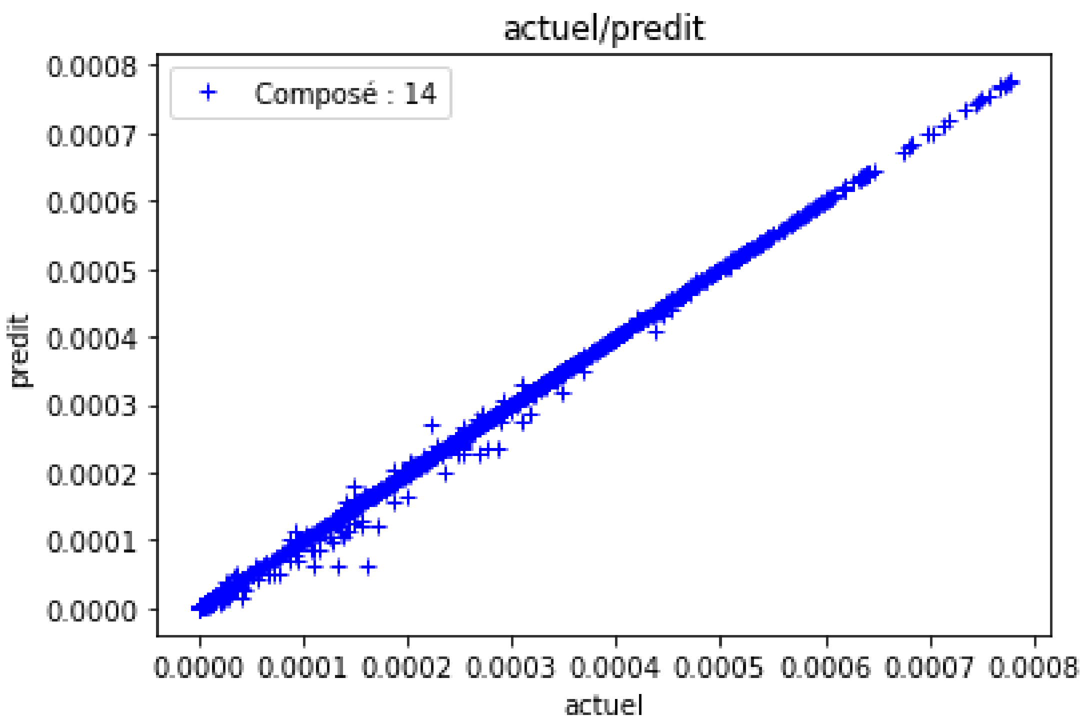


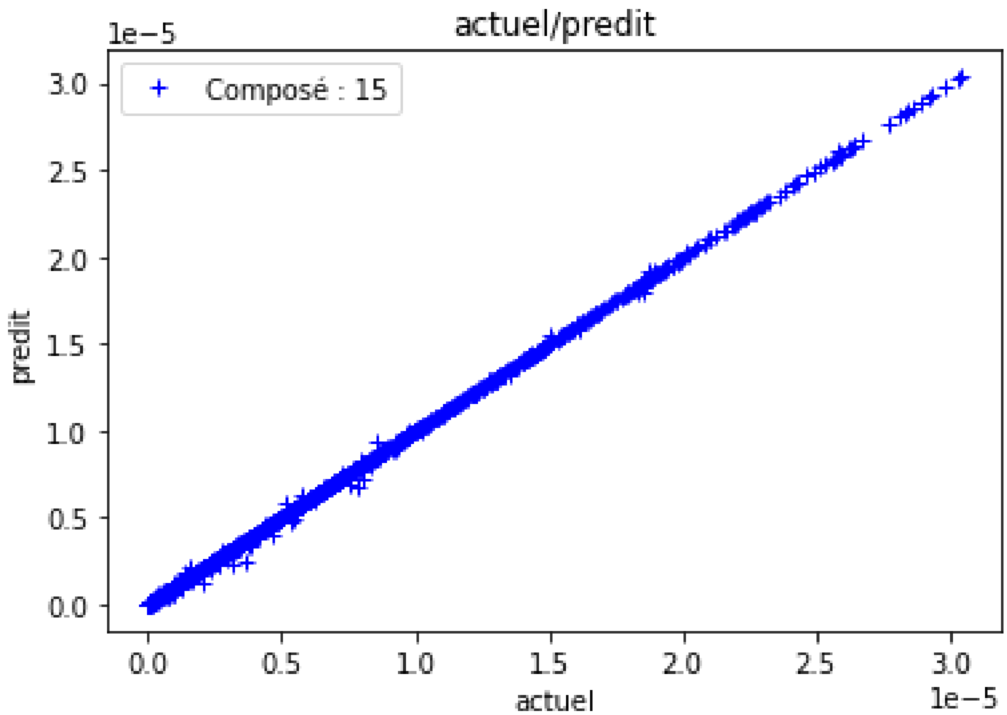
actuel/predit
Composé : 10
0.004
0.003
0.002
0.001
0.000
predit
0.000
0.001
0.002
0.003
0.004
actuel

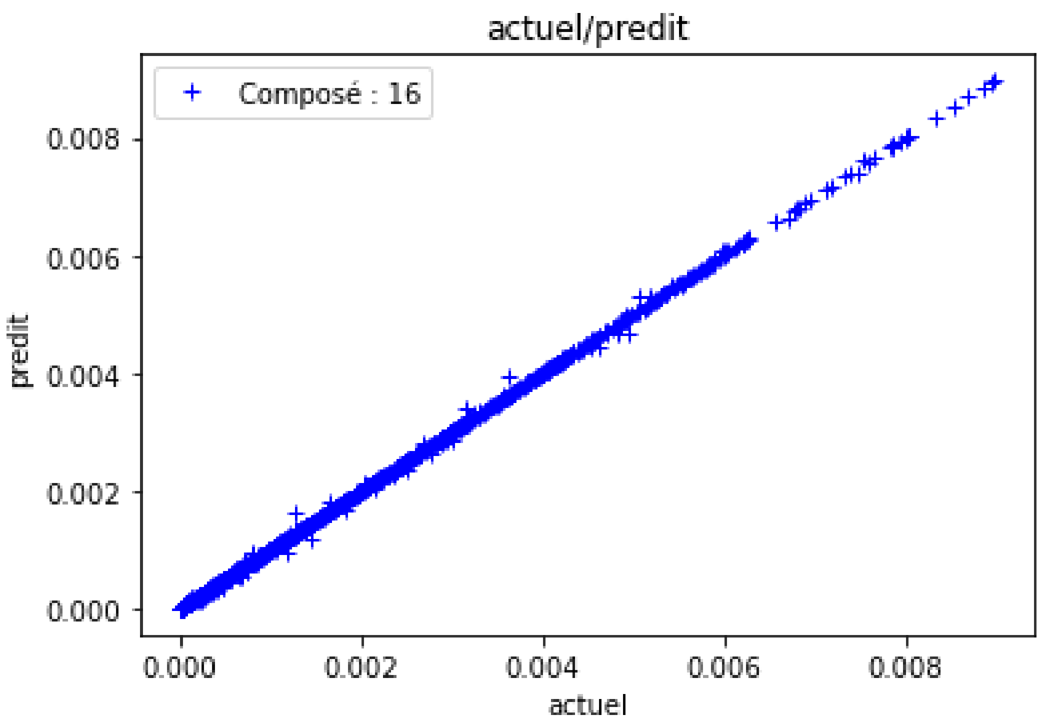

actuel/predit
1e−5
Composé : 11
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
predit
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
actuel
1e−5

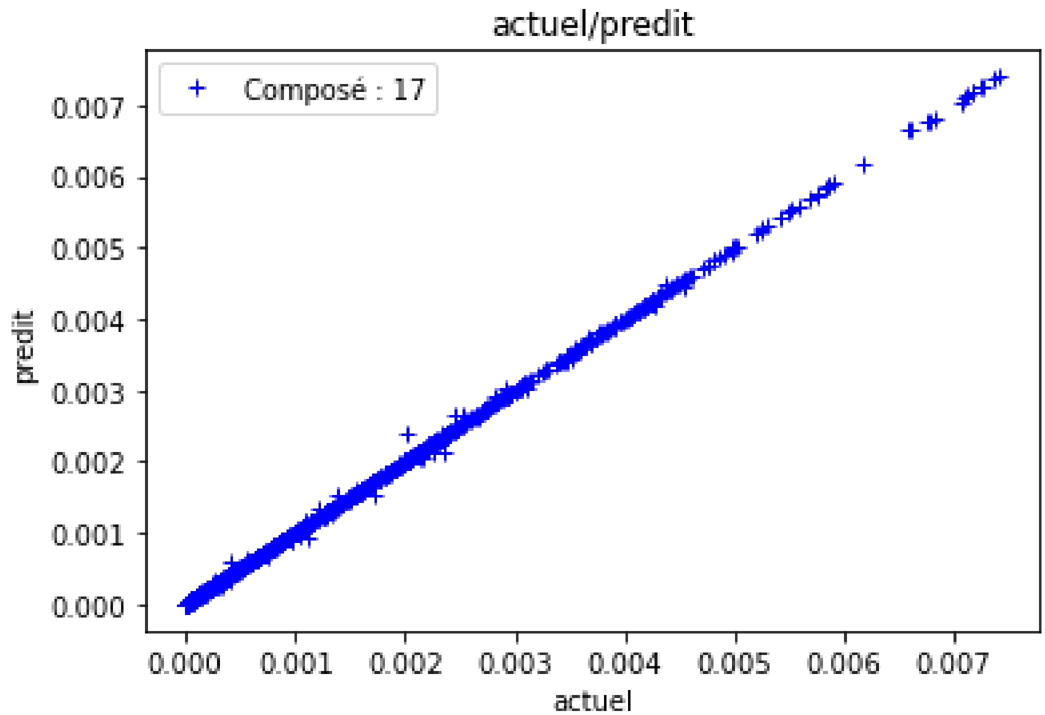

actuel/predit
Composé : 12
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
predit
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
actuel

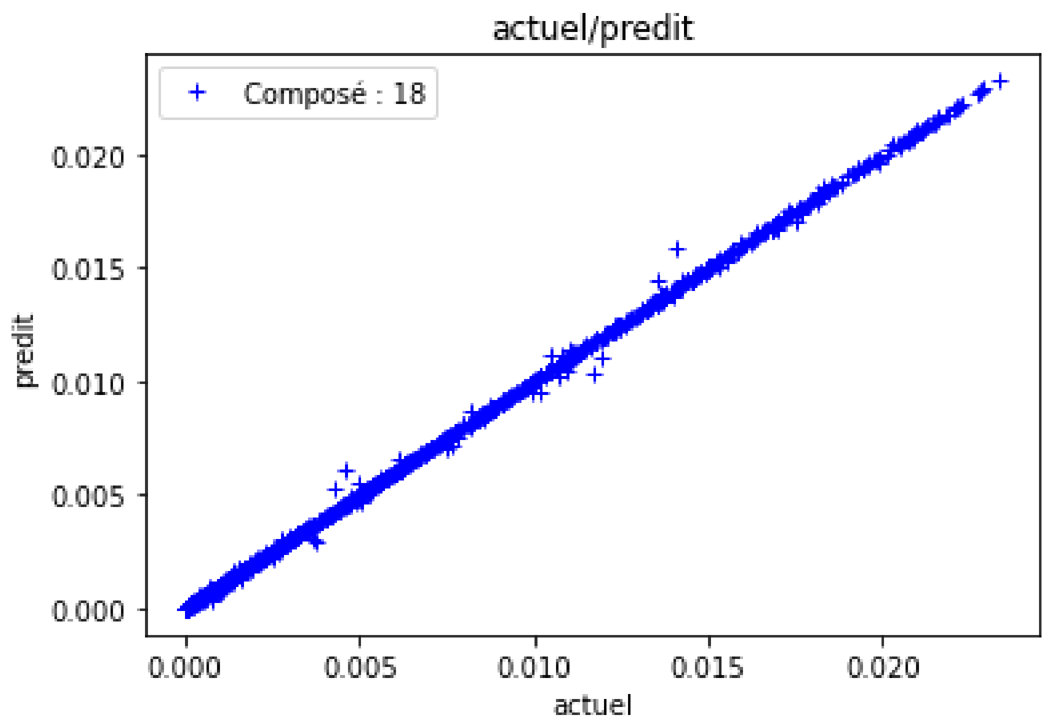

actuel/predit
Composé : 13
predit
actuel
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175

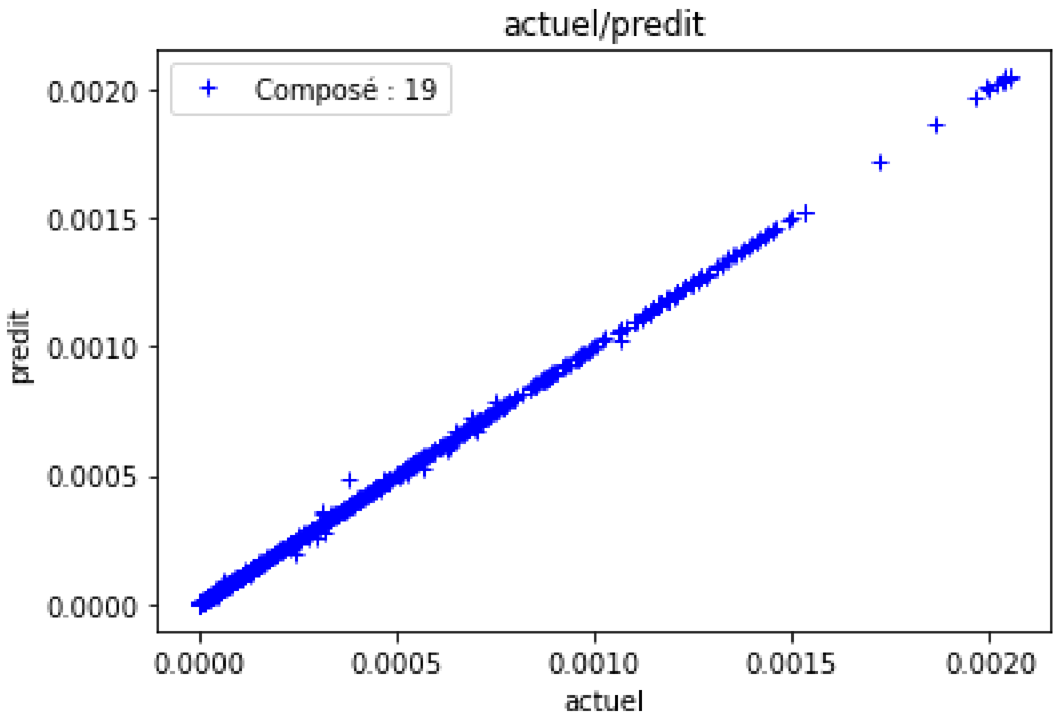

actuel/predit
Composé : 14
predit
actuel
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008

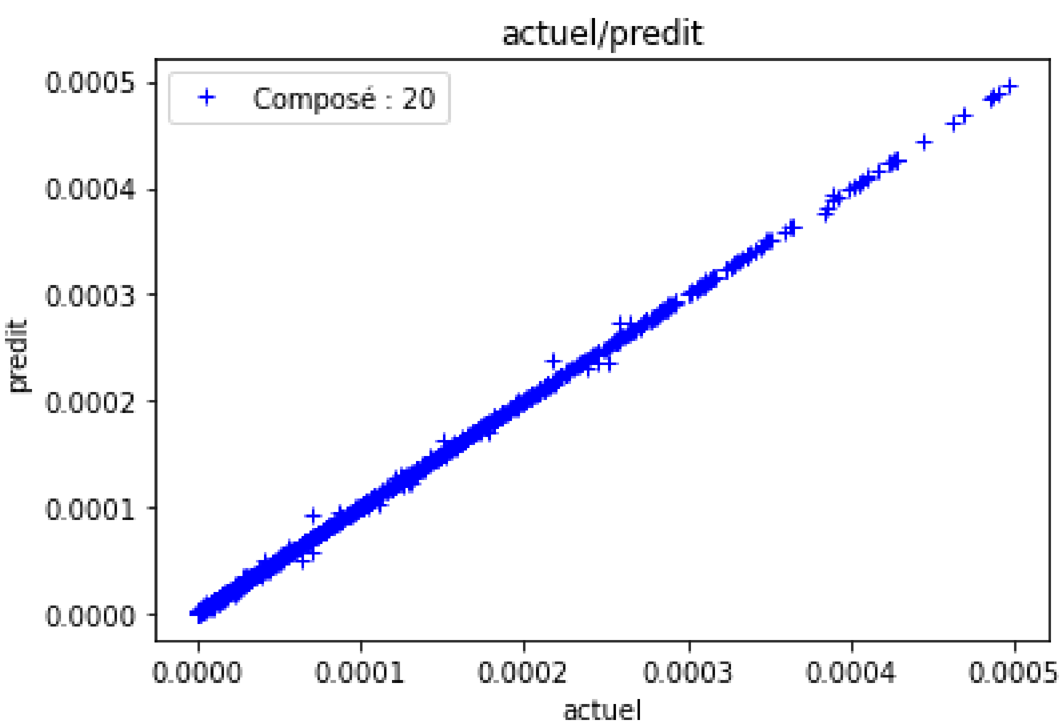

actuel/predit
1e−5
Composé : 15
3.0
2.5
2.0
1.5
1.0
0.5
0.0
predit
0.0
0.5
1.0
1.5
2.0
2.5
3.0
actuel
1e−5

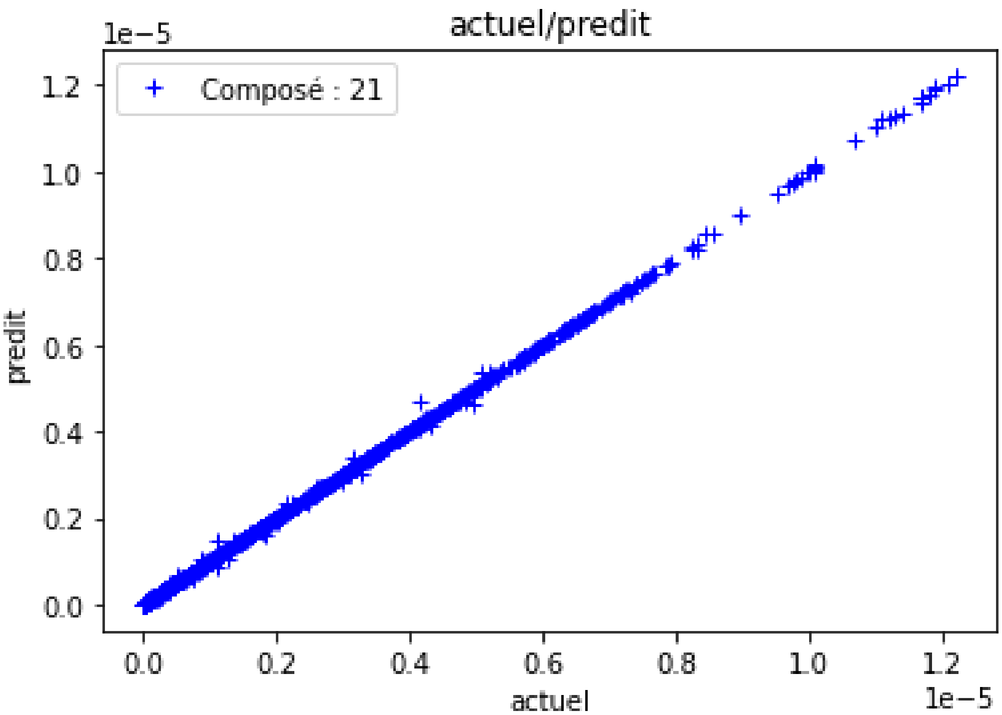

actuel/predit
Composé : 16
0.008
0.006
0.004
0.002
0.000
predit
0.000
0.002
0.004
0.006
0.008
actuel

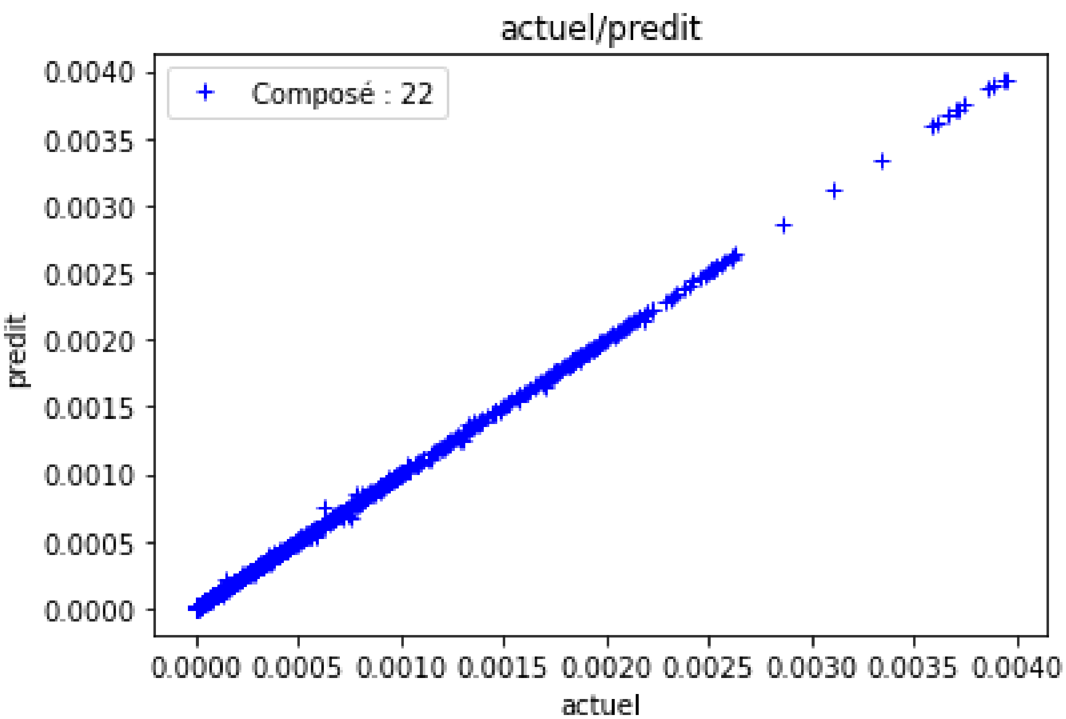

actuel/predit
Composé : 17
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
predit
actuel

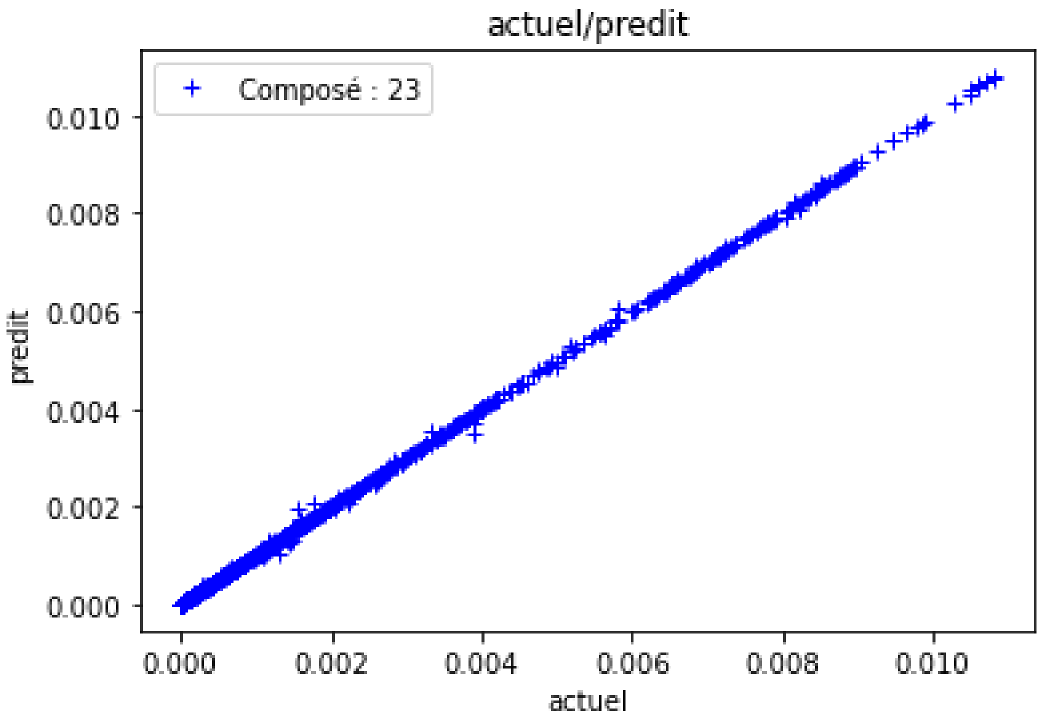

actuel/predit
Composé : 18
0.000
0.005
0.010
0.015
0.020
predit
actuel

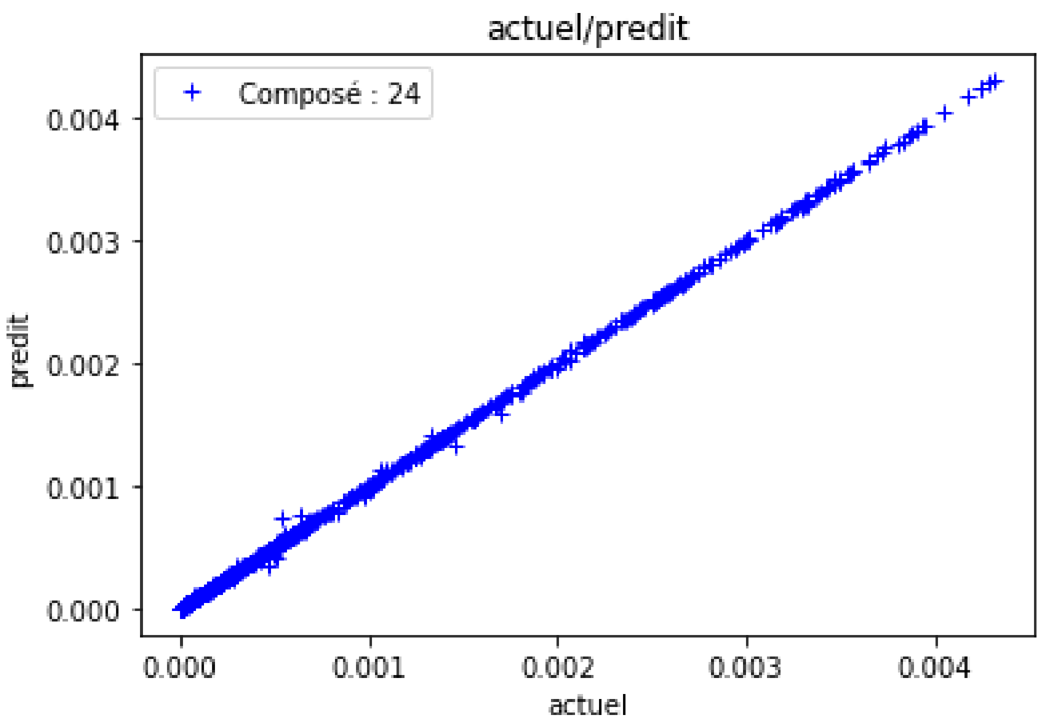

actuel/predit
Composé : 19
0.0020
0.0015
0.0010
0.0005
0.0000
predit
0.0000
0.0005
0.0010
0.0015
0.0020
actuel

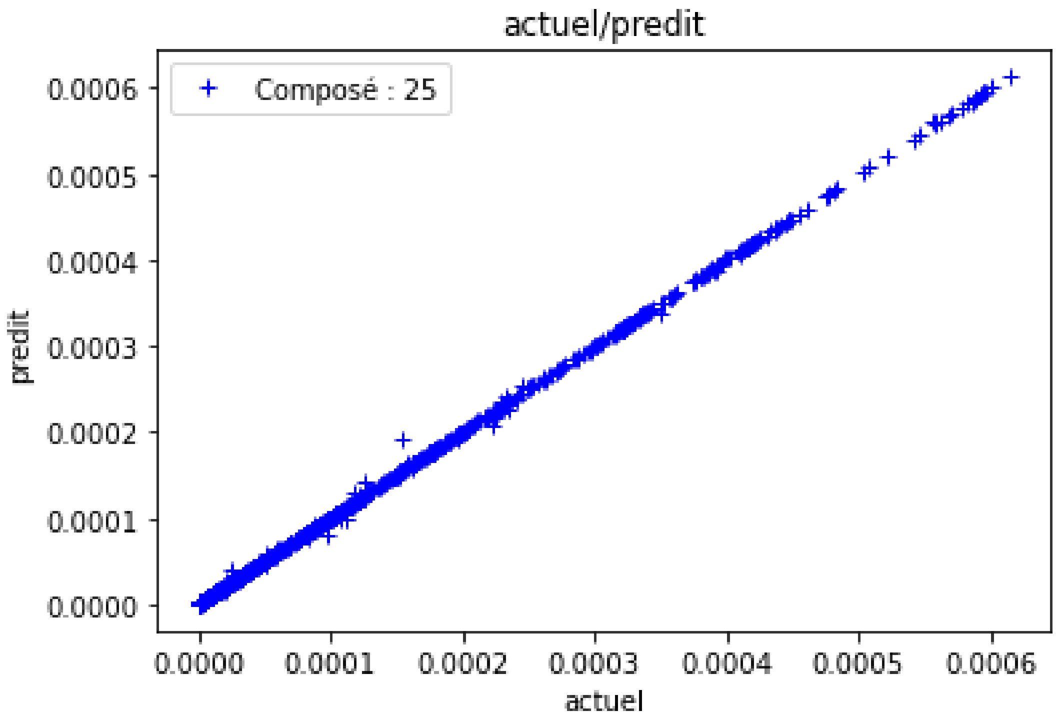

actuel/predit
Composé : 20
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
predit
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
actuel

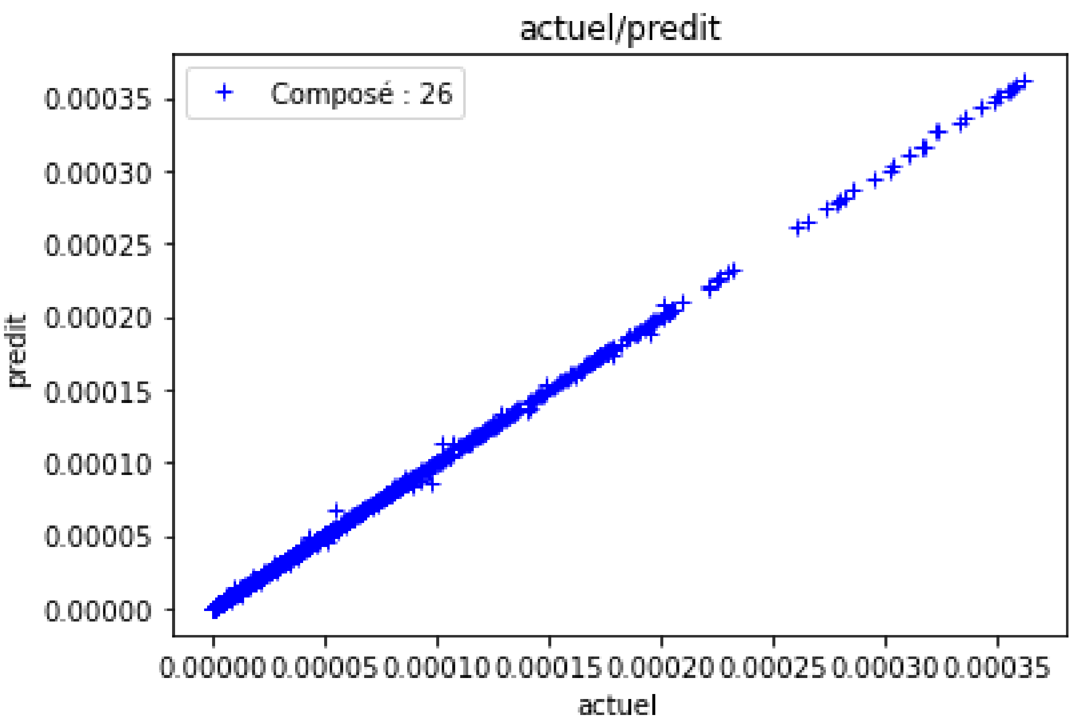

actuel/predit
1e−5
Composé : 21
predit
actuel
1e−5
0.0
0.2
0.4
0.6
0.8
1.0
1.2

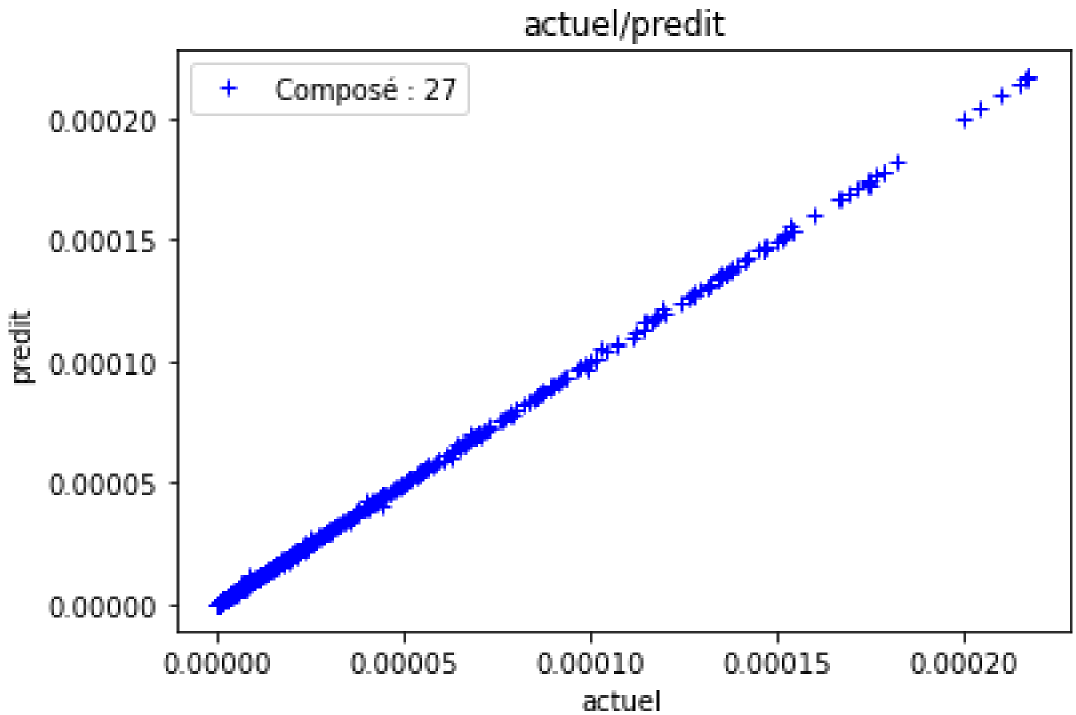

actuel/predit
Composé : 22
predit
actuel
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
0.0040

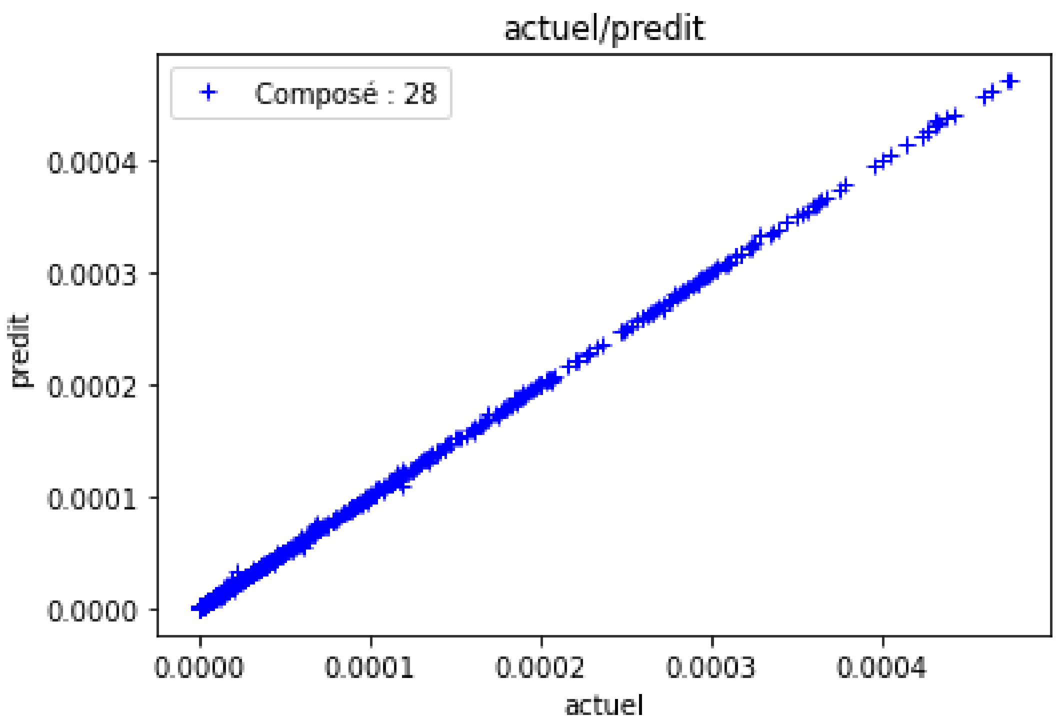

actuel/predit
Composé : 23
0.010
0.008
0.006
0.004
0.002
0.000
predit
0.000
0.002
0.004
0.006
0.008
0.010
actuel

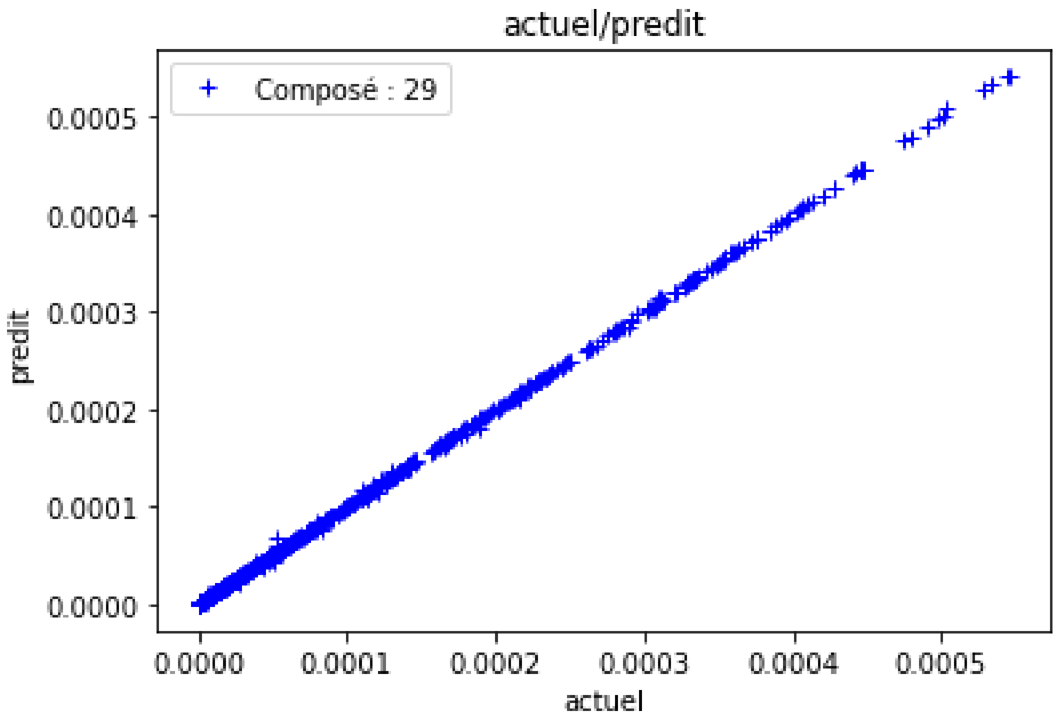

actuel/predit
Composé : 24
0.004
0.003
0.002
0.001
0.000
predit
0.000
0.001
0.002
0.003
0.004
actuel

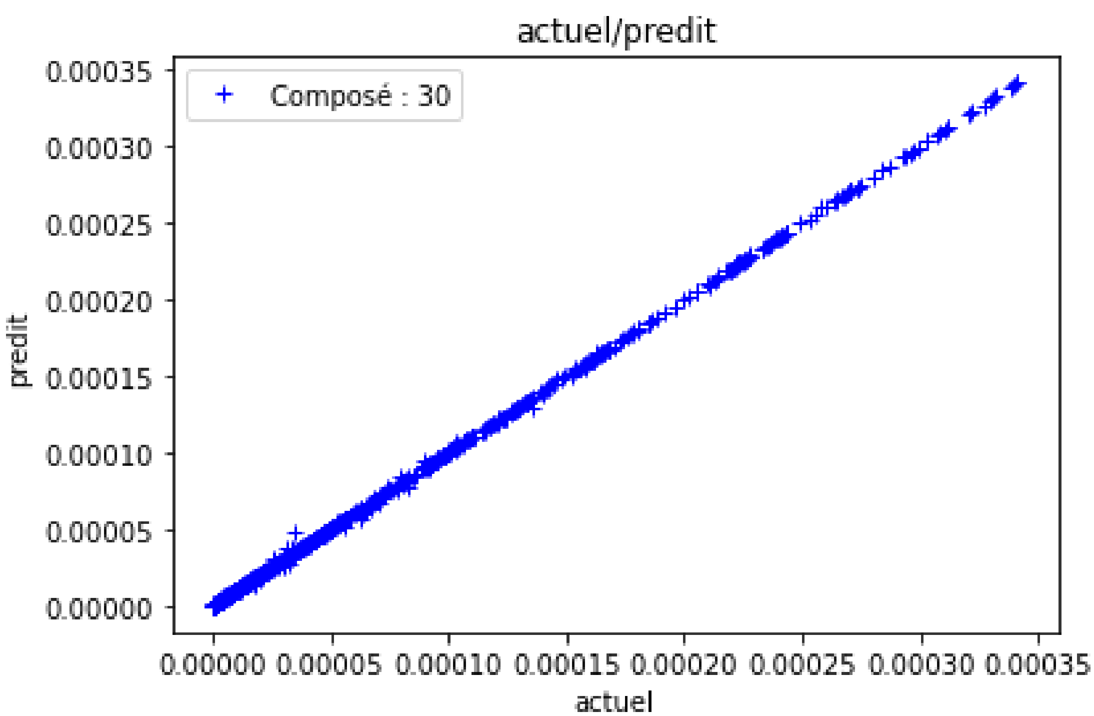

actuel/predit
Composé : 25
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
predit
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
actuel

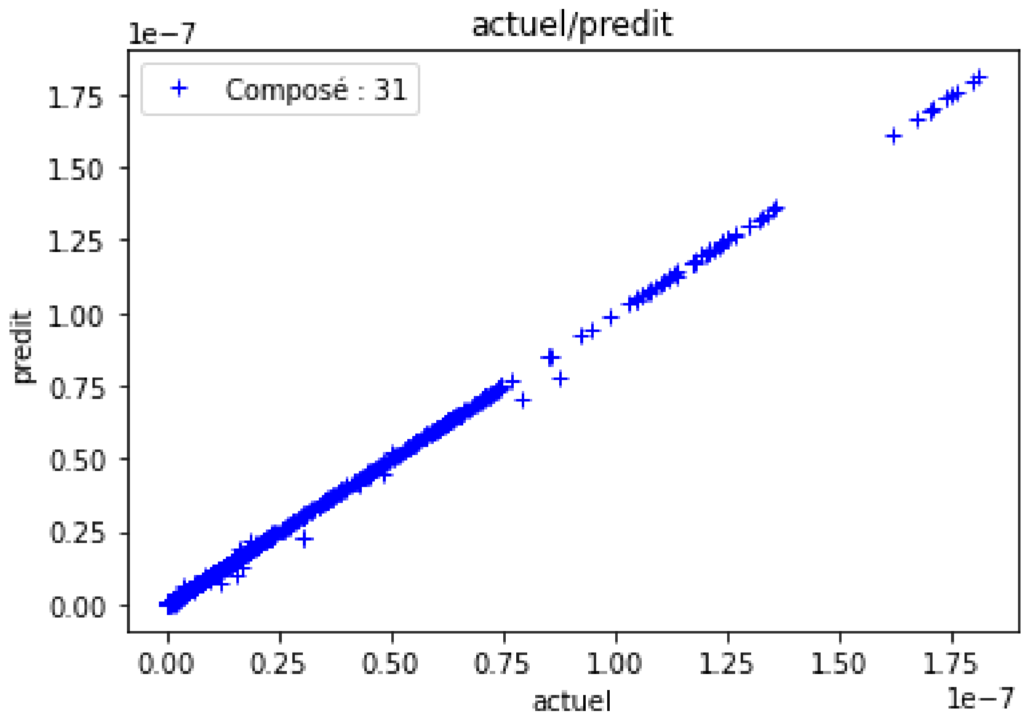

actuel/predit
Composé : 26
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
predit
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
0.00035
actuel


actuel/predit
Composé : 27
0.00020
0.00015
0.00010
0.00005
0.00000
predit
0.00000
0.00005
0.00010
0.00015
0.00020
actuel

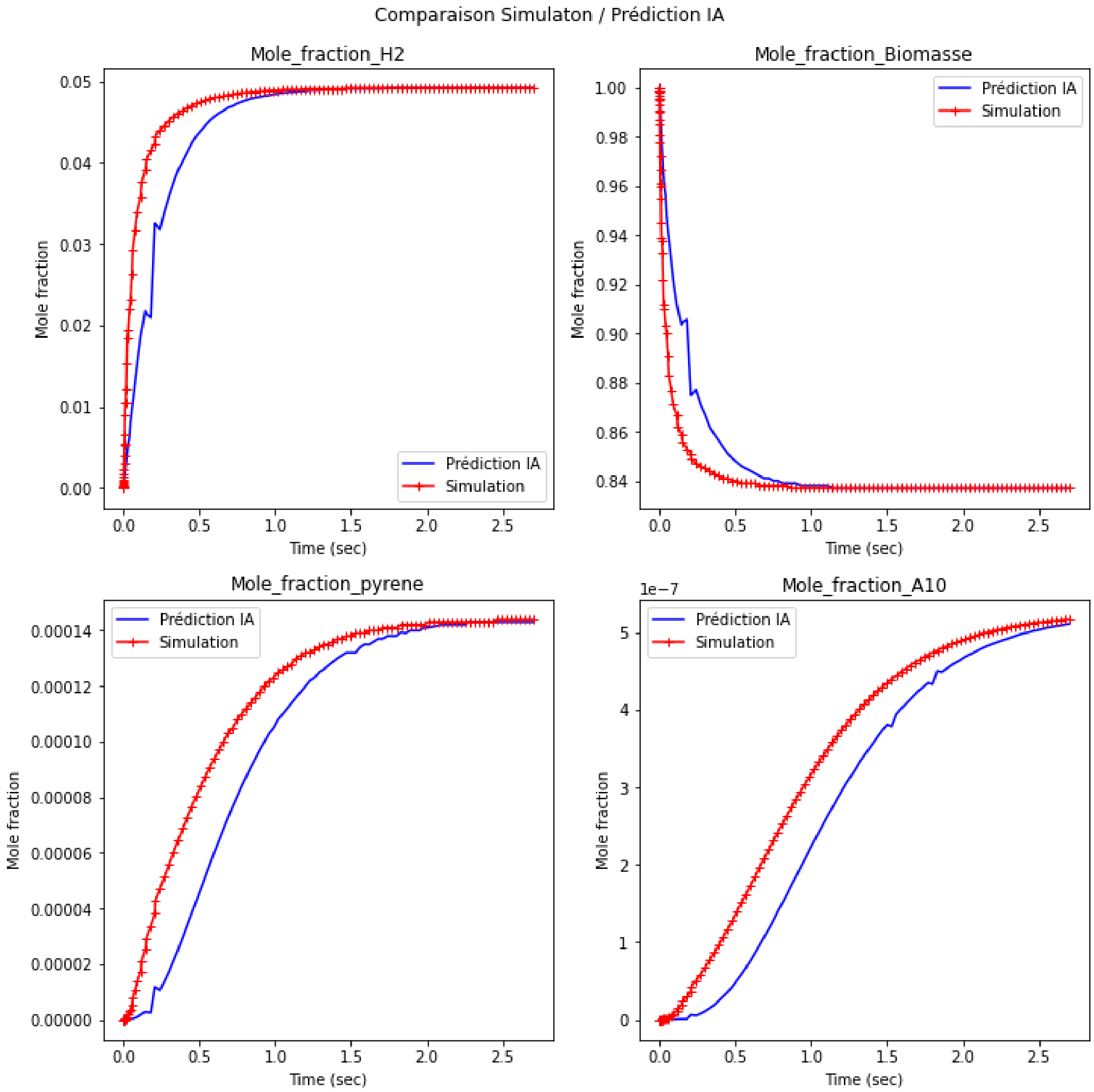

actuel/predit
Composé : 28
0.0004
0.0003
0.0002
0.0001
0.0000
predit
0.0000
0.0001
0.0002
0.0003
0.0004
actuel


actuel/predit
Composé : 29
predit
actuel
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005


actuel/predit
Composé : 30
predit
actuel
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
0.00035



[22]:
```
#
# Courbes de comparaison y_test=f(X_test) vs y_predit=f(X_prédit)
#
from Courbes_Exp_Simu import comparaison_Exp_Sim_4 # Fonction personnelle
#
comparaison_Exp_Sim_4( grid.best_estimator_ , df , [0,2,16,29])
```




```
[23]: #
      # Joblib - exportation du modèle ML sous forme d'un fichier de compilation
      #
      import joblib
      #
      joblib.dump(grid.best_estimator_,'KNN.joblib')
```

```
[23]: ['KNN.joblib']
```

```
[24]: #
      # Essai pour utilisation future
      #
      load_KNN = joblib.load('./KNN.joblib')
```

[25]:

```
#
# Tests sur les métrics
#
print('R2_score')
print('KNN R2 train = ' ,  load_KNN.score(X_train,y_train))
print('KNN R2 test = ' ,  load_KNN.score(X_test,y_test))
```

```
R2_score
KNN R2 train =  0.9999999995148973
KNN R2 test =  0.9997595859006833
```