ESILV - Python for data analysis

CALLAIS Quentin

Dataset assigné

Le dataset qui m'a été assigné se trouve au lien suivant:
http://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation

Ce dataset a été réalisé par le groupe de recherche Milano Chemometrics et QSAR.

On peut retrouver leur site web à ce lien: https://michem.unimib.it/

Aux premiers abords ce dataset semble difficile de compréhension sans un minimum de recherches.

Qu’est-ce que QSAR ?

C’est une relation quantitative structure à activité, c’est le procédé par lequel une structure chimique est corrélée avec un effet bien déterminé comme l’activité biologique ou la réactivité chimique.

Dans notre cas, le dataset se nomme “QSAR biodegradation Data Set” ce qui indique que l’activité biologique ou la réactivité chimique étudiée est la biodégradation d’une structure chimique.

Composition du dataset

Notre dataset comprend 1055 lignes avec 41 + 1 colonnes.

En effet, nous avons 41 variables définissant la structure chimique de la molécule (features) et une variable permettant d'indiquer si cette molécule est classée comme dégradable ou non (label ou target).

dataset

41 molecular descriptors and 1 experimental class:

1) SpMax_L: Leading eigenvalue from Laplace matrix
2) J_Dz(e): Balaban-like index from Barysz matrix weighted by Sanderson electronegativity
3) nHM: Number of heavy atoms
4) F01[N-N]: Frequency of N-N at topological distance 1
5) F04[C-N]: Frequency of C-N at topological distance 4
6) NssssC: Number of atoms of type ssssC
7) nCb-: Number of substituted benzene C(sp2)
8) C%: Percentage of C atoms
9) nCp: Number of terminal primary C(sp3)
10) nO: Number of oxygen atoms
11) F03[C-N]: Frequency of C-N at topological distance 3
12) SdssC: Sum of dssC E-states
13) HyWi_B(m): Hyper-Wiener-like index (log function) from Burden matrix weighted by mass
14) LOC: Lopping centric index
15) SM6_L: Spectral moment of order 6 from Laplace matrix
16) F03[C-O]: Frequency of C - O at topological distance 3
17) Me: Mean atomic Sanderson electronegativity (scaled on Carbon atom)
18) Mi: Mean first ionization potential (scaled on Carbon atom)
19) nN-N: Number of N hydrazines
20) nArNO2: Number of nitro groups (aromatic)
21) nCRX3: Number of CRX3
22) SpPosA_B(p): Normalized spectral positive sum from Burden matrix weighted by polarizability
23) nCIR: Number of circuits
24) B01[C-Br]: Presence/absence of C - Br at topological distance 1
25) B03[C-Cl]: Presence/absence of C - Cl at topological distance 3
26) N-073: Ar2NH / Ar3N / Ar2N-Al / R..N..R
27) SpMax_A: Leading eigenvalue from adjacency matrix (Lovasz-Pelikan index)
28) Psi_i_1d: Intrinsic state pseudoconnectivity index - type 1d
29) B04[C-Br]: Presence/absence of C - Br at topological distance 4
30) SdO: Sum of dO E-states
31) TI2_L: Second Mohar index from Laplace matrix
32) nCrt: Number of ring tertiary C(sp3)
33) C-026: R--CX--R
34) F02[C-N]: Frequency of C - N at topological distance 2
35) nHDon: Number of donor atoms for H-bonds (N and O)
36) SpMax_B(m): Leading eigenvalue from Burden matrix weighted by mass
37) Psi_i_A: Intrinsic state pseudoconnectivity index - type S average
38) nN: Number of Nitrogen atoms
39) SM6_B(m): Spectral moment of order 6 from Burden matrix weighted by mass
40) nArCOOR: Number of esters (aromatic)
41) nX: Number of halogen atoms
42) experimental class: ready biodegradable (RB) and not ready biodegradable (NRB)

Réflexion

Aux vues de mon dataset il m'a paru intéressant de trouver les liens entre la structure de la molécule chimique avec le fait que celle-ci soit biodégradable ou non.

Le grand nombre de variables ne facilite pas la recherche. Il serait intéressant de vérifier la corrélation entre les variable de façon à pouvoir supprimer les variables similaires.

Après avoir réalisé une matrice de corrélation et en prenant une corrélation supérieure à 90% on obtient que 3 variables sont fortements corrélées. Celles-ci ont été supprimé pour la partie machine learning

Réflexion

Notre target étant une variable discrète (0 ou 1) NRB ou RB respectivement, il faut s'orienter sur une classification.

Pour réaliser mon modèle j'ai utilisé deux algorithmes qui me paraissaient pertinent pour une classification.

Le Knn et le random forest.

Après une première étude je me suis rendu compte que le Knn obtenait des performances faibles par rapport au random forest, pour cause, un pré processing inexistant.

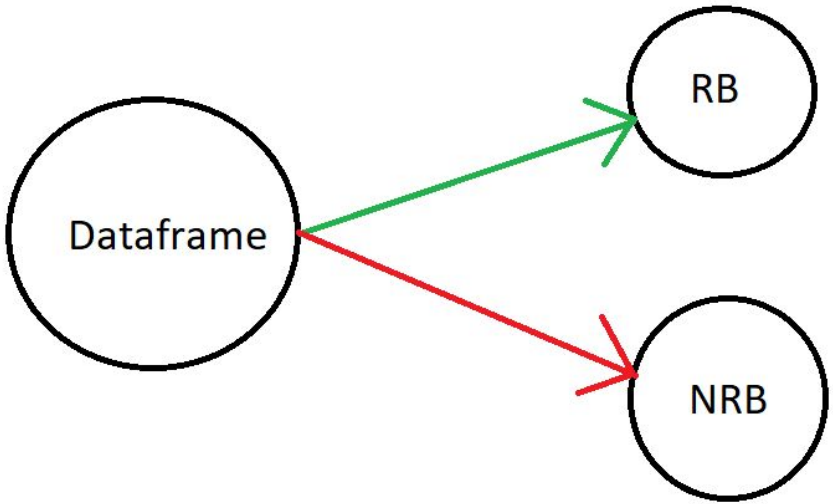
J'ai donc procédé à un pré processing qui m'a permis d'avoir des performances similaires pour les deux algorithmes laissant un léger avantage au random forest.

modèle

Apprentissage supervisé : Classifications

Objectif : Apprendre à notre modèle à prédire la classe expérimentale (RB ou NRB) grâce à des jeux de données dans lesquelles nous lui donnerons les target (apprentissage supervisé).

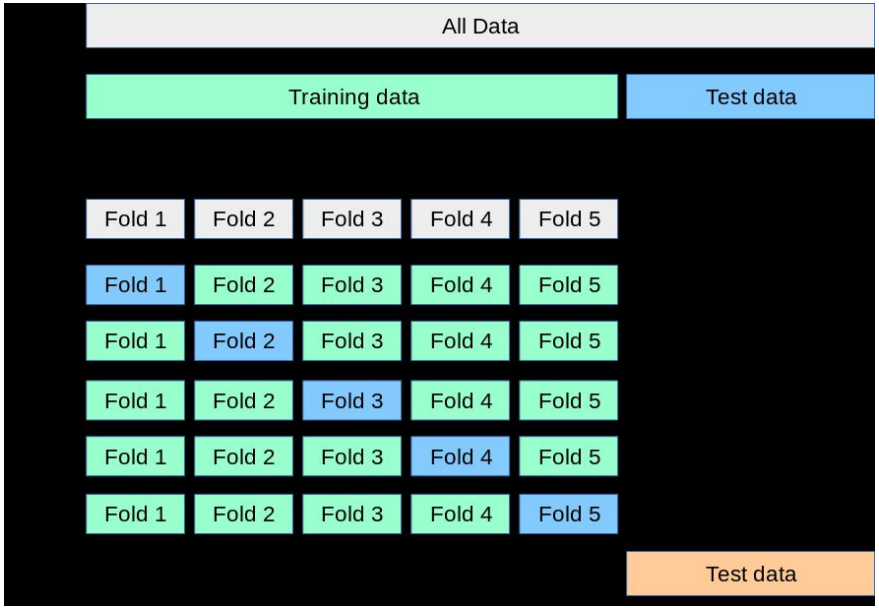
Par la suite l'objectif sera que le modèle soit capable de trouver le bon label en fonction des features que l'on va lui apporter.



Modèle

Dans le but d'obtenir les meilleures performances possibles j'ai utilisé la cross validation qui permettait de parcourir l'ensemble de mon dataframe comme indiqué sur la photo liée.

J'ai pris la valeur arbitraire de 5 pour le découpage du dataframe dans l'ensemble de l'étude.



GridSearch

Dans un but d'effectuer des recherches optimisées j'ai utilisé la GridSearch qui permet de programmer différents hyperparamètres que le modèle va tester et ajuster pour obtenir la meilleure accuracy possible.

Cette méthode permet d'obtenir rapidement la liste des hyperparamètres les plus efficaces, la meilleure accuracy obtenue et donc le meilleur modèle possible.

Problématique rencontrée

Le dataset qu'il a fallu étudier est particulièrement complexe et demande un bagage de connaissance sur la chimie très important si l'on veut pousser ses recherches plus en profondeur.

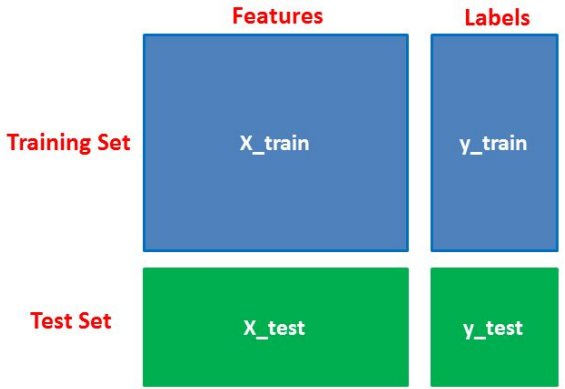
Les résultats finaux de mon modèle ne sont pas optimaux, il aurait été intéressant d'utiliser des techniques de boosting pour améliorer la performance de celui-ci ou tester d'autres classifieurs

Différentes variables créées

Pour répondre à cette étude j'ai créé de multiples variables comme X_train, X_test, Y_train, Y_test qui sont respectivement les features d'entraînement, de tests, et les labels d'entraînements et de tests.

Création également de deux modèles:

ModèleFinalKn et ModèleFinalRd qui correspondent respectivement au modèle construit sur le classifieur Kneighbors et modèle construit sur le classifieur Random Forest.



API

Une api Flask a été créée.

Objectif:

Envoyer les différents paramètres d'une molécule chimique pour obtenir sa classe expérimentale.

```python
@app.route('/')
def hello_world():
    return 'Bienvenue sur mon API, essayez le Endpoints /predict en méthode POST aves les données d\'une mollécule chimique'


@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    X = np.array([v for k, v in data.items()])
    prediction = model.predict([X])
    print(prediction[0])
    if prediction[0] == 0:
        reponse = 'NRB'
    else:
        reponse = 'RB'
    return reponse

if __name__ == "__main__":
    app.run(debug=True)
```

API

Notre API se base sur le modèle le plus efficace que nous avons trouvé, c'est à dire le classifieur random forest.

N'ayant pas développé de front-end j'ai utilisé Postman pour effectuer les requêtes et valider mon API.

API: Cas d'un RB.

POST | localhost:5000/predict

Params | Authorization | Headers (8) | Body

none | form-data | x-www-form-urlencoded

```
{
    "J_Dz(e)":-0.754206,
    "nHM":-0.480821,
    "F01[N-N]":-0.168237,
    "F04[C-N]":-0.410500,
    "NssssC":1.585458,
    "nCb-":-0.749313,
    "C%":-0.283541,
    "nCp":0.775415,
    "nO":-0.451117,
    "F03[C-N]":-0.445567,
    "SdssC":0.262428,
```

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | Visualize | HTML

```
RB
```

Réponse de l'API

Validation avec Postman
J'envoi les données et reçoit une réponse RB (1)

Recherche dans mon jeu de données pour trouver une valeur censée être RB (1)

```
Entrée [195]: for key in X_train:
                  print(key, X_train[key][0])

J_Dz(e) -0.7542063667640078
nHM -0.48082093572289397
F01[N-N] -0.1682374430611899
F04[C-N] -0.41050021369826184
NssssC 1.5854583266308953
nCb- -0.7493128635164362
C% -0.2835407989592972
nCp 0.7754146560710306
nO -0.4511172469117013
F03[C-N] -0.44556718645865995
SdssC 0.2624283649851538
HyWi_B(m) -0.41882766504520946
LOC -0.6207382611339526
F03[C-O] 0.08155937417657 46
Me -0.8287277667298895
Mi 0.18279730970965163
nN-N -0.08746434675422707
nArNO2 -0.21286105660236668
nCRX3 -0.11320502839602563
SpPosA_B(p) -0.16916556022645385
nCIR 0.288681702121762
B01[C-Br] -0.19539022468250006
B03[C-Cl] -0.4211582857738616
N-073 -0.15839191898578664
Psi_i_1d 0.03299357217480883
B04[C-Br] -0.15184943999628206
SdO -0.7325872647566545
TI2_L -0.9015093844558172
nCrt 1.2777136860489728
C-026 -0.5889164535869708
F02[C-N] -0.5574016102144722
nHDon 0.028691205761217687
SpMax_B(m) -0.22143343056315185
Psi_i_A -0.9106848447253153
nN -0.6275005651969832
SM6_B(m) -0.4570926947940005
nArCOOR -0.15895646195030355
nX -0.3255304439305592
```

```
Entrée [194]: Y_train[0]
Out[194]: 1
```

API: Cas d'un NRB.

POST | localhost:5000/predict

Params | Authorization | Headers (8) | Body | Pr

none | form-data | x-www-form-urlencoded | rav

```
27    "B04[C-Br]":-0.15184943999628206,
28    "SdO":-0.7325872647566545,
29    "TI2_L":-0.4438890461033984,
30    "nCrt":-0.20387178484412766,
31    "C-026":0.09490802351194985,
32    "F02[C-N]":0.3453641100752494,
33    "nHDon":0.8349140876514355,
34    "SpMax_B(m)":-0.26180330606915536,
35    "Psi_i_A":-0.7582022692125185,
36    "nN":0.29891445487492024,
37    "SM6_B(m)":-0.4206738172904736,
38    "nArCOOR":-0.15895646195030355,
39    "nX":-0.3255304439305592
40  }
```

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | Visualize | HTML

```
1  NRB
```

Recherche dans mon jeu de données pour trouver une valeur censée être NRB (0)

Validation avec Postman: J'envoi les données et reçoit une réponse NRB (0)

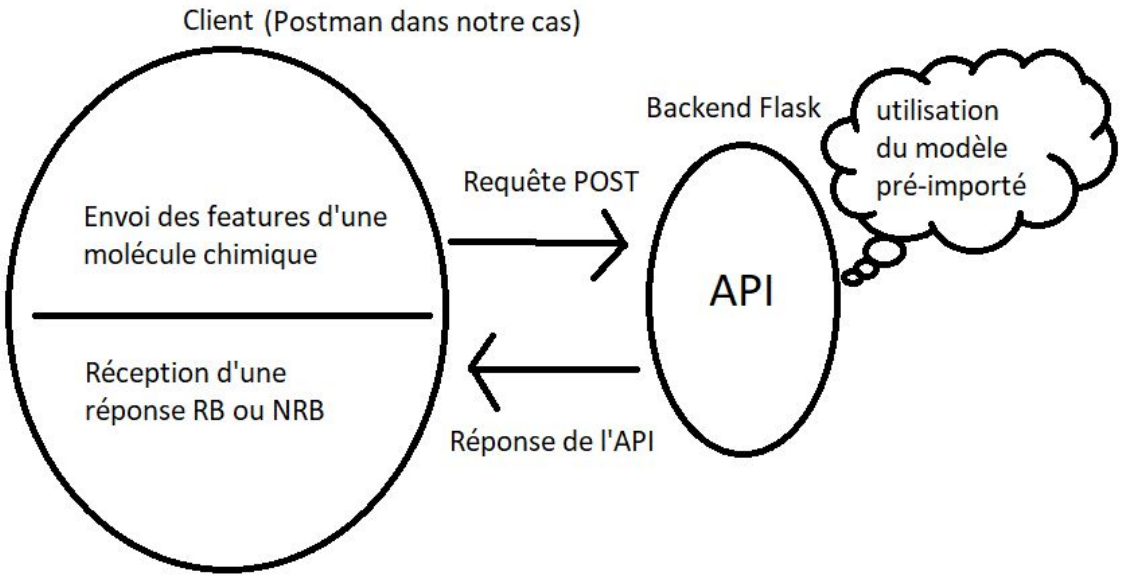
Réponse de l'API

```
Entrée [193]: for key in X_train:
                  print(key, X_train[key][800])

J_Dz(e) -0.1552148938330818
nHM -0.48082093572289397
F01[N-N] -0.1682374430611899
F04[C-N] 0.015147609361559448
NssssC -0.27991692717209804
nCb- 0.1530739135469291
C% 0.21057861062382474
nCp 0.2908523469028476
nO -1.021269929290358
F03[C-N] 0.18149982219684982
SdssC 0.2624283649851538
HyWi_B(m) -0.5069777794352162
LOC -0.18862537200328958
F03[C-O] -0.8027435509790835
Me -0.8509477186192447
Mi -0.022941184396142053
nN-N -0.08746434675422707
nArNO2 -0.21286105660236668
nCRX3 -0.11320502839602563
SpPosA_B(p) 0.31726369165919055
nCIR -0.08832631474678228
B01[C-Br] -0.19539022468250006
B03[C-Cl] -0.4211582857738616
N-073 -0.15839191898578664
Psi_i_1d 0.09355003057533975
B04[C-Br] -0.15184943999628206
SdO -0.7325872647566545
TI2_L -0.4438890461033984
nCrt -0.20387178484412766
C-026 0.09490802351194985
F02[C-N] 0.3453641100752494
nHDon 0.8349140876514355
SpMax_B(m) -0.26180330606915536
Psi_i_A -0.7582022692125185
nN 0.29891445487492024
SM6_B(m) -0.4206738172904736
nArCOOR -0.15895646195030355
nX -0.3255304439305592

Entrée [187]: Y_train[800]
Out[187]: 0
```

Vue générale de l'application



Ressources

http://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation

https://seaborn.pydata.org/introduction.html

https://www.youtube.com/watch?v=mJ9KajSVG0Q

https://www.youtube.com/watch?v=P6kSc3qVph0&feature=youtu.be

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://mrmint.fr/data-preprocessing-feature-scaling-python

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

https://scikit-learn.org/stable/modules/preprocessing.html

Notebook du cours de python for data Analysis