Live Coding

Python et Machine Learning

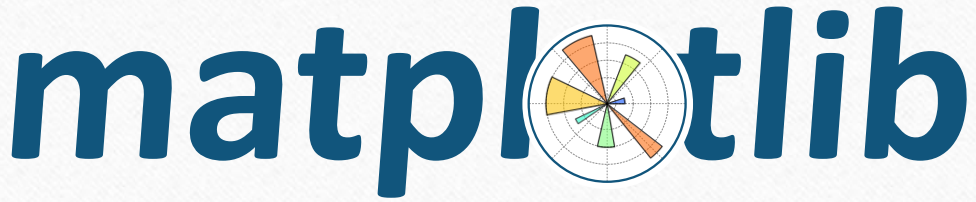
3 mises en situations

- 1. Prédiction via une régression linéaire simple
- 2. Classification d'une espèce de plante grâce au random forest
- 3. Evaluation de l'hyperparamètre k du kNN

Librairies utilisées

https://pandas.pydata.org/pandas-docs/stable/

https://numpy.org/doc/



https://matplotlib.org/

https://scikit-learn.org/stable/

Mise en situation n°1

Régression linéaire simple

1. Régression linéaire simple
A. Importation des données

```python
#Importation de la librairie Pandas
import pandas as pd
df = pd.read_csv("../data/univariate_linear_regression_dataset.csv")
print(df)
```

```
      6.1101    17.592
0     5.5277   9.13020
1     8.5186  13.66200
2     7.0032  11.85400
3     5.8598   6.82330
4     8.3829  11.88600
..       ...       ...
91    5.8707   7.20290
92    5.3054   1.98690
93    8.2934   0.14454
94   13.3940   9.05510
95    5.4369   0.61705

[96 rows x 2 columns]
```

```python
#Sélection de la première colonne du dataset (col0)
#iloc = permet de récup une donnée par sa position

x = df.iloc[0:len(df), 0]

#Sélection de la deuxième colonne du dataset (col1)

y = df.iloc[0:len(df), 1]
```

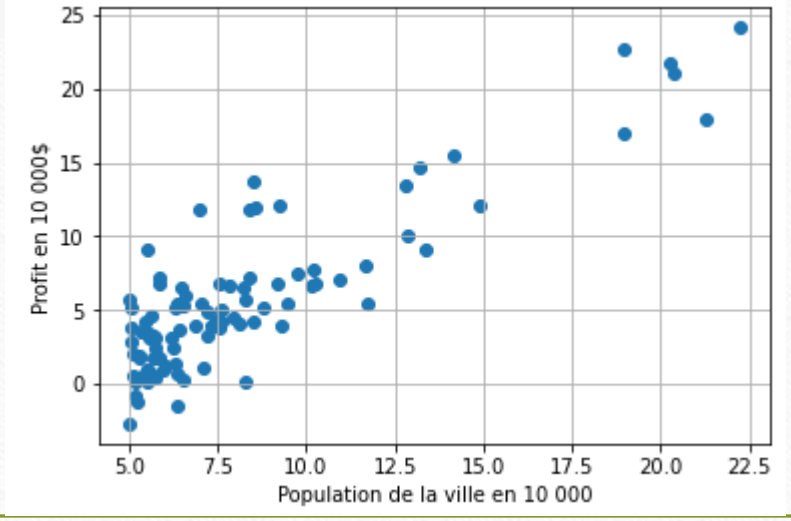
1. Régression linéaire simple
B. Visualisation + fonction prédictive

```python
import matplotlib.pyplot as plt

axes = plt.axes()
axes.grid()
plt.scatter(x,y)
plt.xlabel("Population de la ville en 10 000")
plt.ylabel("Profit en 10 000$")
plt.show()
```

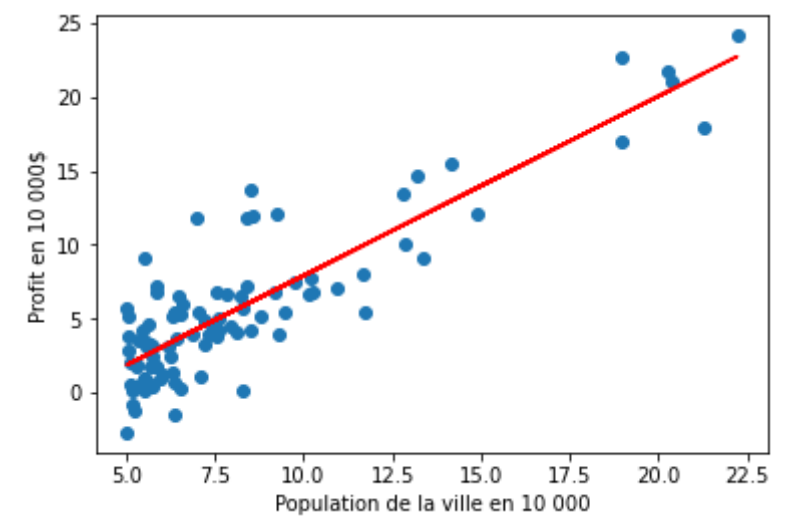
```python
from scipy import stats
slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)

#définition de la fonction prédictive
def predict(x):
    return slope * x + intercept
```



1. Régression linéaire simple
C. Application de la régression + prédiction

```
plt.scatter(x,y)
plt.xlabel("Population de la ville en 10 000")
plt.ylabel("Profit en 10 000$")
plt.plot(x, predict(x), c='r')
plt.show()
```




```
print('Prédiction pour 17.5 : ', predict(17.5))
print('Prédiction pour 20 : ', predict(20))
print('Prédiction pour 22.5 : ', predict(22.5))
print('Prédiction pour 13 : ', predict(13))
```

```
Prédiction pour 17.5 :  17.025572937972186
Prédiction pour 20 :  20.05944107274308
Prédiction pour 22.5 :  23.09330920751398
Prédiction pour 13 :  11.56461029538457
```

Mise en situation n°2

Random forest

2. Random forest

A. Importation des données + visualisation

```
from sklearn import datasets
iris = datasets.load_iris()

#Transformation de l'array en dataframe
import pandas as pd
data = pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target,
})
data.head()
```

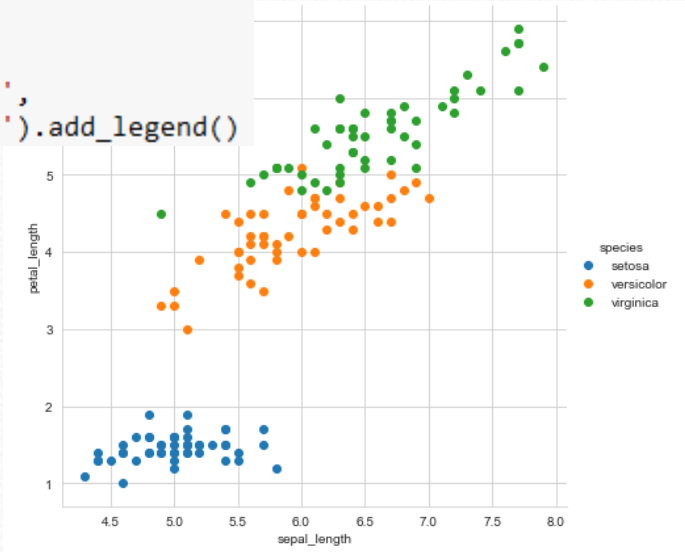
	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')

sns.set_style("whitegrid")

sns.FacetGrid(iris, hue ="species",
              height = 6).map(plt.scatter,
                              'sepal_length',
                              'petal_length').add_legend()
```



2. Random forest
B. Entrainement de l'algorithme + prédiction

```python
from sklearn.model_selection import train_test_split

X = data[['sepal length', 'sepal width', 'petal length', 'petal width']]
y = data['species']

# Split le dataset en training et test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

from sklearn.ensemble import RandomForestClassifier

# Création d'un classifieur
clf = RandomForestClassifier(n_estimators=100)

# Entrainement du modèle
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

from sklearn import metrics
print("Accuracy : ", metrics.accuracy_score(y_test, y_pred))
```

Prédiction pour une plante

Prenons trois exemples :

- plante 1 : 3cm longueur sépales, 5cm largeur sépales, 4cm longueur pétales et 2cm largeur pétales
- plante 2 : 10cm longueur sépales, 5cm largeur sépales, 14cm longueur pétales et 3cm largeur pétales
- plante 3 : 1cm longueur sépales, 2cm largeur sépales, 1cm longueur pétales et 2cm largeur pétales

```
clf.predict([[3,5,4,2]])
```

```
array([2])
```

```
clf.predict([[10,5,14,3]])
```

```
array([2])
```

```
clf.predict([[1,2,1,2]])
```

```
array([0])
```

Mise en situation n°3

Evaluation de l'hyperparamètre k du kNN

3. Evaluation de l'hyperparamètre k
A. Importation des données + standardisation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('../data/winequality-white.csv', sep=";")

# Je sépare les vins médiocres des bons vins => le but est de transformer le
# problème en un problème de classification!
y_class = np.where(y<6, 0, 1)

# Séparation des données en training set et test set
from sklearn import model_selection
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y_class,test_size=0.3)
from sklearn import preprocessing

# Je standardise les données d'entrainement
std_scale = preprocessing.StandardScaler().fit(X_train)
X_train_std = std_scale.transform(X_train)

# J'applique la même transformation sur les données de test
X_test_std = std_scale.transform(X_test)
```

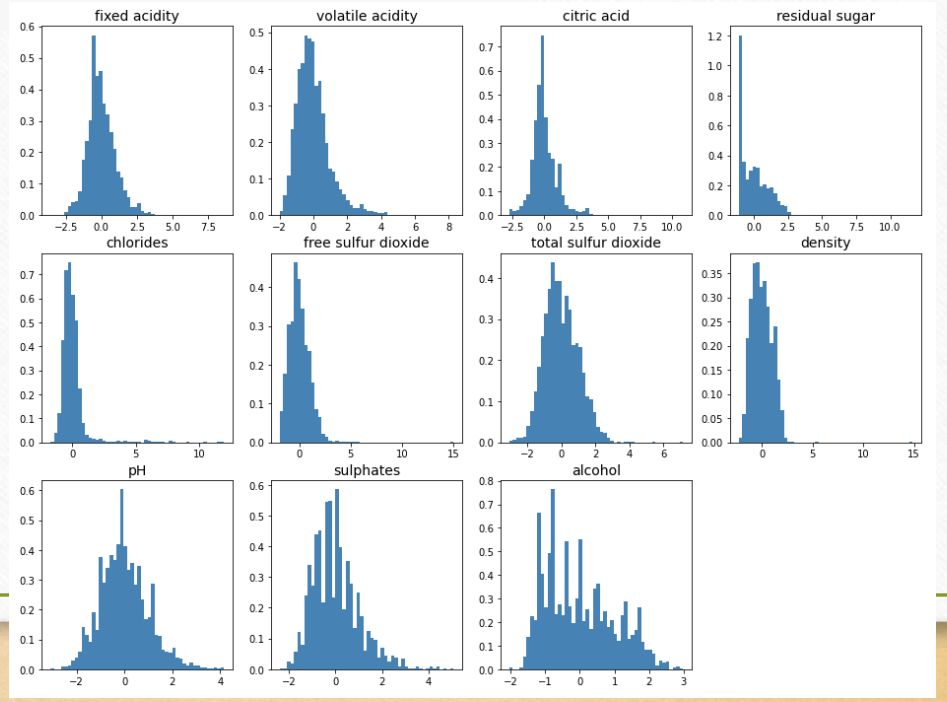
```
# Récupération des données et stockage dans une variable

X = data[data.columns[:-1]].values
y = data['quality'].values
```

3. Evaluation de l'hyperparamètre k
B. Visualisation

```
fig = plt.figure(figsize=(16, 12))

for i in range(X_train_std.shape[1]):
    ax = fig.add_subplot(3,4, (i+1))
    h = ax.hist(X_train_std[:, i], bins=50, color = 'steelblue', density=True, edgecolor='none')
    ax.set_title(data.columns[i], fontsize=14)
```



3. Evaluation de l'hyperparamètre k
C. Validation croisée

```python
from sklearn import neighbors, metrics

# Fixe des valeurs d'hyperparamètres
param_grid = {'n_neighbors':[3,5,7,9,11,13,15]}

# Choix du score à optimiser => proportion de prédictions correctes
score = 'accuracy'

# Classifieur kNN
clf = model_selection.GridSearchCV(
    neighbors.KNeighborsClassifier(),
    param_grid,
    cv = 5,
    scoring = score)

# Optimisation du classifieur sur le training set
clf.fit(X_train_std, y_train)

# Affichage du meilleur hyperparamètre
print("Meilleur hyperparamètre :")
print(clf.best_params_)
```

```python
# Afficher les performances
print("Résultats de la validation croisée :")
for mean, std, params in zip(
        clf.cv_results_['mean_test_score'], # score moyen
        clf.cv_results_['std_test_score'],  # écart-type du score
        clf.cv_results_['params']           # valeur de l'hyperparamètre
    ):

    print("{} = {:.3f} (+/-{:.03f}) for {}".format(
        score,
        mean,
        std*2,
        params
    ) )
```

```
Meilleur hyperparamètre :
{'n_neighbors': 13}
Résultats de la validation croisée :
accuracy = 0.761 (+/-0.018) for {'n_neighbors': 3}
accuracy = 0.762 (+/-0.029) for {'n_neighbors': 5}
accuracy = 0.763 (+/-0.022) for {'n_neighbors': 7}
accuracy = 0.765 (+/-0.027) for {'n_neighbors': 9}
accuracy = 0.761 (+/-0.014) for {'n_neighbors': 11}
accuracy = 0.766 (+/-0.019) for {'n_neighbors': 13}
accuracy = 0.763 (+/-0.023) for {'n_neighbors': 15}
```

Ressources

- https://github.com/CalcagnoLoic/veille-becode/tree/main/Live%20coding
- https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pour-la-data-science
- https://openclassrooms.com/fr/courses/4297211-evaluez-les-performances-dun-modele-de-machine-learning
- https://openclassrooms.com/fr/courses/4444646-entrainez-un-modele-predictif-lineaire
- https://openclassrooms.com/fr/courses/4470521-modelisez-vos-donnees-avec-les-methodes-ensemblistes
- https://openclassrooms.com/fr/courses/4470406-utilisez-des-modeles-supervises-non-lineaires

Merci de votre attention

☺