

Que ce soit de la régression ou de la classification, vous serez amené à utiliser le même code :

- 1) Sélectionnez un modèle et précisez ses hyperparamètres. Les modèles sont disponibles dans la documentation de Scikit Learn, par exemple « `LinearRegression()` » pour la régression linéaire ; « `SVC` » pour Support Vector Machine ; « `LogisticRegression()` » pour la régression logistique ; « `RandomForestClassifier()` ».
- 2) Entraînez le modèle : `model.fit(X,y)` ;
- 3) Évaluez le modèle : `model.score(X,y)` ;
- 4) Utilisez le modèle (si on est satisfait) : `model.predict(X)`.

Exercice 1 : Régression linéaire

- 1) Importez les modules nécessaires à la réalisation de cet exercice à savoir :

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RidgeRegression
```

- 2) Construisez un jeu de données formé de m échantillons. Ces échantillons sont générés de manière aléatoire comme suit :

```
np.random.seed(0)
m=100
X = np.linspace(0,10,m).reshape(m,1)
Y = X + np.random.random_sample((m,1))
```

Que représentent X et Y ?

- 3) Affichez le nuage de points en utilisant : `plt.scatter(X,y)`. Commentez le graphique.
- 4) On souhaite représenter ce nuage de points par un modèle de régression linéaire :
`model = LinearRegression()` :
 - a) Entraînez ce modèle (`model.fit(X,y)`).
 - b) Évaluez ce modèle (`model.score(X,y)`).
 - c) Superposez sur le nuage de points, le modèle prédit en utilisant `plt.plot(model.predict, c='r')`.
 - d) Commentez les résultats.

- 5) Construisez un nouveau jeu de données formé de m échantillons. Ces échantillons sont générés de manière aléatoire comme suit :

```
X = np.linspace(0,10,m).reshape(m,1)
Y = X**2 + np.random.random(m,1)
```

- 6) Affichez le nuage de points en utilisant : `plt.scatter(X,y)` pour $m=100$. Commentez le graphique.

- 7) On souhaite représenter ce nuage de points par un modèle de régression linéaire :
`model = LinearRegression()` :
 - a) Identifiez les caractéristiques du modèle.
 - b) Entraînez ce modèle (`model.fit(X,y)`).
 - c) Évaluez ce modèle (`model.score(X,y)`). L'apprentissage est-il réussi ?
 - d) Superposez sur le nuage de points, le modèle prédit (`plt.plot(model.predict, c='r')`) pour différentes valeurs de m (100, 1000, 10000).
 - e) Commentez les résultats.

- 8) Pour remédier au problème de la question précédente, choisissez un autre modèle (voir Fig. 1) par exemple le modèle SVR :

`model SVR(C=100)` :

- a) Identifiez les caractéristiques du modèle.
- b) Entraînez ce modèle (`model.fit(X,y)`).
- c) Évaluez ce modèle (`model.score(X,y)`). L'apprentissage est-il réussi ?
- f) Tracez le modèle prédit (`plt.plot(model.predict, c='r')`) pour différentes valeurs de m (100, 1000, 10000).
- d) Commentez les résultats.

Exercice 2 : Classification KNN

L'objectif de cet exercice est de travailler avec des données réelles du Titanic.

1) Importez les librairies ci-dessous :

```
Import pandas as pd
Import seaborn as sns
```

2) Téléchargez le jeu de données du Titanic en utilisant seaborn

```
Titanic = sns.load_dataset('titanic')
Titanic.shape
Titanic.head()
```

3) Analysez ce jeu de données en vous appuyant sur Titanic.head().

4) On souhaite nettoyer ce jeu de données pour supprimer les données manquantes et récupérer uniquement les données sur lesquelles nous souhaitons travailler.

a) Sélectionnez uniquement les colonnes ['survived', 'pclass', 'sex', 'age'] :

```
Titanic = ['survived', 'pclass', 'sex', 'age']
```

b) Supprimez les données manquantes :

```
Titanic.dropna(axis=0,inplace=True)
```

c) Numérisez le jeu de données. Remplacez par exemple 'male' par zéro et 'female' par un :

```
Titanic['sex'].replace(['male', 'female'], [0,1], inplace = True)
Titanic.head()
```

c) Vérifiez que jeu de données est bien numérisé. Si non procédez aux diverses numérisations.

4) Le modèle de classification choisi permettra de savoir si un passager du Titanic survivra au naufrage. Le modèle doit donc être entraîné.

a) On a un tableau pandas, il va falloir qu'on le divise en deux tableaux X et Y

```
Y = Titanic['survived']
X = Titanic.drop('survived', axis=1)
```

Expliquez l'intérêt de cette écriture. Que représentent X et Y.

b) En vous appuyant sur Fig. 1, sélectionnez le modèle de classification KNeighborsClassifier en faisant appel à :

```
from sklearn.neighbors import KNeighborsClassifier
```

c) Entraînez le modèle (model.fit(X,Y)).

d) Évaluez le score (model.score(X,y)). Commentez le résultat.

5) Étant donné que le modèle est entraîné, il s'agit maintenant de prédire la survie d'un passager.

a) Construisez une fonction « survie » qui prend comme paramètre « pclass », le « sex » et « âge », comme suit :

```
def survie(model, pclass=3, sex=0, age=26)
x = np.array([pclass, sex, age]).reshape(1,3)
print(model.predict(x))
```

a) Avec les paramètres pclass=3, sex=0, age=26, le passager a-t-il survécu (survie(model)) ?

b) Calculez la probabilité pour appartenir à la classe 0 et à la classe 1 en ajoutant à la fonction :

```
print(model.predict_proba(x))
```

6) Faites varier le nombre de voisins de 1 à 10 du modèle KNeighborsClassifier.

a) Retenez le nombre de voisin qui donne le meilleur score.

b) Qu'en est-il de la prédiction de survie ?