

Manipulation pratique

Regression lineaire

**Manipulation 1**: Déroulement de l'algorithme de régression linéaire

**Préparé par :** Hafed Benteftifa  
  
© Hafed Benteftifa et Nesrine Zemirli 2015-2016

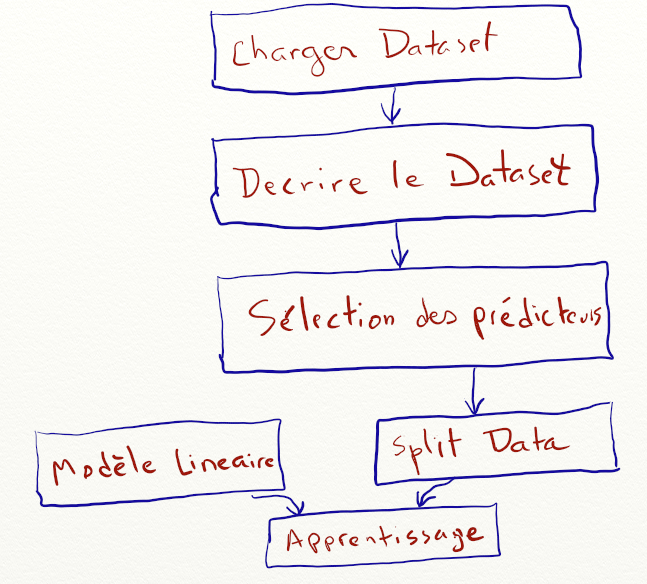
Ce document ne peut être utilisé dans le cadre d’une formation, publication papier, site internet ou tout support sans mon accord express.

Aucune reproduction, même partielle, ne peut être faite de ce document et de l'ensemble de son contenu : textes, images, etc. sans mon autorisation express. Pour toutes informations, communiquer avec moi sur [info@degenio.com](mailto:info@degenio.com).

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Changement** |
| 10 juin 2016 | 1.0 | Version initiale |
|  |  |  |
|  |  |  |

**Démarche**

Le processus d'apprentissage que l'on va développer avec le modèle de régression linéaire est décrit dans le digramme suivant. Il faut noter que ce processus est assez générique pour être repris pour les autres modèles de régression et de classification.

****

Les modules python utilisés dans cette pratique sont scikit-learn, numpy et matplotlib.

**Manipulation 1**: Utilisation d’un algorithme de régression linéaire

**Objectif**

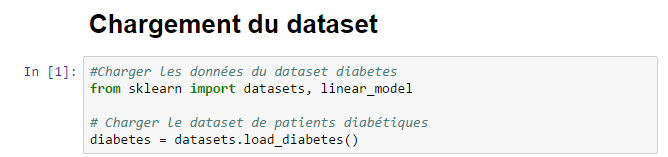
Dérouler un processus de prédiction en utilisant un modèle de régression linéaire.

**Préliminaire**

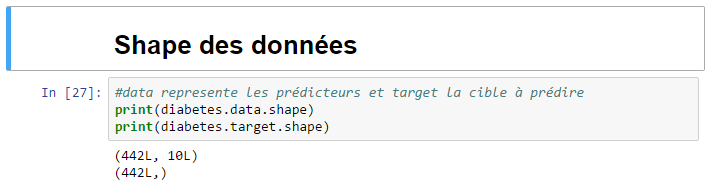
* Anaconda ou pycharm est disponible.

**Approche**

1. Procéder au chargement du dataset de patients diabétiques comme suit:

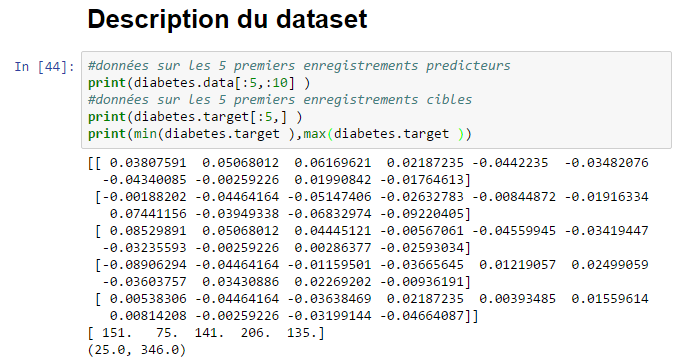


2. Le dataset diabetes consiste en les informations suivantes:

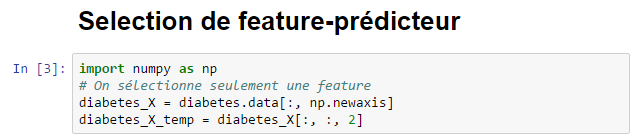


On voit ainsi qu'il y'a 10 prédicteurs potentiels et 442 enregistrements. La cible concerne la progression de la maladie et les prédicteurs consistent en différents paramètres dont l'âge, sexe, BMI, etc... (voir référence 1)

3. Les statistiques descriptives simples sur ce dataset sont les suivantes



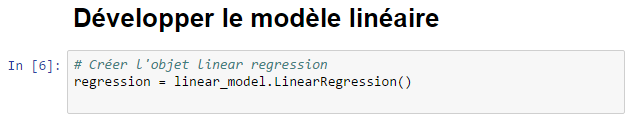
2. Au niveau des prédicteurs d'intérêt, on choisira un seul prédicteur comme suit:



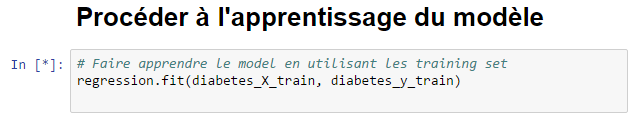
3. On procède maintenant à la division du dataset en ensemble d'apprentissage (80%) et de test (20%) comme suit:



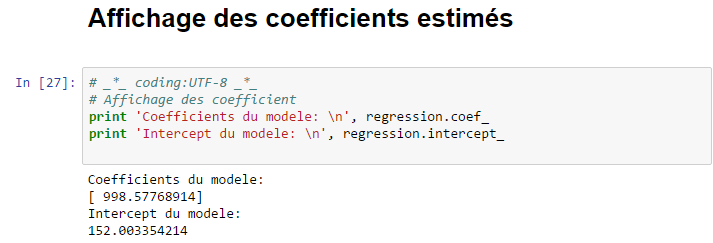
4. On passe maintenant au développement du modèle linéaire comme suit:



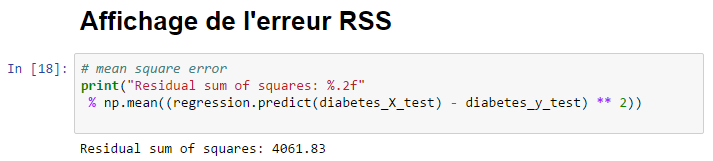
5. La prochaine étape concerne l'apprentissage du modèle proprement dit. On utilisera les ensembles de training sur X et Y comme suit:



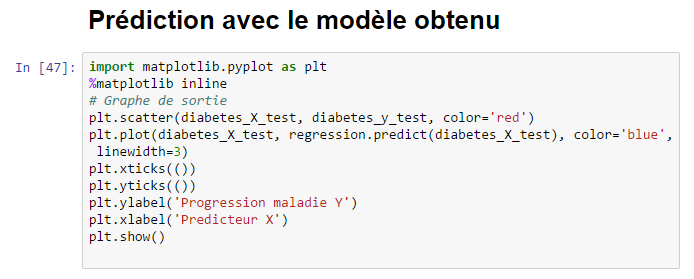
6. Une fois le modèle défini, on affiche les coefficients estimées:



7. On affiche aussi l'erreur RSS:



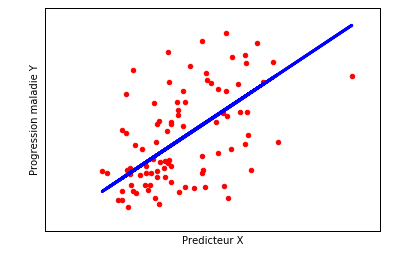
8. Sur la base de l'ensemble de test, on procède à la prédiction en utilisant le modèle que l'on a obtenu:



On notera que pour que l'affichage avec matplotlib se fasse au niveau du notebook, on a utilisé la directive

%matplotlib inline

Le résultat obtenu sera alors



**Références**

1. <http://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf>
2. http://dsdeepdive.blogspot.com/2015/06/linear-regression-with-python.html

**Code final**

# \_\*\_ coding:UTF-8 \_\*\_

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model

# Charger le dataset de patients diabétiques

diabetes = datasets.load\_diabetes()

# On sélectionne seulement une feature

diabetes\_X = diabetes.data[:, np.newaxis]

diabetes\_X\_temp = diabetes\_X[:, :, 2]

# Décomposer les données en training/testing

diabetes\_X\_train = diabetes\_X\_temp[:-20]

diabetes\_X\_test = diabetes\_X\_temp[-20:]

# Décomposer les targets en training/testing

diabetes\_y\_train = diabetes.target[:-20]

diabetes\_y\_test = diabetes.target[-20:]

# Créer l'objet linear regression

regression = linear\_model.LinearRegression()

# Faire apprendre le model en utilisant les training set

regression.fit(diabetes\_X\_train, diabetes\_y\_train)

# Affichage des coefficient

print('Coefficients: \n', regression.coef\_)

# mean square error

print("Residual sum of squares: %.2f"

% np.mean((regression.predict(diabetes\_X\_test) - diabetes\_y\_test) \*\* 2))

# Explained variance score: 1 prediction parfaite

print('Variance score: %.2f' % regression.score(diabetes\_X\_test, diabetes\_y\_test))

# Graphe de sortie

plt.scatter(diabetes\_X\_test, diabetes\_y\_test, color='red')

plt.plot(diabetes\_X\_test, regression.predict(diabetes\_X\_test), color='blue',

linewidth=3)

plt.xticks(())

plt.yticks(())

plt.ylabel('Progression maladie Y')

plt.xlabel('Predicteur X')

plt.show()