SUP'COM
INDP2
2021-2022
Inférence statistique

TD machine sur la régression

1 Objectifs

On veut prédire le prix d'une maisons en s'appuyant sur les caractéristiques de la zone où se trouve celle-ci. On va procéder à une régression linéaire multiple pour expliquer le prix d'une maison (variable à expliquer) par les caractéristiques de la zone où elle se trouve (variables explicatives). Les scripts seront mis à votre disposition dans votre espace moodle.

2 Jeu de données

Le jeu de données que nous utiliserons est le *Boston Housing Dataset*. Il concerne un échantillon de $N = 506$ maisons (on dit instances) situées dans divers endroits de la ville de Boston. Il fournit le prix de chaque maison ainsi que les valeurs des 13 caractéristiques décrites dans le tableau 1.

TABLE 1 – Caractéristiques des zones cibles.

Caractéristique	Description
CRIM	taux de crimes par habitant
ZN	proportion de zones résidentielles sur des lots de 25,000 pieds carrés (sq.ft.)
INDUS	les zones d'activité non commerciale de la ville
CHAS	variable binaire qui vaut 1 si des terrains bordent la rivière Charles et 0 sinon
NOX	concentration d'oxydes nitriques
RM	nombre moyen de pièces par logement
AGE	proportion de logements occupés par leurs propriétaires et construits avant 1940
DIS	distances pondérées à 5 centres de recrutement
RAD	indice d'accessibilité aux autoroutes radiales
TAX	taux de l'impôt foncier sur 10 000$
PTRATIO	taux d'encadrement (rapport nombre d'élèves par nombre d'enseignants)
B	$1000(Bk - 0.63)^2$ où Bk est la proportion de personnes de couleur noire
LSTAT	pourcentage de personnes à faibles revenus

Le prix médian de la maison (en kilodollars) est également fourni dans la variable MEDV.

3 Jeu de données

3.1 Importation du jeu de données

On va utiliser scikit-learn pour bénéficier des outils de statistiques. De plus, on peut importer directement les données à partir du scikit-learn. Pour cela, on suit les étapes suivantes.

1. On commence par importer les librairies.
   import numpy as np

import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import sklearn
import statsmodels.api as sm
import seaborn as sns
sns.set_style("whitegrid")
sns.set_context("poster")
from matplotlib import rcParams

2. On importe les données et on les stocke dans la variable boston.
from sklearn.datasets import load_boston
boston = load_boston()

3.2 Formation du dataframe

1. La variable boston est un dictionnaire (sorte de listequi au lieu d'utiliser des index, utilise des clés alphanumériques). On peut vérifier ses clés de la manière suivante.
print(boston.keys())
2. Pour vérifier sa taille (nombre N d'instances en ligne et nombre de caractéristiques en colonnes, il suffit d'écrire :
print(boston.data.shape)
3. Pour avoir les noms de attributs, il suffit de taper :
print(boston.feature_names)
4. Un descriptif des attributs peut être obtenu en exécutant le code :
print(boston.DESCR)
On retrouve les mêmes informations apparaissant dans le tableau 1.
5. On convertit les données du fichier boston.data en pandas en appelant la pd.DataFrame(). En sortie, on obtient le dataframe appelé bos c.à.d. une matrice pouvant avoir des colonnes de types différents :
bos = pd.DataFrame(boston.data)
6. On peut lire les 5 premières données avec bos.head() :
print(bos.head())
7. Comme les noms des colonnes ne sont pas directement insérés dans le dataframe, on va convertir le numéro de colonne en un nom pour faciliter le répérage des attributs :
bos.columns = boston.feature_names
8. On peut vérifier en lisant de nouveau les 5 premières données avec les noms des colonnes :
print(bos.head())
9. La colonne-cible associée à la variable à expliquer est disponible dans un autre attribut appelé target. On vérifie la forme du boston.target :
print(boston.target.shape)
10. On insère cette colonne-cible dans le dataframe et on vérifie le contenu des 5 premières données du dataframe augmenté :
bos['MEDV'] = boston.target
print(bos.head())

4 Quelques statistiques de base

1. Distribution des valeurs de la colonne-cible *Y= bos['MEDV']*
   plt.figure( )
   plt.title("Distributions of the prices (in k$)")
   plt.hist(Y, bins = 30)
   plt.show() Commenter l'allure de la distribution.
2. Calculer la matrice d'auto-corrélation des attributs de taille 14×14 et affichage sous forme de *heatmap* :
   correlation_matrix = bos.corr().round(2)
   sns.heatmap(data=correlation_matrix, annot=False)
3. **Ajouter le code pour afficher les valeurs de leurs corrélations avec MEDV. Commenter comment est la corrélation des attributs LSTAT et RM avec la variable-cible MEDV.**

5 Régression sur LSTAT

1. On importe de sklearn le package de régression linéaire :
   from sklearn.linear_model import LinearRegression
   lm = LinearRegression()
   from sklearn.metrics import r2_score
2. On met en forme les données explicatives : *X1_1D=bos['LSTAT']*
   X1 = np.array(X1_1D).reshape((-1, 1))
3. On affiche le nuage de points LSTAT et MEDV :
   plt.figure(1)
   plt.scatter(X1, Y, marker='o')
   plt.title("Cloud points")
   plt.xlabel('LSTAT')
   plt.ylabel('PRICE')
4. On crée le modèle de régression : *lm.fit(X1,Y)*
   Y_pred1 = lm.predict(X1)
5. On affiche les valeurs des paramètres du modèle : *print('intercept= :.2f'.format(lm.intercept_))*
   print('Coefficients= ', lm.coef_)
6. On affiche la valeur prédite en fonction de la valeur réelle du prix :
   plt.figure( )
   plt.scatter(Y,Y_pred1)
   plt.plot(Y,Y, color = 'red', linestyle = 'solid')
   plt.xlabel("Real price")
   plt.ylabel("Predicted price")
   plt.title(" LSTAT")
   plt.show()
7. On affiche la valeur du résidu en fonction de la valeur réelle du prix :
   plt.figure( )
   plt.scatter(Y, Y_pred1 - Y,color = "blue", s = 10, label = 'Residual')
   plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2,color = "red",)
   plt.xlabel("Real price")
   plt.ylabel("Residual")

plt.title(" Residual with LSTAT")
plt.show()

8. On trace et affiche la droite de régression :
   plt.figure( )
   plt.plot(X1, Y_pred1, color = "red")
   plt.scatter(X1,Y)
   plt.xlabel("LSTAT")
   plt.ylabel("Price")
   plt.title(" Regression line with LSTAT")
   plt.show()
9. On évalue les performances par l'erreur quadratique moyenne minimale et le coefficient de détermination :
   mse_1 = sklearn.metrics.mean_squared_error(Y, Y_pred1)
   print('Mean square error = :.2f'.format(mse_1))
   from sklearn.metrics import r2_score
   r2_1 = r2_score(Y, Y_pred1)
   print('Coeff of determination = :.2f'.format(r2_1))

6 Régression multiple

1. On recommence en utilisant la régression multiple de MEDV par LSTAT.
2. **Ajouter le code pour tracer et afficher l'hyperplan de régression sur le nuage de points 3D (LSTAT,RM,MEDV).**
3. On recommence en utilisant la régression multiple de MEDV par tous les 13 attributs.
4. **Expliquez l'évolution de l'erreur quadratique moyenne minimale et du coefficient de détermination en fonction du nombre de variables explicatives. Conclure.**