TP 01 SAD : Régression Linéaire :

Introduction :

Vous aurez besoin d'avoir installé scikit-learn .

Je vous recommande aussi d'utiliser des classeurs iPython / Spyder ou Jupiter qui sont pratiques pour travailler étape par étape en data science, et afficher vos résultats au fur et à mesure directement dans un navigateur web.

Problématique :

Notre problématique et celle des loyers qui peuvent être estimés en fonction de la superficie des logements. La question qu'on essaie de résoudre est :

Étant donné les caractéristiques d'un appartement, combien devrait on normalement payer de loyer?

Imaginons pour l'instant que la seule caractéristique dont nous disposons est la surface de l'appartement. Notre training set est un ensemble de N = 545 observations de surface et leur loyer associé : (x,y)=(surface,loyer)

Vous pouvez télécharger le fichier qui contient les données depuis « Votre Groupe ».

Ce sont des données réelles récupérées sur des sites de location, pour quelques arrondissements parisiens. Quelques points aberrants ont été enlevé (notamment sur les surfaces trop grandes).

Etape 1 :

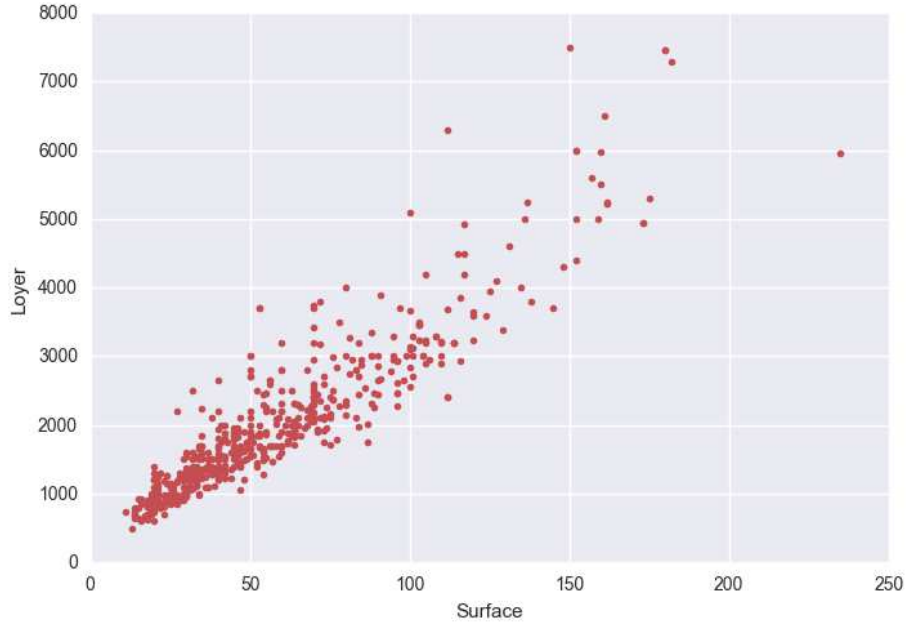
Importer les librairies dont vous aurez besoin dans ce TP : numpy, pandas, matplotlib.pyplot

Etape 2

Charger et Afficher les données d'entraînement : créez un nom da dataframe en utilisant la commande pd.read_csv. Le nom du fichier est 'house.csv'

Afficher le nuage de points dont on dispose en utilisant la commande plt.plot() et plt.show()

RESULTAT :



Qu'est ce que vous remarquez ?

Etape 3 :

Apprentissage : trouver le θ optimal

Pour la régression linéaire, la solution de l'équation de minimisation est exacte :

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

Décomposez le dataset et transformer le en 2 matrices X et Y pour pouvoir effectuer les calculs en utilisant la commande np.matrix().T

Calculez theta en utilisant la commande « np.linalg.inv » ; et « dot »

Imprimer le résultat theta en utilisant la commande print()

RESULTAT :

On trouve donc la valeur numérique de θ pour ces données :

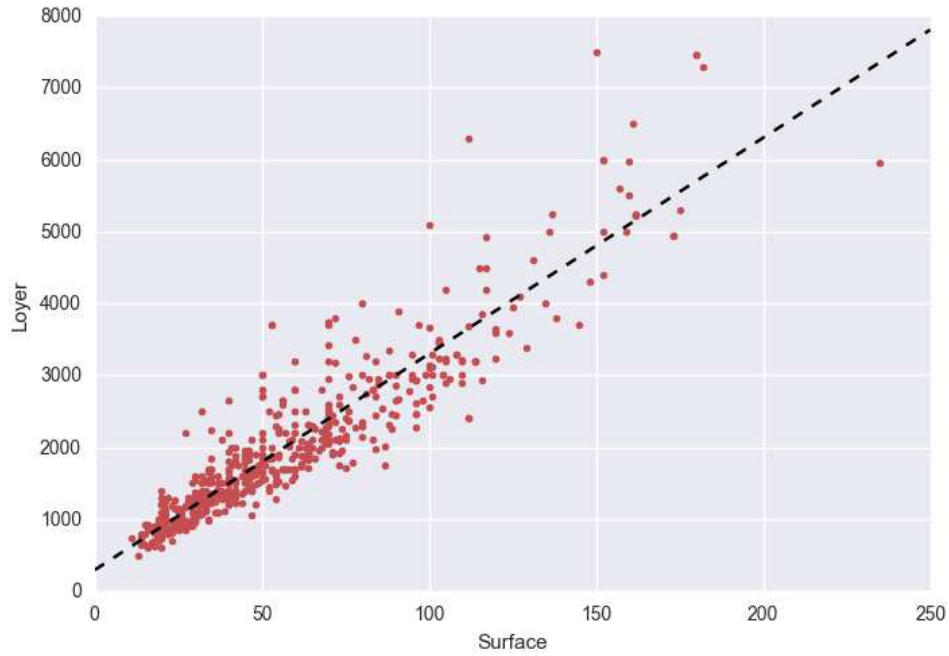
[[828.37]

[29.8]]

Donnez l'équation (approximative) du modèle final qui fitte les données .

Représentez graphiquement la droite trouvée en utilisant les commandes : plt.xlabel(), plt.ylabel(), plt.plot()(affichez la droite entre 0 et 250) et plt.show()

RESULTAT :



Etape 4 : Utiliser le modèle pour effectuer des prédictions

Maintenant qu'on a notre paramètre θ, c'est à dire qu'on a trouvé la droite qui fitte le mieux nos données d'entrainement, on peut effectuer des prédictions sur de nouvelles données, c'est à dire prédire le loyer en fonction de la surface qu'on nous donne en entrée, en appliquant directement la formule du modèle trouvée.

Par exemple, si on l'applique pour une surface de 35m carré ?

On obtient une estimation du loyer égale à :

1872.54

Essayez avec n'importe quelle surface !

Etape 5 :

Mais on peut aussi utiliser un algorithme appelé *Descente de Gradient* pour trouver une approximation de la solution. C'est en particulier utile lorsqu'on a beaucoup de données d'exemples, car c'est assez long pour un ordinateur de calculer la solution exacte ci-dessus (on calcule un inverse de matrice, ce qui n'est pas gratuit en temps de calcul !).

Cette régression est déjà implémentée dans le package scikit-learn. On peut l'utiliser directement de la manière suivante :

```
from sklearn import linear_model

regr = linear_model.LinearRegression()
```

```
regr.fit(surface, loyer)

regr.predict(donnee_test)
```

Ecrire un programme qui utilise ces commandes et trouver les valeurs prédites pour les données de Test.

Aller plus loin : le “vrai” travail de modélisation

Maintenant qu’on a trouvé un premier modèle, il serait possible de tester plein d’hypothèses différentes pour aller plus loin et améliorer ses performances.

Que se passe-t-il si :

- on change l’hypothèse de linéarité (une droite) et qu’on en prend une autre (un polynôme du second degré par exemple) ?
- on teste le modèle avec d’autres types d’erreurs que la distance euclidienne ?
- on ajoute des features (dimensions) supplémentaires en entrée ?
- au fur et à mesure que la surface augmente, les données on l'air d'être de plus en plus "éparses", comment intégrer ce comportement dans ma modélisation ?

Toutes ces questions peuvent (et devraient !) être testées. Trouver la configuration la plus pertinente revient du coup à tester les différents modèles qui en découlent et trouver celui qui convient le mieux en terme de performances...

Pas de panique, un cours entier sera dédié à l’évaluation et l’amélioration des performances du modèle.

En résumé

- À partir d'une problématique et d'un dataset, nous avons considéré une **hypothèse** de travail pour contraindre le modèle : ici nous nous sommes placés dans le cas d'une régression linéaire, qui correspond à contraindre la forme du modèle à une droite.
- Nous avons décomposé l'entraînement de ce modèle sur les observations, afin de déterminer le paramètre (pente et ordonnée à l'origine) de la droite optimale pour ces données. C'est cette partie que l'on appelle **apprentissage du modèle**.
- À l'aide du modèle ainsi trouvé, nous pouvons maintenant effectuer des **prédictions** de montant de loyer à partir de n'importe quelle surface donnée.
- On peut toujours **améliorer ce modèle** (une fois qu'on saura évaluer ses performances) en testant par exemple d'autres hypothèses, en ajoutant de nouvelles caractéristiques sur les observations ou en testant d'autres types de loss qui seront peut-être plus appropriés pour ce cas... Le travail du data scientist ne s'arrête pas là !