R5.A.12 & R5.B.10 - Modélisation Mathématique - TP 05

Antoine Nongaillard 2024–2025

La **régression linéaire** est sans aucun doute l'algorithme de **Machine Learning (ML)** le plus simple qui soit. Comme tous les algorithmes de ML, il est instancié à partir d'exemples, et permet ensuite de généraliser à de nouvelles données. Ce modèle est suffisamment simple pour qu'une solution mathématique immédiate existe dans tous les cas. Il «suffit» de *calibrer le modèle* et non de l'*entraîner* comme avec les réseaux de neurones ou les approches basées sur l'apprentissage par renforcement.

En guise d'illustration on prendra ici un exemple trivial avec quelques données, qui pourraient par exemple corres- pondre à prédiction du poids d'un individu à partir de sa taille, la température en fonction de l'altitude, ou les ventes d'un produit en fonction de sa qualité. Toutes les méthodes du framework que nous allons utiliser, **ScikitLearn**, fonctionnent avec des `np.array`.

La technique d'utilisation est toujours la même :

1. récupérer les données sous forme de `np.array`
2. créer un modèle
3. calibrer ce modèle avec la méthode `fit(...)`
4. généraliser (faire des prédictions) avec la méthode `predict(...)`

In [1]:
```
from IPython.core.display import HTML
def AFFICHER(letexte) :
    display(HTML(f"<div class=\"alert alert-box alert-info\">{letexte}</d
AFFICHER(f"<b>Titre :</b><BR>{'but '*3}")
```

Titre :
but but but

In [2]:
```
# imports
# exo 1
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_s
import matplotlib.pyplot as plt
# exo 2
from sklearn.preprocessing import PolynomialFeatures
# exo 3
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
# chemins pour lecture des fichiers de données

chemin = 'data/'
```

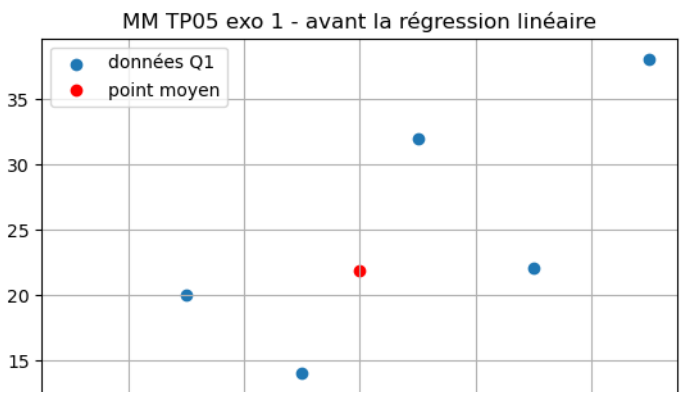
Exercice 1 : Une première regression linéaire

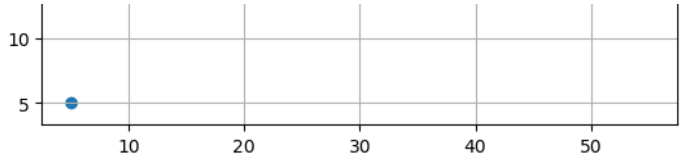
Q1. Mettez en forme les observations (x_i, y_i) suivantes afin qu'elles soient utilisables :

```
    (5, 5), (15,20), (25, 14), (35, 32), (45, 22), (55,
38)
```

Affichez-les comme nuage de points sur une grille.

In [4]:
```
#
```





Q2. Créez le modèle de régression qui représente le mieux ces observations, c'est-à-dire la droite minimisant l'écart avec les points.

In [5]:

ATTENTION au '.reshape((-1, 1))' !!!

In [6]:
```
# Linear Model :
#
```

Q3. Quel est l'équation de la droite obtenue ainsi ? Autrement dit, quelles sont les valeurs de la pente et de l'ordonnée à l'origine ?

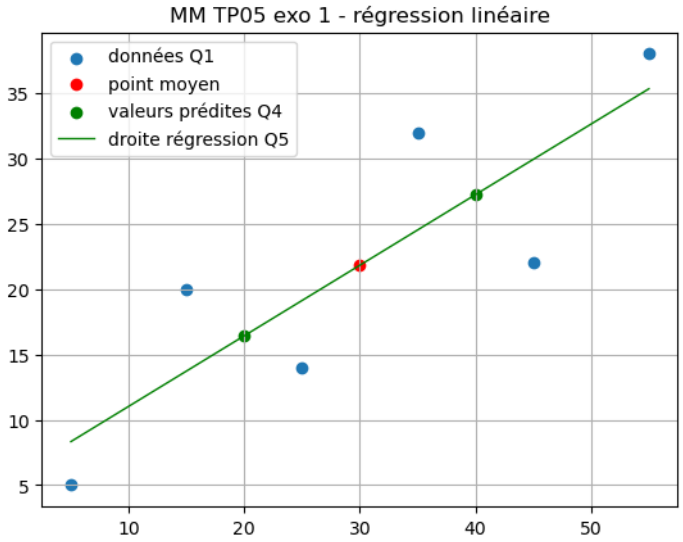
In [7]:
```
#
```
```
intercept = ordonnée à l'origine :   LM.intercept_=5.633333333333333
slope     = pente                :   LM.coef_=array([0.54])
```

Q4. Puisque nous disposons d'un modèle, effectuons quelques prédictions. Quelles sont les valeurs prédites pour $x_1 = 20$ et $x_2 =40$?

In [8]:
```
#
```
```
valeurs testées     : x_test=array([20, 40])
predicted response  : y_pred=array([16.43333333, 27.23333333])
```

Q5. Tracez la droite de régression ainsi que le nuage de points. Vérifiez que vous retrouvez bien les valeurs prédites pour les nouvelles observations données précédemment.

In [9]:
```
#
```
```
Tracé de la droite de régression passant par les points :
X=array([ 5., 55.])
Y=array([ 8.33333333, 35.33333333])
```



Q6. Quantifiez la qualité du modèle en affichant les scores R2, MSE, RMSE et MAE.

In [10]:
```
#
```
```
coeff de détermination                : R2   =  0.715875613747954
erreur quadratique moyenne            : MSE  =  33.75555555555556
racine de l'erreur quadratique moyenne : RMSE =  5.809953145728075
erreur absolue moyenne                : MAE  =  5.466666666666668
```

Exercice 2 : Le cas polynomial

La régression polynomiale se gère de la même manière que la régression linéaire, mais avec une étape supplémentaire. Il est en effet nécessaire de transformer le tableau des entrées pour inclure des termes non linéaires.

Q1. Mettez en forme les observations (x_i, y_i) suivantes afin qu'elles soient utilisables :

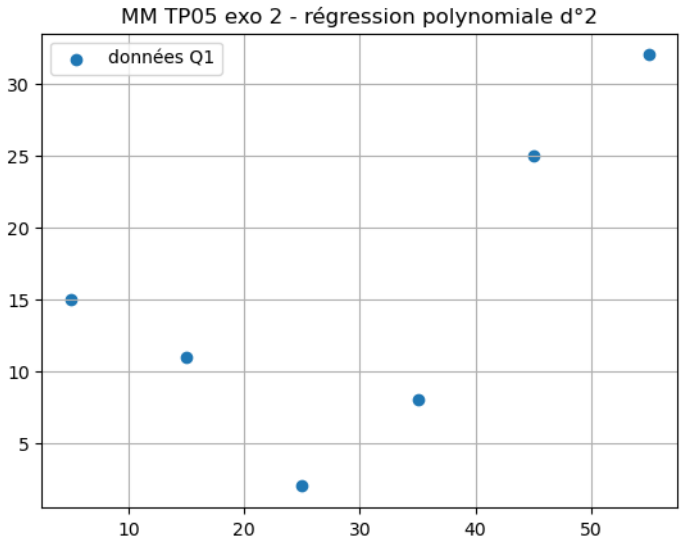
```
    (5, 15), (15, 11), (25, 2), (35, 8), (45, 25), (55,
32)
```

Affichez-les comme nuage de points sur une grille.

In [11]:
```
#
```

```
données Exo 2 Q1 :
x=array([ 5, 15, 25, 35, 45, 55])
y=array([15, 11,  2,  8, 25, 32])
```

Attention, il faudra mettre les abscisses x sous forme de matrice-colonne par un 'reshape(-1,1)' !!!



Q2. Créez le modèle de régression et calibrez-le avec les données précédentes.

In [12]:
```
#
```

Q3. Puisque nous disposons d'un modèle, effectuons quelques prédictions. Quelles sont les valeurs prédites pour $x_1 = 20$ et $x_2 =40$?

In [13]:
```
#
```

```
x_test=array([20, 40])
y_pred=array([ 6.25803571, 13.85803571])
```

Q4. Affichez les coefficients du polynôme, puis calculez les scores R2, MSE, RMSE et MAE afin d'évaluer la qualité du modèle.

In [14]:
```
#
```

```
intercept    : 21.37232142857143
coefficients : [-1.32357143  0.02839286]
```

In [15]:
```
#
```

Pour comprendre et vérifier... mais en pratique on utilisera 'PM.predict(...)'

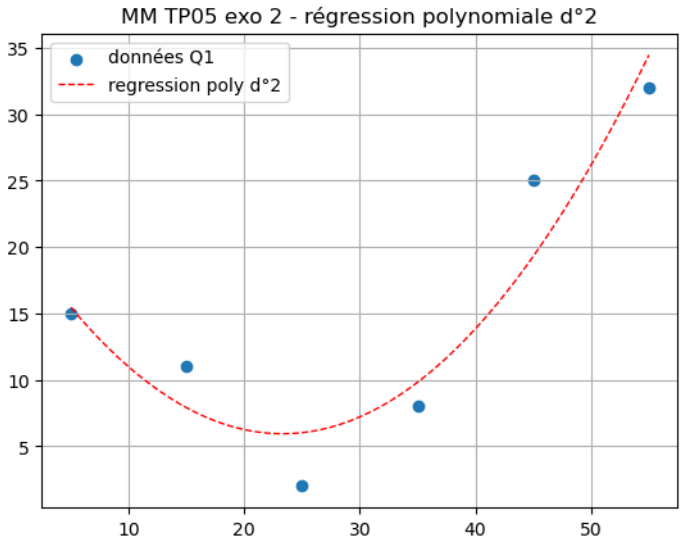
```
x -> 21.37232142857143 + -1.3235714285714286 * x + 0.02839285714285714 *
x**2

+20.000 -> +6.258
+40.000 -> +13.858
```

In [16]:
```
# Quelques mesures
#
```

```
R2   =  0.8908516262498564
MSE  =  11.305952380952382
RMSE =  3.3624325095014744
MAE  =  2.928571428571427
```

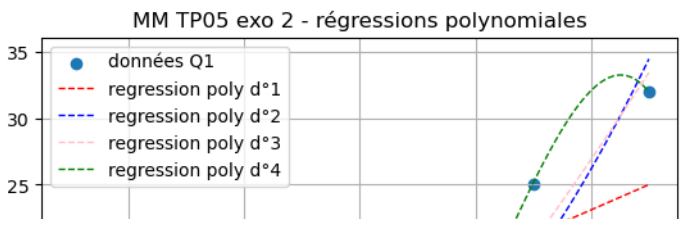
Q5. Affichez les données initiales ainsi que le polynôme calculé.

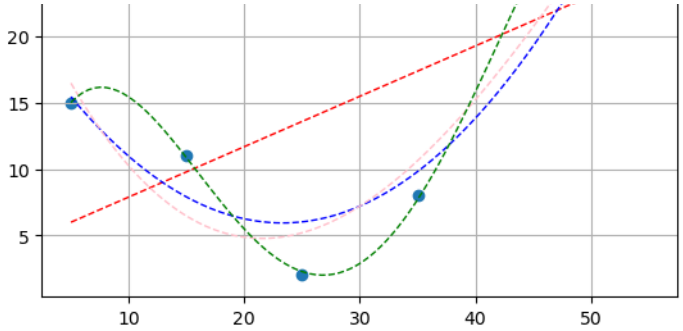
In [17]:
```
#
```



In [18]:
```
# NON DEMANDÉ dans le sujet de TP
```

```
========================================
= Régression par un polynôme de degré 1 =
========================================
R2   =  0.4065969428801288
MSE  =  61.46666666666666
RMSE =  7.840068026915753
MAE  =  7.0
========================================
= Régression par un polynôme de degré 2 =
========================================
R2   =  0.8908516262498564
MSE  =  11.305952380952379
RMSE =  3.362432509501474
MAE  =  2.9285714285714355
========================================
= Régression par un polynôme de degré 3 =
========================================
R2   =  0.9030890446819652
MSE  =  10.038359788359775
RMSE =  3.1683370698774738
MAE  =  2.9285714285722797
========================================
= Régression par un polynôme de degré 4 =
========================================
R2   =  0.9996871368552784
MSE  =  0.03240740740740451
RMSE =  0.18002057495576584
MAE  =  0.14814814810214175
```





Q6. Malheureusement, vous n'avez que quelques points pour établir votre modèle.

Q6.1. Récupérez le fichier `data_mm05_additional.csv`. Incorporez ces nouvelles données et évaluez la précision de votre modèle avec ces nouvelles données (R2, MSE, RMSE, MAE). Affichez finalement tous les points et le polynôme de régression.

In [19]:
```
#
```

modèle polynomial de degré 2 créé sur 6 points, appliqué sur 200 points :

```
R2   =  0.3417158959894877
MSE  =  81.74625831627304
RMSE =  9.041363742061982
MAE  =  7.397123864218331
```

Q6.2. Trouvez un nouveau modèle si nécessaire qui obtiendrait de meilleurs résultats.

Exercice 3 : Prédiction de loyers

Vous êtes consulté par une agence immobilière pour prédire les loyers des différents arrondissements de Paris, afin de les aider à prendre des décisions d'achat d'appartements.

Q1. Récupérez les données du fichier `data_mm05_house.csv`. Affichez les 5 premières lignes du fichier pour comprendre sa structure ainsi que le nombre de lignes qu'il contient.

In [23]:
```
#
```

	price	surface	arrondissement
0	1820	46.1	1.0
1	1750	41.0	1.0
2	1900	55.0	1.0
3	1950	46.0	1.0
4	1950	49.0	1.0

Écrivons les arrondissements comme des entiers...

	price	surface	arrondissement
0	1820	46.10	1
1	1750	41.00	1
2	1900	55.00	1
3	1950	46.00	1
4	1950	49.00	1
...	...	...	...
822	850	35.00	10
823	700	10.00	10
824	1280	34.00	10
825	1545	65.00	10

826	1000	21.43	10

827 rows × 3 columns

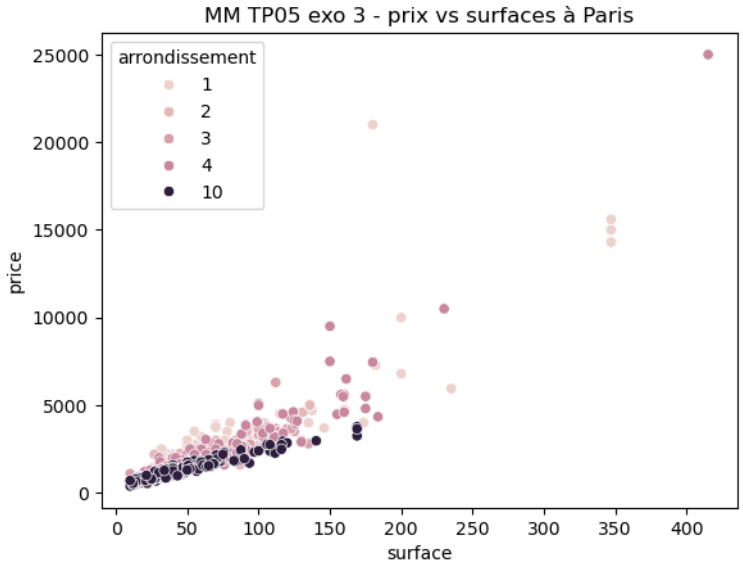
Q2. Affichez la description statistique des données, et représentez les données sous forme d'un nuage de points pour déterminer la présence éventuelle d'outliers.

In [ ]:

In [24]:
```
#
```

```
price           0
surface         5
arrondissement  5
dtype: int64
```

In [25]:
```
#
```



Q3. Nettoyez les données et ignorez les outliers. Ré-affichez le nuage de points correspondant. Faites-en sorte de pouvoir distinguer les arrondissements.

In [26]:
```
#
```

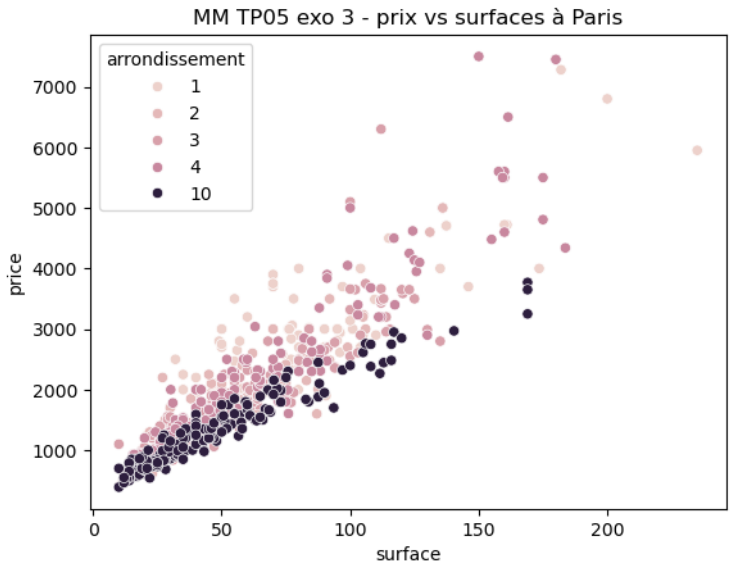
```
(812, 3)
```

	price	surface	arrondissement
0	1820	46.10	1
1	1750	41.00	1
2	1900	55.00	1
3	1950	46.00	1
4	1950	49.00	1
...	...	...	...
807	850	35.00	10
808	700	10.00	10
809	1280	34.00	10
810	1545	65.00	10
811	1000	21.43	10

812 rows × 3 columns

```
price           0
surface         0
arrondissement  0
dtype: int64
```

In [27]:
```
#
```

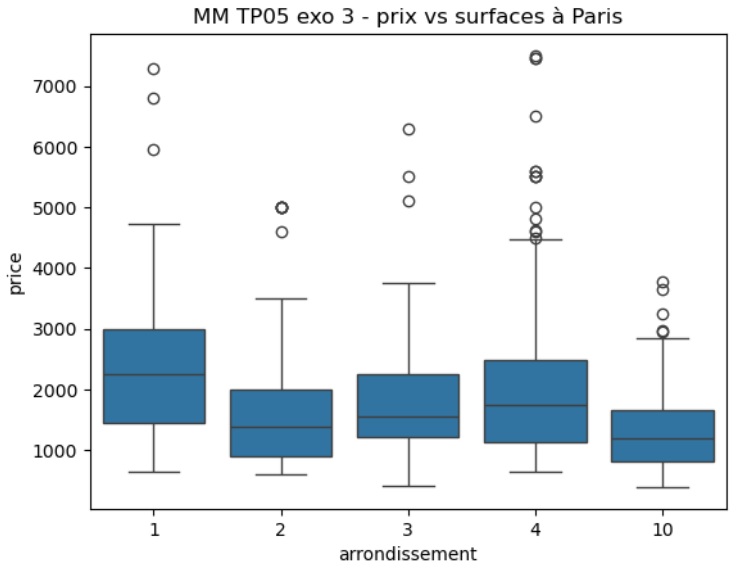


Q4. Représentez la répartition des prix par arrondissement.

Est-elle plus ou moins semblable dans les différents arrondissements ?

Où les prix sont-ils le moins élevés ?

In [28]:
```
#
```



Q5. Vaut-il mieux faire un unique modèle de régression commun quelque soit l'arrondissement, ou un modèle différent par arrondissement ?

Q5.1. Afin d'évaluer correctement les modèles, divisez votre jeu de données. Conservez 30% des données pour les tests.

In [29]:
```
xtrain, xtest, ytrain, ytest = train_test_split(data[['surface', 'arrondi
```

In [30]:
```
#
```

```
'xtrain'
```

	surface	arrondissement
679	39.0	10
164	58.0	2
258	18.0	2
263	27.0	2
787	61.0	10
...	...	...
667	21.0	10
467	72.2	4
555	50.0	4
160	46.0	2
747	93.6	10

568 rows × 2 columns

```
'ytrain'
```

	price
679	1280
164	2400
258	799
263	1166
787	1574
...	...
667	800
467	1800
555	1380
160	2100
747	1700

568 rows × 1 columns

```
'xtest'
```

	surface	arrondissement
120	235.0	1
801	14.0	10
697	100.0	10
245	30.0	2
560	46.3	4
...	...	...
553	44.0	4
226	65.0	2
326	39.0	3
396	23.0	3

102	73.6	1

244 rows × 2 columns

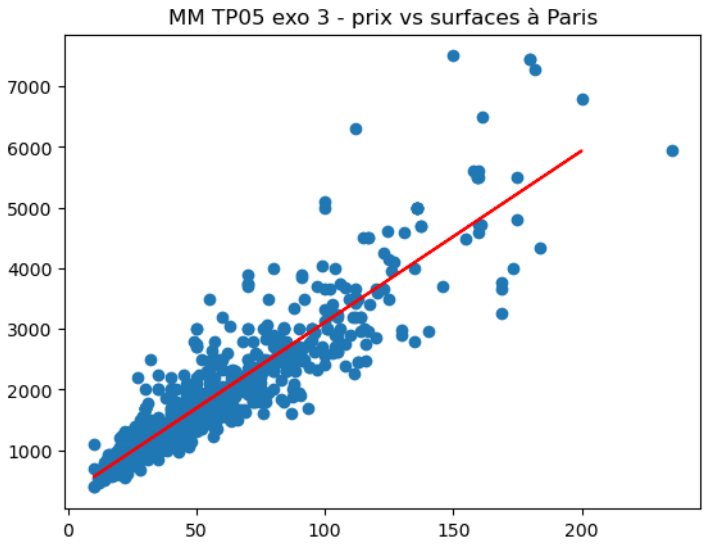
'ytest'

	price
120	5950
801	650
697	2400
245	1700
560	1624
...	...
553	1460
226	2175
326	1295
396	816
102	2587

244 rows × 1 columns

Q5.2. Créez le modèle de régression unique pour l'ensemble des arrondissements. Tracez le et évaluer ses scores.

In [31]:
```
#
```

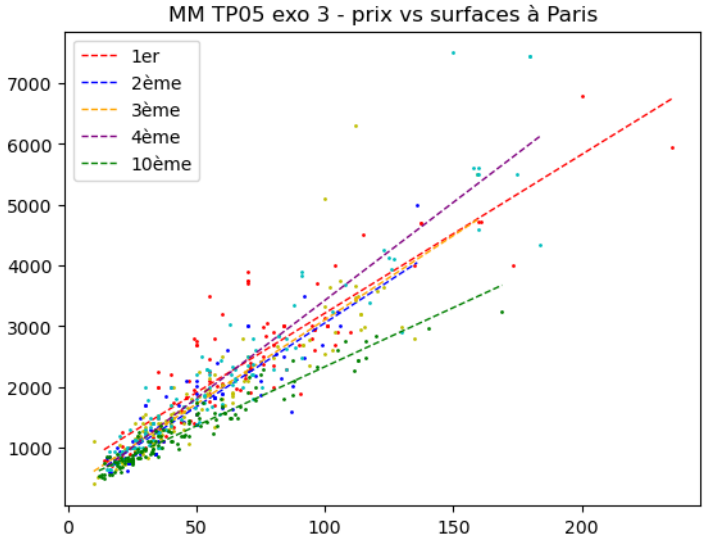


In [32]:
```
x = xtest[['surface']]
print("R2   = ", LM.score(x,ytest))
print("MSE  = ", mean_squared_error(ytest, LM.predict(x)))
print("RMSE = ", root_mean_squared_error(ytest, LM.predict(x)))
print("MAE  = ", mean_absolute_error(ytest, LM.predict(x)))
```

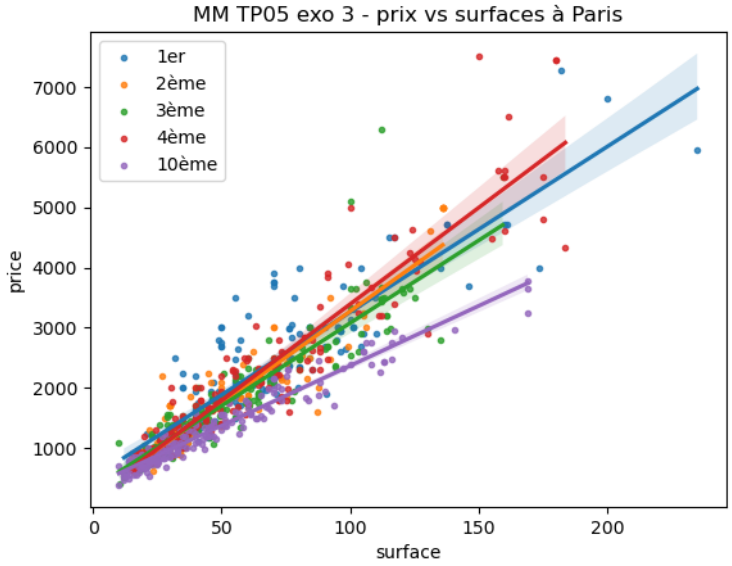
```
R2   =  0.8213272442793308
MSE  =  220013.18888134815
RMSE =  469.05563516639273
MAE  =  301.2138667027993
```

Q5.3. Créez les modèles propres à chaque arrondissement. Tracez-les et évaluer leurs scores.

In [33]:
```
#
```


MM TP05 exo 3 - prix vs surfaces à Paris
1er
2ème
3ème
4ème
10ème
7000
6000
5000
4000
3000
2000
1000
0
50
100
150
200

In [34]:
```
#
```


MM TP05 exo 3 - prix vs surfaces à Paris
1er
2ème
3ème
4ème
10ème
price
surface
7000
6000
5000
4000
3000
2000
1000
0
50
100
150
200

In [ ]:
```

```