

**Université
de Rennes**



Rapport TP Final MPC

**Roberto Henry Chacon Suarez
Manh-Huan Nguyen**

M1 Miage - ISTIC - 2022/2023

Ce rapport a pour but d'expliquer la réalisation des différentes étapes du TP final de MPC, dont le but est de prédire les prix de maisons à partir d'un dataset.

Le code que nous avons réalisé tout au long du projet se trouve dans le Jupyter-notebook, nommé **tpFinal.ipynb**.

Le dataset

L'objectif du projet est de prédire le prix des maisons. Le dataset contient 13397 maisons composées des 20 caractéristiques :

- **id** - identifiant unique d'une maison
- **pricePrice** - la variable cible à prédire
- **bedroomsNumber** - nb des chambres
- **bathroomsNumber** - nb des salles de bain
- **sqft_livingsquare** - mètreage de la maison
- **sqft_lotsquare** - mètreage du lot
- **floorsTotal** - nb d'étages de la maison
- **waterfront** - vue sur le bord de la mer
- **view** - Indice de 0 à 4 de la qualité de la vue de la propriété
- **condition** - Qualité moyenne de la condition
- **grade** - note globale attribué au logement (sur la base du système de notation de King County)
- **sqft_above** - Mètreage de la maison hors sous-sol
- **sqft_basement** - Mètreage du sous-sol
- **yr_built** - Année de construction
- **yr_renovated** - Année de rénovation
- **zipcode** - Code postal
- **lat** - Coordonnées de latitude
- **long** - Coordonnées de longitude
- **sqft_living15** - La superficie de l'espace intérieur du logement pour les 15 voisins les plus proches.
- **sqft_lot15** - La superficie des terrains des 15 voisins les plus proches.

Regression Simple

L'objectif de cette partie est de trouver la meilleure variable pour prédire les prix des maisons.

Nous nous sommes donc intéressés sur 3 critères :

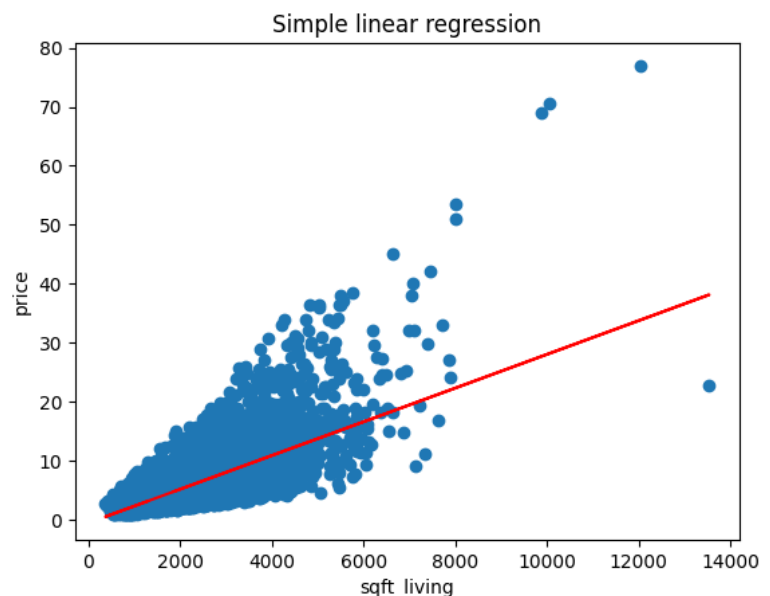
1. Le coefficient de détermination R^2 .
2. La quantité d'information non expliquée par le modèle I_r .
3. L'erreur de généralisation MSE .

En parcourant toutes les variables et en comparant le **coef. de détermination**, nous avons trouvé les résultats suivants :

```
R^2 = 0.49640102806105757
I_r = -2.376054908381775e-11
I_t = 186163.56522629363
Meilleure variable = sqft_living
```

La meilleure variable est le **sqft_living**, dont le modèle est le suivant :

OLS Regression Results						
Dep. Variable:	price			R-squared:	0.496	
Model:	OLS			Adj. R-squared:	0.496	
Method:	Least Squares			F-statistic:	1.320e+04	
Date:	Fri, 28 Apr 2023			Prob (F-statistic):	0.00	
Time:	11:09:15			Log-Likelihood:	-32042.	
No. Observations:	13397			AIC:	6.409e+04	
Df Residuals:	13395			BIC:	6.410e+04	
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-0.5466	0.057	-9.646	0.000	-0.658	-0.436
sqft_living	0.0029	2.49e-05	114.907	0.000	0.003	0.003
Omnibus:	9718.781	Durbin-Watson:		1.981		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		441688.826		
Skew:	2.993	Prob(JB):		0.00		
Kurtosis:	30.485	Cond. No.		5.65e+03		



Même si le R^2 n'est pas forcément très proche de 1, nous pouvons remarquer qu'il existe une corrélation significative entre la variable **sqft_living** et **price**.

En outre, afin d'obtenir l'erreur de généralisation du modèle trouvé, nous avons implémenté la méthode de train/test split vue en cours et montrée en TP.

Nous avons donc suivi la même stratégie de créer les 3 fonctions qui faciliteront le reste des tests :

- Une fonction **my_regression** qui permet de calculer un modèle de régression sur un dataset.
- Une fonction **my_prediction** qui permet de calculer une prédiction en fonction d'un modèle.
- Une fonction **generalization_error_split** qui permet d'estimer l'erreur de généralisation par la méthode train / test split.

Finalement, grâce à ces fonctions nous avons obtenu le *MSE* de notre modèle :

```
MSE =  
generalization_error_split(train,test,houses.columns.get_loc('sqft_living'),hou  
ses.columns.get_loc('price'))  
  
7.001875041181125
```

Regression Multiple

Dans cette partie nous nous sommes servis de la fonction créée juste avant pour obtenir le modèle :

```
multiple_model = my_regression(houses, range(1,18), 0)  
multiple_model.summary()
```

OLS Regression Results			
Dep. Variable:	price	R-squared:	0.698
Model:	OLS	Adj. R-squared:	0.698
Method:	Least Squares	F-statistic:	1820.
Date:	Fri, 28 Apr 2023	Prob (F-statistic):	0.00
Time:	11:09:16	Log-Likelihood:	-28615.
No. Observations:	13397	AIC:	5.727e+04
Df Residuals:	13379	BIC:	5.740e+04
Df Model:	17		

Nous pouvons remarquer que ce modèle compte un meilleur coefficient de détermination R^2 que dans le modèle d'une seule variable.

Également, à l'aide des fonctions implémentées avant, nous avons calculé l'erreur de généralisation :

```
# On obtient les train et test sets
train , test = train_test_split(houses, test_size = 0.30, random_state=20)

# On calcule le MSE
MSE = generalization_error_split(train, test, range(1,18), 0)

4.448779409174057
```

Nous pouvons remarquer que le modèle à 18 variables a un MSE plus petit ($MSE = 4.44$) que celui d'une variable ($MSE = 7.001$). Donc il est clairement plus intéressant.

Sélection des variables

Cette partie consiste à trouver le meilleur sous-ensemble de variables parmi les variables prédictives (X_1, X_{18}) qui conduisent au meilleur modèle.

Il existe plusieurs procédures, mais nous nous sommes centrés sur (en utilisant l'erreur de généralisation comme critère de performance) :

- La Recherche en avant (Forward Search).
- La Recherche en arrière (Backward Search).
- La Recherche Exhaustive.

Nous avons implémenté des fonctions pour effectuer chacune de ces procédures dans le notebook. Les résultats obtenus sont les suivants :

Téchnique	Meilleure Variable	MSE
Forward Search	<i>waterfront</i> (6)	4.217840151252102
Backward Search	<i>sqft_basement</i> (11)	4.217840151253607
Exhaustive Search	NA	NA

Nous pouvons remarquer que la Forward Search est celle qui a trouvé le meilleur modèle avec un MSE légèrement inférieur à celui de la Backward Search.

Cependant, nous avons implémenté la recherche exhaustive mais nous n'avons pas pu tester dû au fait que la méthode évalue toutes les combinaisons possibles et elle est coûteuse en temps.

Modèles non-linéaires

Les modèles non linéaires permettent de modéliser des données plus complexes et ont tendance à être plus précis, mais aussi à être plus difficiles à interpréter que les modèles linéaires.

Nous avons implémenté la technique vue en cours et en TP : La régression polynomiale. Pour cela, nous nous sommes servis de la fonction `add_polynomial_feature`, laquelle ajoute des variables polynomiales à un dataset donné.

Ensuite, nous avons cherché le meilleur modèle en comparant les MSE d'une liste de degrés. Nous avons donc obtenu les résultats suivants :

```
Meilleur degré : 2
Meilleur MSE : 3.577486104959523
```









Nous pouvons voir que, en considérant les 18 variables, le meilleur degré est le 2 avec un MSE de 3.57.

Finalement, à partir de ce modèle nous avons estimé les prédictions des prix du dataset **house_competition**, pour les déposer ensuite sur Kaggle.

Le score obtenu est le suivant (le classement peut avoir changé) :

Public Private

This leaderboard is calculated with approximately 50% of the test data. The final results will be based on the other 50%, so the final standings may be different.

#	Team	Members	Score	Entries	Last	Join
1	konan		2.64956	3	16h	
2	Roberto Chacon	 robertochacon on	2.85297	1	4m	
 Your First Entry! Welcome to the leaderboard!		 Roberto Chacon & Manh-Huan Nguyen				
3	Olivier Da Pozzo	 Kaggle Novice	2.86883	6	2h	
4	HUGO THOMAS & ROSANE BENKH ELIL	 Rennes, Brittany, France M1 MIAE Student	3.01863	3	1d	
5	Meril Mianguilla		3.24531	3	2d	
6	Hugo		3.36559	7	19h	