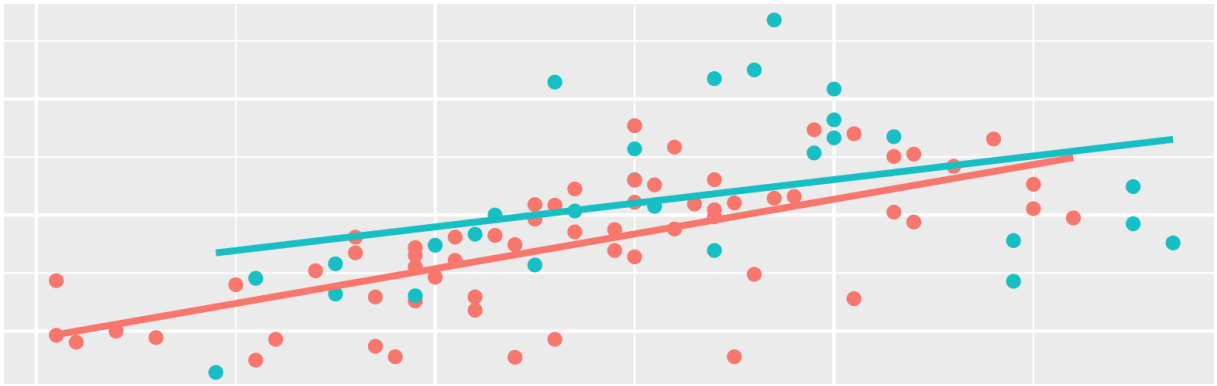
Prévisions des prix du logement

A l'aide d'un Modèle de régression linéaire multiple



- ELASRAOUI Badreddine
- EL JABIRY Reda
- EDDAOU M'hamed Issam

Index

Introduction

L'analyse de régression linéaire sert à prévoir la valeur d'une variable en fonction de la valeur d'une autre variable. La variable dont vous souhaitez prévoir la valeur est la variable dépendante. La variable que vous utilisez pour prévoir la valeur de l'autre variable est la variable indépendante.

Ce type d'analyse estime les coefficients de l'équation linéaire, impliquant une ou plusieurs variables indépendantes, qui estiment le mieux la valeur de la variable dépendante. La régression linéaire consiste en la détermination d'une droite ou d'une surface qui réduit les écarts entre les valeurs de sortie prévues et réelles. Il existe des calculatrices de régression linéaire simple qui utilisent une méthode des moindres carrés pour découvrir la ligne la mieux adaptée pour un ensemble de données appariées. La valeur de X (variable dépendante) est ensuite estimée à partir de Y (variable indépendante).

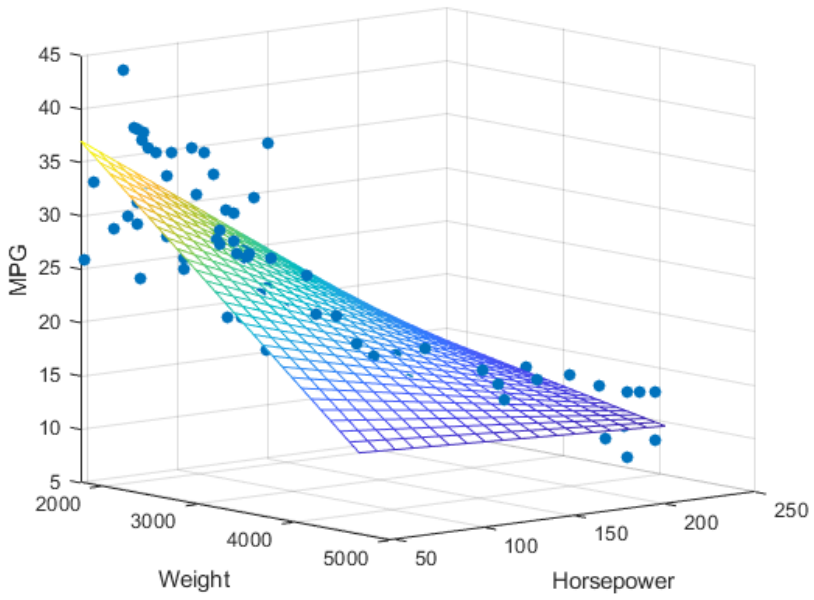


Figure 1 : Exemple d'un régression linéaire multiple

Outils utilisés

1. Jupyter Notebook :

Les notebooks Jupyter sont des cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web , on va l'utilisée comme un compilateur python dans le segment de ce projet

2. Librairies python :

- ***Pandas*** **:** Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles

- ***Matplotlib*** **:** est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous forme de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy

- ***Statmodels*** **:** Statsmodels est un package Python qui permet aux utilisateurs d'explorer des données, d'estimer des modèles statistiques et d'effectuer des tests statistiques

- ***Scikit-Learn*** **:** est une bibliothèque libre Python destinée à l'apprentissage automatique

Présentation des données

Dans notre projet on va utiliser une data Set de logement de l'USA , qui contient 5000 observations et 7 colonne :

- **''Avg. Area Income''** : revenu moyen de la région
- **''Avg. Area House Age'' :** âge moyen de la maison
- **'' Avg. Area Number of Rooms '' :** nombre moyen de chambres
- **'' Avg. Area Number of Bedrooms '' :** nombre moyen de chambres
- **'' Area Population '' :** Population de la région
- **'' Price '' :** Prix
- **'' Address '' :** Adresse

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanielltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

Figure 2 : Quelque lignes du base de données

Source : *https://www.kaggle.com/datasets/vedavyasv/usa-housing*

Présentation du projet :

Prévisions des prix du logement

I- Chargement des données :

A l'aide de la méthode **read_csv** , on va chargé la base de données qui est en format csv.

```
df = pd.read_csv("USA_Housing.csv")
```

Figure 3 : Chargement des données

II- Nettoyage des données :

1. Vérification des valeurs manquantes

```
df.isna().sum()
```

```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
Address                         0
dtype: int64
```

Figure 4 : Nettoyage des données

⇨ On remarque qu'il y a aucune valeur manquante dans notre base de données .

2. Vérification des valeurs doublantes :

```
df.duplicated().sum()
```

```
0
```

Figure 5 : Valeur doublantes

⇨ On remarque qu'il y a aucune valeur doublante dans notre base de données

3. Vérifier les types des colonnes :

```
df.dtypes
```

```
Avg. Area Income              float64
Avg. Area House Age           float64
Avg. Area Number of Rooms     float64
Avg. Area Number of Bedrooms  float64
Area Population               float64
Price                         float64
Address                        object
dtype: object
```

Figure 6 : Type des colonnes

⇨ On remarque que tout les colonnes sont en format numérique ''float64'' , sauf la colonne d'adresse qui est ''objet = String '' car les observations sont déjà des chaine de caractères

⇨ On n'a pas besoin de changer les types des colonnes

4. Vérification des Outliers :

On va vérifier l'existence des outliers dans notre dataset a l'aide des **boxPlots**

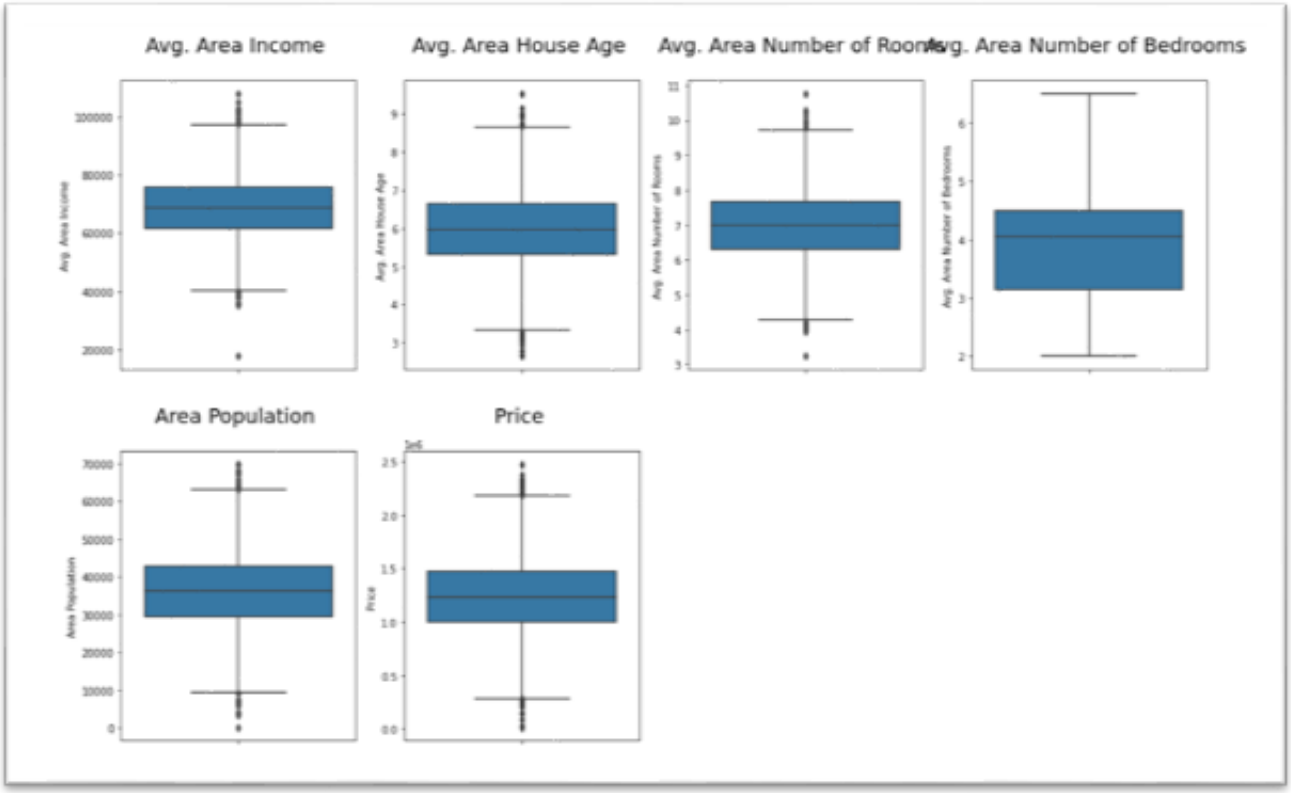


Figure 7 : BoxPlots des colonnes

' En statistique, une donnée aberrante (ou horsain, en anglais outlier) est une valeur ou une observation qui est « distante » des autres observations effectuées sur le même phénomène, c'est-à-dire qu'elle contraste grandement avec les valeurs « normalement » mesurées. ''

- ⇨ On remarque qu'il n'y a pas des données aberrantes trop affectueuse dans les données sauf pour la variable : **'Avg. Area Income'' , '' Avg. Area Number of Bedrooms ''**

- ⇨ On a pas des grands valeurs des outliers , alors on va laisser notre Base de données sont modification

III- Ingénierie des fonctionnalités

1. Dimensions du dataset

```
df.shape
(5000, 7)
```

Figure 8 : Dimensions

⇨ On a 5000 **lignes** et 7 **colonnes** dans notre base de données

⇨ Ça veut dire qu'on a 6 variables explicatives et 1 variable cible et 5000 observations

2. Statistiques récapitulatives

```
df.describe()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

Figure 8 : Statistiques

⇨ La figure ci-dessous montre des statistiques récapitulatives sur les variables explicatives pour bien comprendre la distribution de chaque variable

⇨ elle montre : Nombre d'observation , la moyenne , l'écart type , minimum , Q1 , la median ou Q2 , Q3 et le maximum

3. Suppression des variables inutiles

On va supprimer la colonne " Adress " car nous somme intéressé au variables quantitatives

```
df.drop(['Address'],axis=1,inplace=True)
```

Figure 10 : Suppression d'adresse

IV. Analyse des données et visualisations :

1. Scatter plot ou nuage des points :

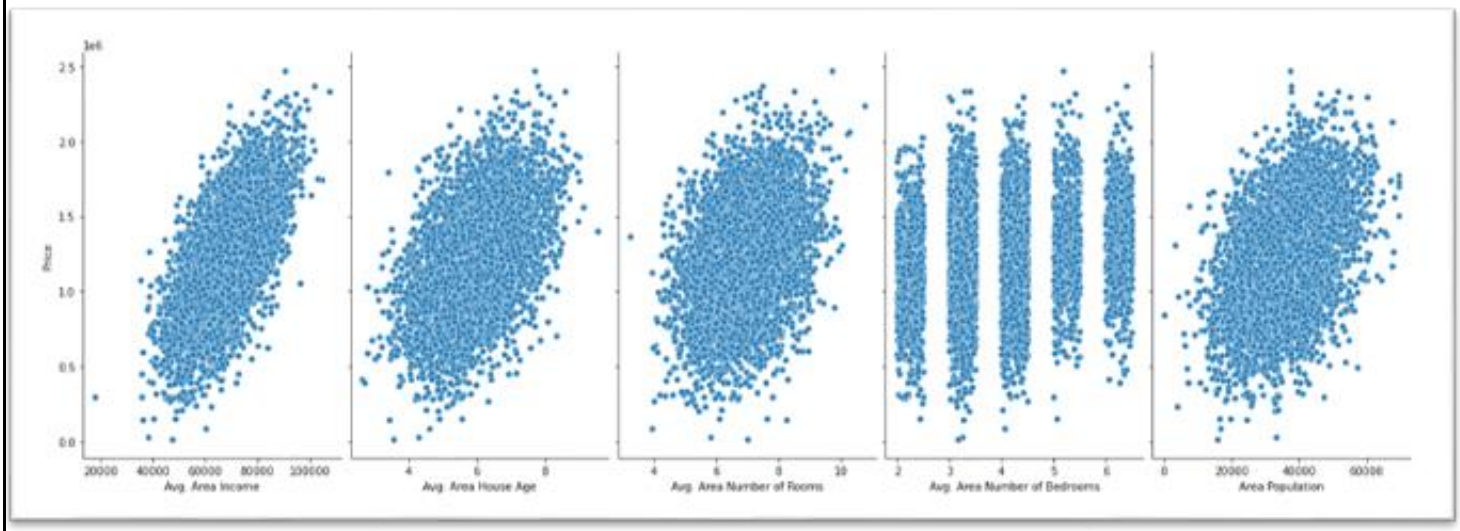


Figure 11 : Scatter plot

⇨ Le nuage des points montre la corrélation entre la variable cible '' Price '' et les variables explicatives

⇨ On remarque que les variables :
'' Avg. Area income ''
'' Avg. Area House Age ''
'' Avg. Areda Number of Rooms ''
'' Area Population ''
Ont une corrélation positive avec la variable cible

S'il y a une correlation positive entre x et y , alors si x augmente y augmente aussi et si x diminue y aussi

- ⇨ On remarque aussi qu'il y a pas une correlation entre ‘’**'Price"** et ‘’ **Avg. Area number of bedrooms"**

2. Heat Map :

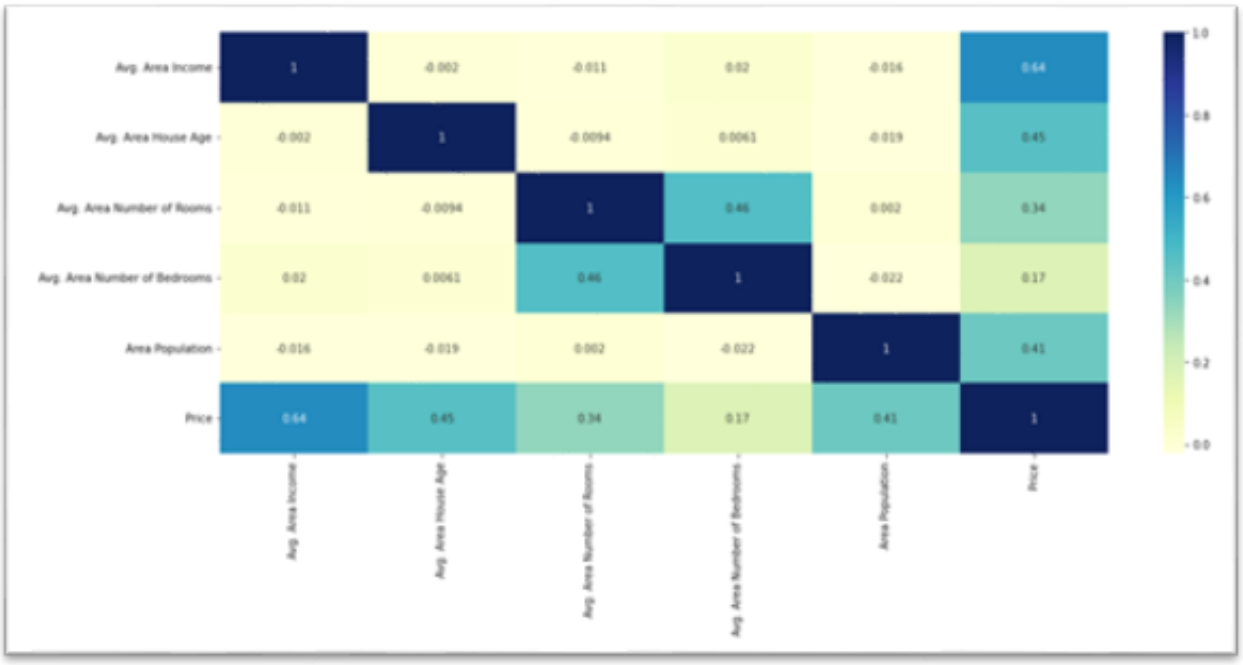


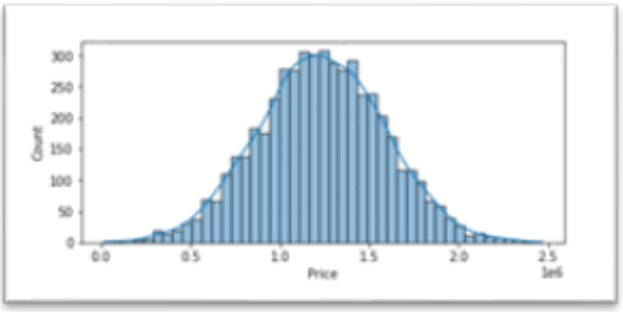
Figure 12 : Heat map

‘ Dans une heatmap, les valeurs présentées peuvent être des nombres compris entre 0 et 1 qui représentent la force de la corrélation entre deux variables. Plus la valeur est proche de 1, plus la corrélation est forte. Inversement, plus la valeur est proche de 0, plus la corrélation est faible."

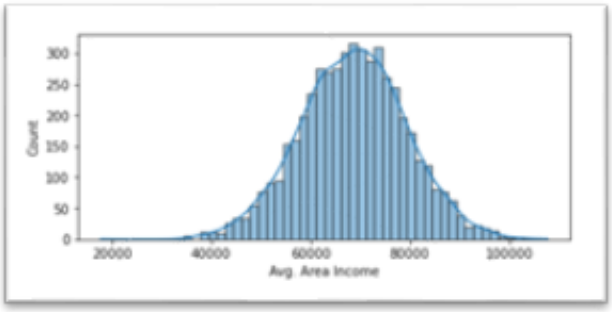
- ⇨ On remarque qu'il y a une forte corrélation entre la variable cible et la variable ‘’ **Avg. Area income ‘’** de **0.64 ,** alors on peut dire que ce variable est le plus significatif dans notre modèle
- ⇨ On remarque qu'il y a pas une corrélation entre ‘’Price’’ et la variable ‘’**Avg. Area Number of Bedrooms** ‘’ comme on a vu dans le nuage des points

3. Histogrammes

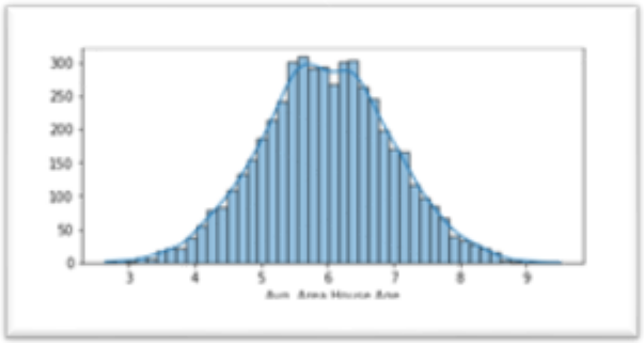
Price



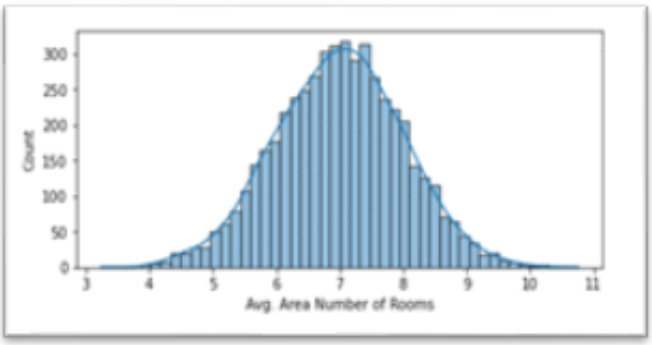
Avg. Area Income



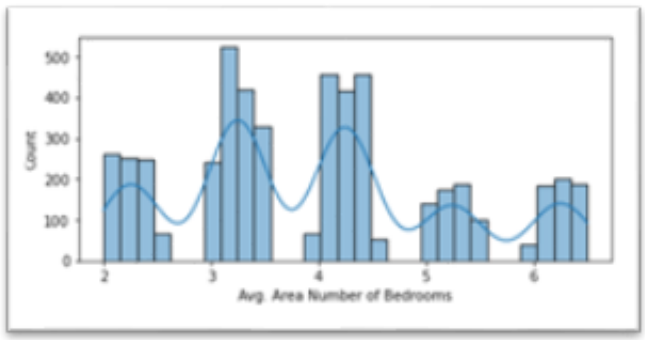
Avg Area House Age



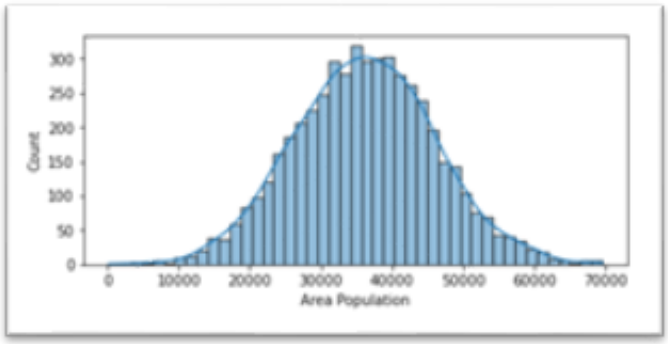
Avg. Area Number of rooms



Avg Area Number of Bedrooms Age



Area Population



‘ Un histogramme est un graphique qui permet de représenter la distribution de fréquence des valeurs d'une variable quantitative ‘’

- ⇨ On remarque que ‘’ Price ‘’ a une distribution normale
- ⇨ On remarque que les variable descriptives ont une distribution presque normale sauf la variable ‘’ **Avg. Area Number of Bedrooms** ‘’

Lorsque la variable cible suit une distribution normale, il est plus facile de comprendre et d'interpréter les résultats du modèle de régression linéaire

V. Création du modèle :

1. Standardisation des données

```
df_train , df_test = train_test_split(df, train_size=0.7 ,test_size = 0.3, random_state=2)

X_train = df_train[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Area Population','Avg. Area Number of
y_train = df_train['Price']

scaler = StandardScaler().fit(X_train)

X_train_stand = scaler.transform(X_train)

X_test = df_test[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Area Population','Avg. Area Number of Be
y_test = df_test['Price']

X_test_stand = scaler.transform(X_test)
```

Figure 13 : Standardisation

⇨ Premièrement, on a commencé par diviser notre dataset en dataset d'entraînement et dataset de test.

⇨ Deuxièmement, définir X_train et y_train , X_test et y_test.

⇨ Troisièmement, standardiser les attributs.

2. Création du modèle :

```
X_train_lm = sm.add_constant(X_train_stand)
lr1 = sm.OLS(y_train, X_train_lm).fit()
```

- Obtenir les paramétres du modéle:

$$f_{w,b}(x) = w_0x_0 + w_1x_1 + \ldots + w_{n-1}x_{n-1} + b$$

```
lr1.params
```

```
const    1.231455e+06
x1       2.301567e+05
x2       1.644490e+05
x3       1.202534e+05
x4       1.514681e+05
x5       2.993028e+03
dtype: float64
```

```
X_test_ = sm.add_constant(X_test_stand)
y_pred = lr1.predict(X_test_)
```

Figure 14 : Création du modèle

⇨ Création du modèle à l'aide de la méthode des moindres carrés ordinaires
⇨ On peut obtenir les paramètres estimés à l'aide de **.params**
⇨ Utiliser notre modèle afin de prédire les prix des maisons du Dataset de test.

3. Calcule des coefficients :

```
from sklearn.metrics import r2_score
R_squared = r2_score(y_test, y_pred)
print('Coefficient de détérmination = %.3f'%R_squared)
R_squared_adjusted = 1-(1-R_squared)*((len(X_test)-1)/(len(X_test)-len(X_test.iloc[0])-1))
print('Coefficient de détérmination ajusté = %.3f'%R_squared_adjusted)
```

```
Coefficient de détérmination = 0.920
Coefficient de détérmination ajusté = 0.920
```

- Calcule de MSE

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
9831074697.740435
```

Figure 15 : Calcule des MSE R^2 R^2a

⇨ Calcul du coefficient de détermination et du coefficient de détermination ajusté et l'erreur quadratique moyenne

MSE = 9831074697.740435
R² = 0.920
R2 adj = 0.920

Interprétation du R² : Notre modèle explique 92% de la dispersion

4. Visualisation des valeurs observées et prédites.

Le graphe ci-dessous représente les indices des observations, les valeurs observées(en bleu) et les valeurs prédites(en rouge) correspondantes.

On remarque que l'erreur entre les valeurs observées et prédites est **petite**.

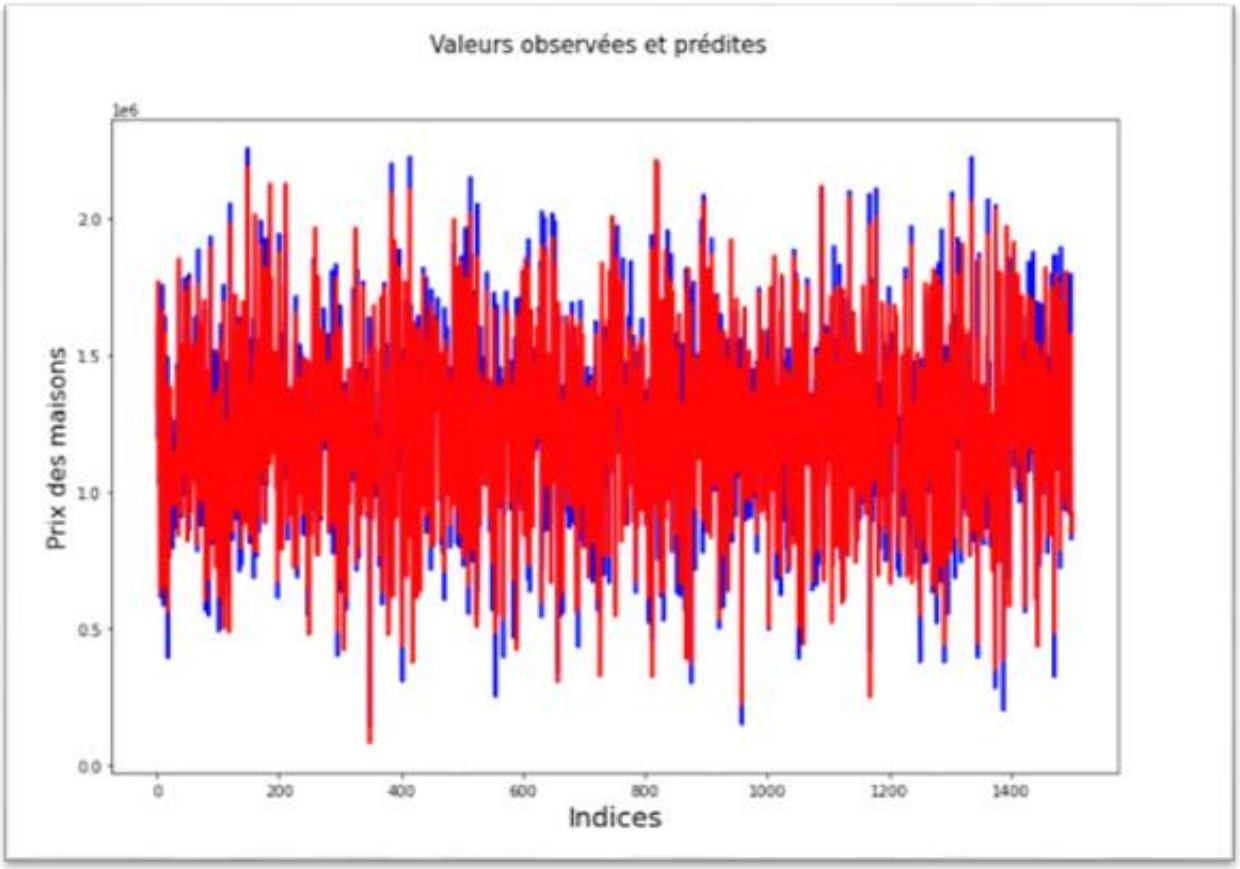


Figure 16 : Visualisation des valeurs prédits

5. Analyse des résidus.

On va visualiser les erreurs :

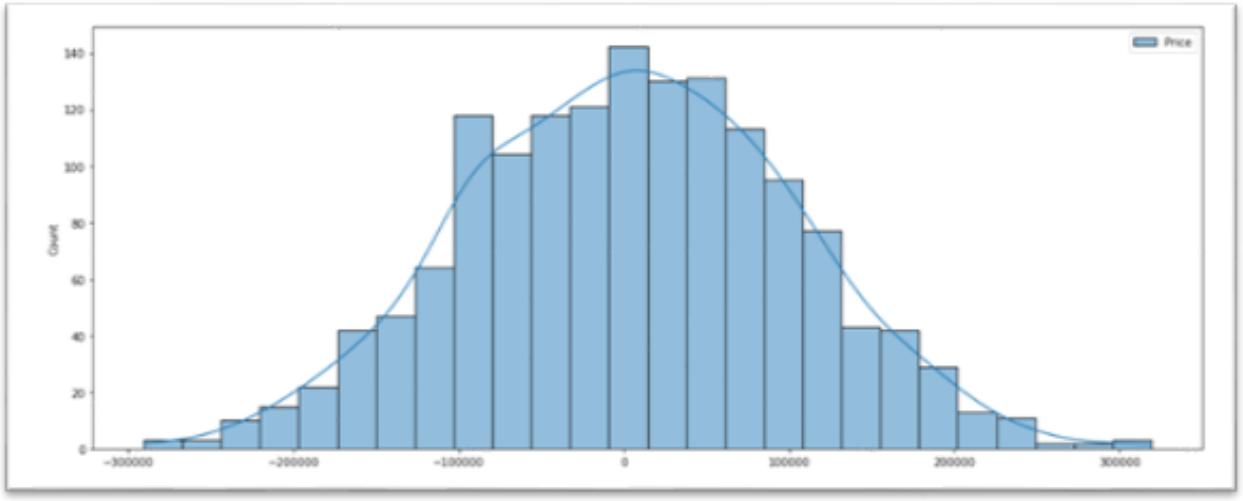


Figure 17 : Histogramme des erreurs

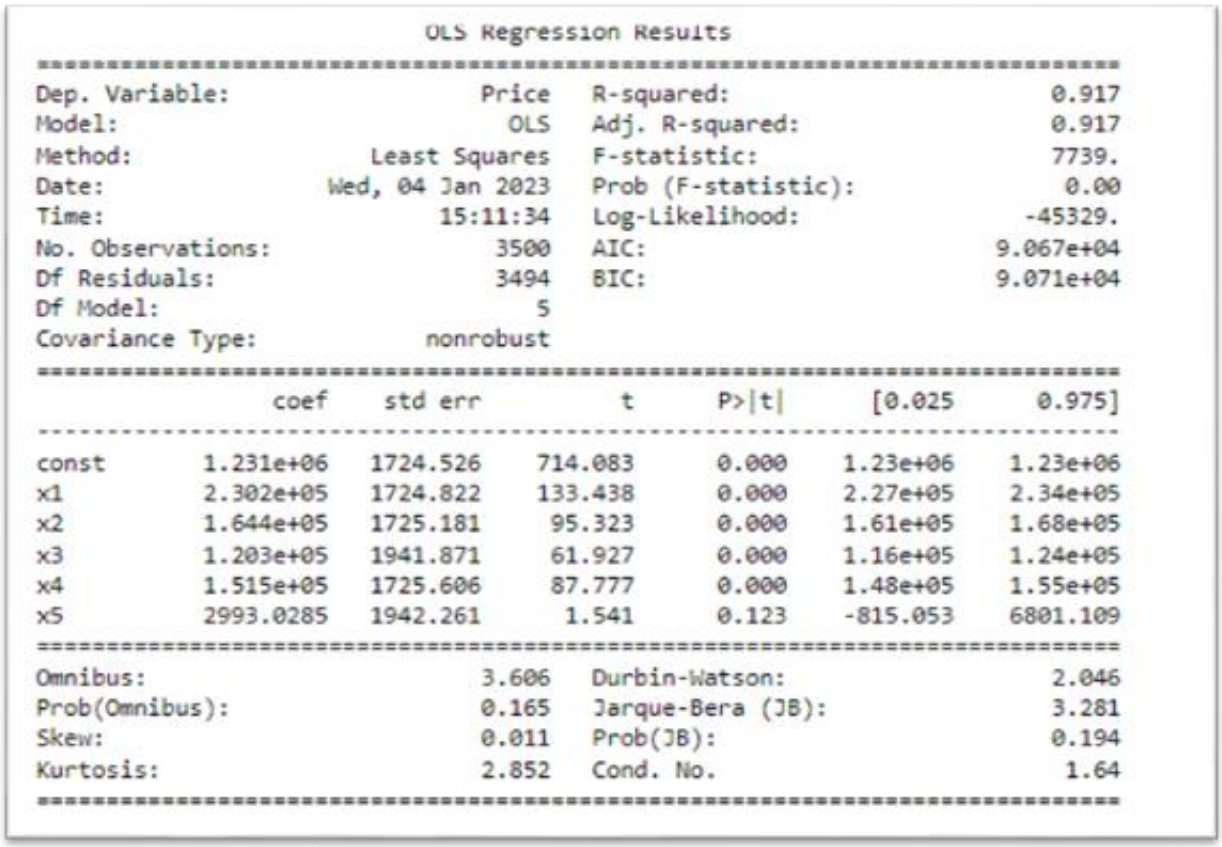
Le modèle linéaire (gaussien) suppose que la variable erreur statistique suit une loi gaussienne. On remarque que les résidus suivent une loi Normale, d'où on peut dire que notre **modèle est adéquat**.

6. Test d'hypothèse

Définition :

Un test d'hypothèse (ou test statistique) est une démarche qui a pour but de fournir une règle de décision permettant, sur la base de résultats d'échantillon, de faire un choix entre deux hypothèses statistiques.

D'après la librairie **Statmodels** :



OLS Regression Results

Dep. Variable:	Price	R-squared:	0.917
Model:	OLS	Adj. R-squared:	0.917
Method:	Least Squares	F-statistic:	7739.
Date:	Wed, 04 Jan 2023	Prob (F-statistic):	0.00
Time:	15:11:34	Log-Likelihood:	-45329.
No. Observations:	3500	AIC:	9.067e+04
Df Residuals:	3494	BIC:	9.071e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.231e+06	1724.526	714.083	0.000	1.23e+06	1.23e+06
x1	2.302e+05	1724.822	133.438	0.000	2.27e+05	2.34e+05
x2	1.644e+05	1725.181	95.323	0.000	1.61e+05	1.68e+05
x3	1.203e+05	1941.871	61.927	0.000	1.16e+05	1.24e+05
x4	1.515e+05	1725.606	87.777	0.000	1.48e+05	1.55e+05
x5	2993.0285	1942.261	1.541	0.123	-815.053	6801.109

Omnibus:	3.606	Durbin-Watson:	2.046
Prob(Omnibus):	0.165	Jarque-Bera (JB):	3.281
Skew:	0.011	Prob(JB):	0.194
Kurtosis:	2.852	Cond. No.	1.64

Figure 18 : Résumé du modèle

- **Test de Student:**
  D'après le résumé statistique, on a la p-value de la t-calculée de x5 " Avg. Area Number of Bedrooms " supérieur à 0.05. Donc on accepte **H0 : B5 = 0** au seuil de 0.05, alors on élimine cet attribut et on réestime le modèle

- **Test Global de Fisher:**
  On remarque que F-calculée vaut : 7739, sa p-value qui vaut 0
  Donc on rejette H0 : Bj = 0 (j=1,..,p) au risque de 0.05, ce qui signifie qu'au moins l'un des paramètres est différent de 0. (ce qui est prouvé par le test de Student, on a tous les paramètres non nuls sauf B5)

VI. Réestimation du modèle :

D'après l'étape de visualisation et les tests d'hypothèse, on va éliminer la variable '' Avg. Area Number of Bedrooms ''

1. Création du modèle :

On trouve les paramètre de nouveau modèle

```
lr2.params

const    1.231455e+06
x1       2.301965e+05
x2       1.644585e+05
x3       1.216284e+05
x4       1.513951e+05
dtype: float64
```

Figure 19 : Paramètres du modèle

2. Calcule la moyenne des moindres carrées :

```
from sklearn.metrics import mean_squared_error
mse1 = mean_squared_error(y_test1, y_pred1)
print(mse1)

9823431323.317665
```

Figure 20 : nouveau MSE

MSE2 = 9823431323.317665

⇨ On remarque que la valeur du MSE a diminué.

3. Analyse des résidus

On va visualiser les nouveaux erreurs

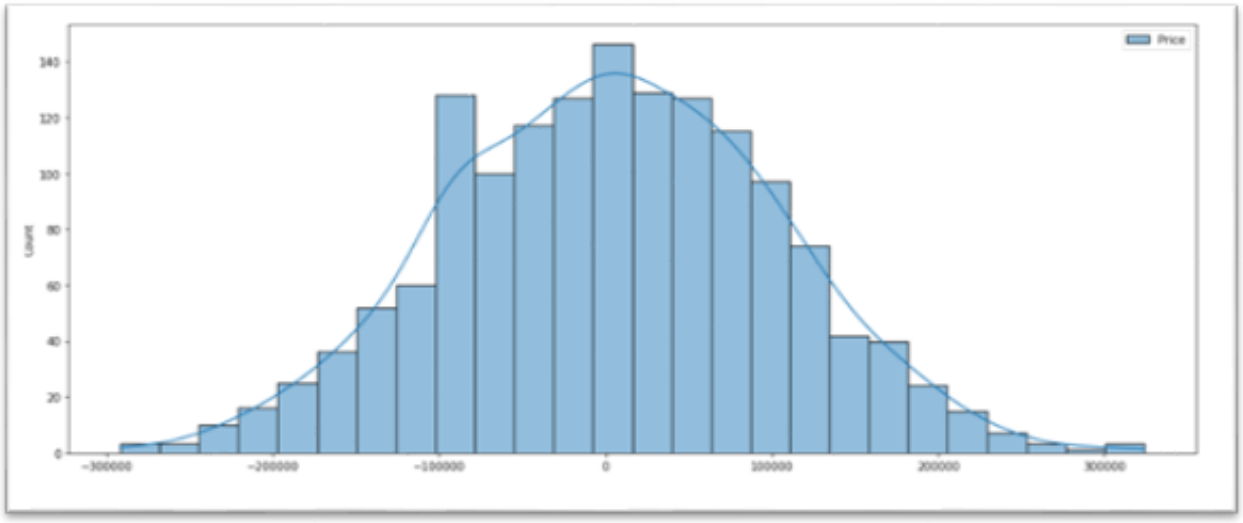


Figure 21 : nouveaux résidus

⇨ On remarque que les nouveaux résidus suivent une loi Normale, d'où on peut dire que notre **modèle est adéquat**.

Alors notre modèle finie est :

Y = 1.231 e+06 + 2.301 e+05 ***x1*** + 1.644 e+05 ***x2*** + 1.2162 e+05 * ***x3*** + 1.5139 e+05 ***x4***

Conclusion

Dans ce rapport on a essayé de créer un modèle de régression linéaire multiple pour prédire le prix de logement

Pour cela on a utilisé une data set des prix de logement d'USA qui contient 7 Colonnes

- **'Avg. Area Income''** **:** revenu moyen de la région
- **''Avg. Area House Age '' :** âge moyen de la maison
- **'' Avg. Area Number of Rooms '' :** nombre moyen de chambres
- **'' Avg. Area Number of Bedrooms '' :** nombre moyen de chambres
- **'' Area Population '' :** Population de la région
- **'' Price '' :** Prix
- **'' Address '' :** Adresse

Pour trouver le modèle adéquat on a passer par les étapes suivant :

1. Chargement des données
2. Nettoyage des données
3. Ingénierie des fonctionnalités
4. Analyse des données et visualisations
5. Création du modèle
6. Réestimation du modèle

Et finalement on a trouvé le modèle suivant :

Y = 1.231 e+06 + 2.301 e+05***x1*** + 1.644 e+05 ***x2*** + 1.2162 e+05 * **x3** + 1.5139 e+05***x4***

FIN.