Rapport du projet de régression linéaire

Réaliser par : Hayder BOUAZIZ, Eya BESBES et Mayssa HEMDANA

Etablissement académique : Faculté des sciences de Tunis - IDS4 (2022/2023)

Sommaire

Liste des figures

1 Introduction

La régression linéaire est un outil statistique utilisé pour établir la relation entre une variable dépendante et une ou plusieurs variables indépendantes. Cette technique est largement utilisée dans les domaines tels que l'analyse financière, l'économétrie, la biostatistique et la recherche opérationnelle pour prévoir les tendances futures et les relations causales. Dans ce rapport, nous allons explorer les différentes méthodes de régression linéaire, leur utilisation et leur pertinence dans divers domaines d'application.

2 Régression linéaire

L'estimation du prix des diamants est cruciale pour le marché de la pierre précieuse. Elle permet aux acheteurs de connaître la valeur réelle d'un diamant avant de l'acheter et aux vendeurs de fixer un prix juste pour leur pierre. Les experts en diamants utilisent des critères tels que la taille, la couleur, la pureté et le poids pour évaluer la valeur d'un diamant. Il est important de noter que la valeur d'un diamant varie considérablement en fonction de ces critères. Ainsi, une estimation précise du prix d'un diamant est nécessaire pour assurer l'équité dans les transactions commerciales et protéger les intérêts des acheteurs et des vendeurs. C'est dans ce contexte que nous cherchons à trouver un modèle linéaire qui pourrait expliquer les prix des diamants.

2.1 Régression linéaire simple

Pour commencer, nous allons essayer de modéliser le problème sous forme linéaire simple. Pour cela, nous prenons comme variable à expliquer le prix d'un diamant (Y) et comme variable explicative le nombre de carats du diamant (X). Nous disposons de 53940 observations.

2.1.1 Rappel mathématique

Notre modèle s'écrit de la manière suivante :

$$Y = \alpha + \beta X + \epsilon$$

Nous cherchons les paramètres $\hat{\alpha}$ et $\hat{\beta}$ minimisant la quantité suivante :

$$\sum_{i=1}^{n} (\epsilon_i)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Ça donne :

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

et

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Pour mesurer la qualité de la régression, nous utilisant le coefficient de détermination:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\epsilon_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

2.1.2 Implémentation en R

On importe et on visualise les données

```
data=read.csv('diamonds.csv')
x=data$carat
y=data$price
df<- data.frame(data$carat,data$price)
```

```
```{r caption="Data frame is now printed using `kable`.",render=lemon_print}
head(df)
```
```

	data.carat <dbl>	data.price <int>
1	0.23	326
2	0.21	326
3	0.23	327
4	0.29	334
5	0.31	335
6	0.24	336

Figure 1: Aperçu des données

On construit le modèle grâce à la méthode lm

```
plot(x,y,col='green',xlab='Carat du diamond',ylab='Prix en USD $')
model<- lm(y~x)
summary(model)
coefficients(model)
abline(model)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-18585.3   -804.8    -18.9    537.4  12731.7

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2256.36      13.06  -172.8   <2e-16 ***
x            7756.43      14.07   551.4   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1549 on 53938 degrees of freedom
Multiple R-squared:  0.8493,    Adjusted R-squared:  0.8493
F-statistic: 3.041e+05 on 1 and 53938 DF,  p-value: < 2.2e-16

(Intercept)           x
  -2256.361    7756.426
```

Figure 2: Résumé

Notre modèle s’écrit de la manière suivante :

$$y = -2256.3 + 7756.4x$$

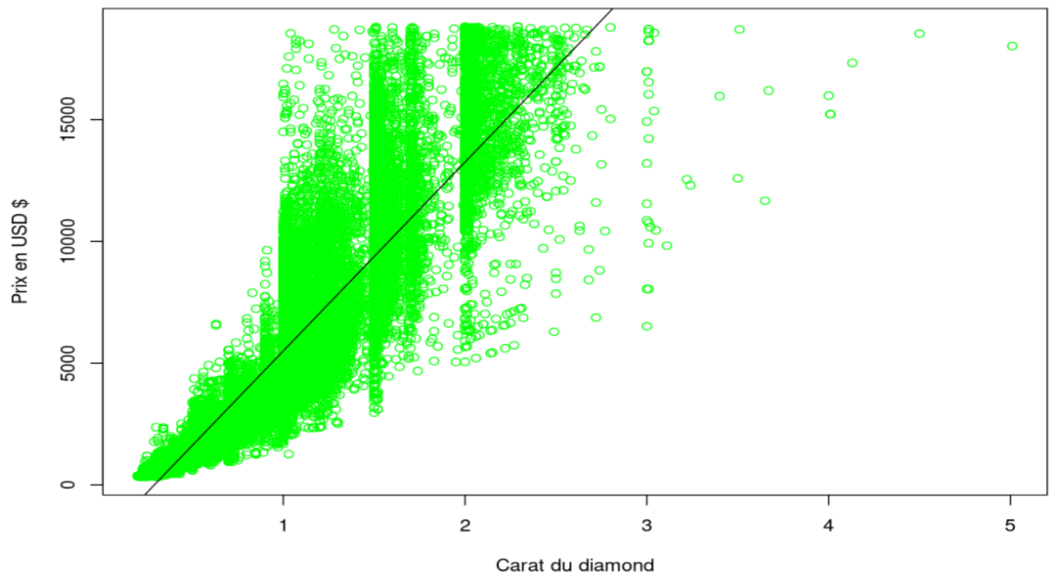


Figure 3: La droite de régression

Nous remarquons que le coefficient de détermination vaut 0.84 ce qui signifie que notre modèle se rapproche de la réalité.

2.1.3 Implémentation en Python

On importe les bibliothèques ainsi que les données.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
d = pd.read_csv("diamonds.csv")
d.describe()
```

	carat	price
count	53940.000000	53940.000000
mean	0.797940	3932.799722
std	0.474011	3989.439738
min	0.200000	326.000000
25%	0.400000	950.000000
50%	0.700000	2401.000000
75%	1.040000	5324.250000
max	5.010000	18823.000000

Figure 4: Aperçu des données

On retrouve bien les mêmes coefficients.

```
[ ] reg=LinearRegression()
    x_train,x_test,y_train,y_test=train_test_split(d[['carat']],d[['price']],test_size=0.3,random_state=1)
    reg.fit(x_train,y_train)
```

```
[ ] alpha=reg.intercept_
    alpha[0]
```

```
-2244.264549822503
```

```
[ ] beta=reg.coef_
    beta[0][0]
```

```
7756.103217498858
```

```
t=np.linspace(0,3,10)
plt.plot(d[['carat']],d[['price']],'g+')
plt.ylabel('Prix en USD $')
plt.xlabel('Carat du diamond')
plt.plot(t,alpha[0]+beta[0][0]*t,color='black')
```

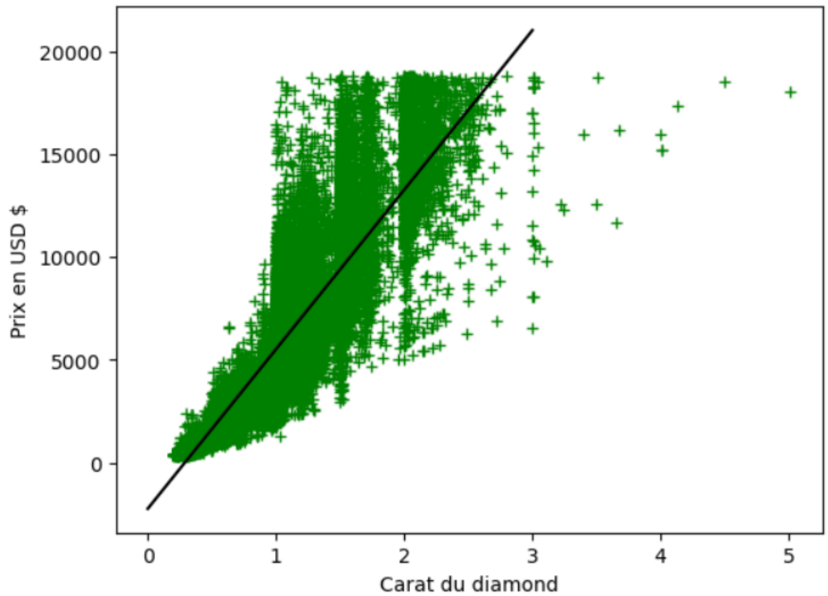
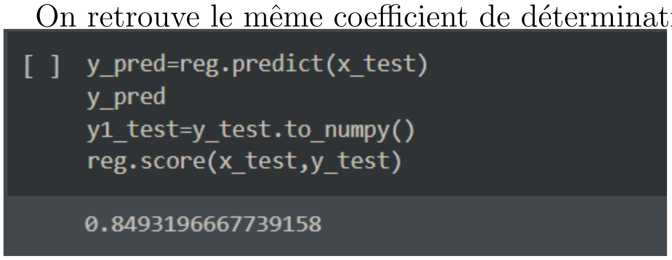


Figure 5: Droite de régression

On retrouve le même coefficient de détermination.



```
[ ] y_pred=reg.predict(x_test)
    y_pred
    y1_test=y_test.to_numpy()
    reg.score(x_test,y_test)

    0.8493196667739158
```

On compare les valeurs réelles de y aux valeurs prédites.

```
[ ] d_= pd.DataFrame({"Valeurs actuelles":y1_test[:,0],"Valeurs prédites":y_pred[:,0]})
    d_
```

	Valeurs actuelles	Valeurs prédites
0	564	315.249512
1	5914	7063.059311
2	2562	2564.519445
3	537	392.810544
4	5964	7063.059311
...	...	...
16177	905	547.932608
16178	3392	4270.862153
16179	802	160.127448
16180	864	237.688480
16181	4142	5511.838668

16182 rows × 2 columns

2.2 Régression linéaire multiple

Pour essayer de se rapprocher encore plus de la réalité on cherche à déterminer le prix à partir des variables suivantes : le nombre de carats du diamant (X1),sa profondeur(X2), sa taille de sa table (X3), sa largeur (X4) , sa longueur(X5) et sa profondeur total (X6).

2.2.1 Rappel mathématique

Notre modèle s'écrit de la manière suivante :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \epsilon$$

Cette fois on trouve les coefficients suivants :

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

2.2.2 Implémentation en R

Construction du modèle grâce à la fonction lm.

```
data_1<- read.csv("diamonds.csv",sep=',')
regression1<-lm(price~carat+depth+table+x+y+z,data=data_1)
summary(regression1)
df<- data.frame(x,y)
df
pairs(df)


acf(residuals(regression1), main="regression1")
confint(regression1)
```

```
Call:
lm(formula = price ~ carat + depth + table + x + y + z, data = data_1)

Residuals:
     Min       1Q   Median       3Q      Max
-23878.2   -615.0    -50.7    347.9  12759.2

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 20849.316    447.562  46.584  < 2e-16 ***
carat       10686.309     63.201 169.085  < 2e-16 ***
depth        -203.154      5.504 -36.910  < 2e-16 ***
table        -102.446      3.084 -33.216  < 2e-16 ***
x           -1315.668     43.070 -30.547  < 2e-16 ***
y              66.322     25.523   2.599  0.00937 **
z              41.628     44.305   0.940  0.34744
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1497 on 53933 degrees of freedom
Multiple R-squared:  0.8592,    Adjusted R-squared:  0.8592
F-statistic: 5.486e+04 on 6 and 53933 DF,  p-value: < 2.2e-16
```

Figure 6: Résumé

	2.5 %	97.5 %
(Intercept)	19972.09167	21726.54115
carat	10562.43500	10810.18317
depth	-213.94191	-192.36620
table	-108.49073	-96.40057
x	-1400.08590	-1231.24978
y	16.29628	116.34693
z	-45.20974	128.46513

Figure 7: Intervalles de confiances

Notre modèle s'écrit de la manière suivante :

$$y = 20849.3 + 10686x_1 - 203.1x_2 - 102.4x_3 - 1315.6x_4 + 66.3x_5 + 41.6x_5$$

et

$$R^2 = 0.8592$$

2.2.3 Implémentation en Python

On importe les bibliothèques et les données.

	Unnamed: 0	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Figure 8: Résumé des données

On visualise la distribution des données.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Using pairplot we'll visualize the data for correlation
sns.pairplot(data, x_vars=['carat', 'depth','table','x','y','z'],
             y_vars='price', size=3, aspect=1, kind='scatter')
plt.show()
```

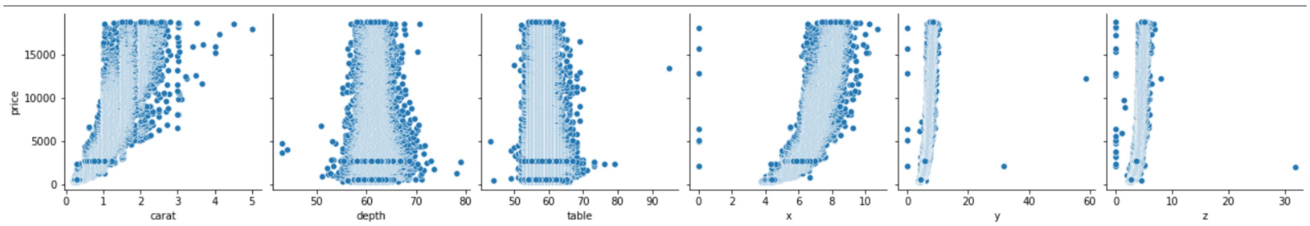


Figure 9: Distribution des données

```
sns.heatmap(data.corr(), cmap="YlGnBu", annot = True)
plt.show()
```

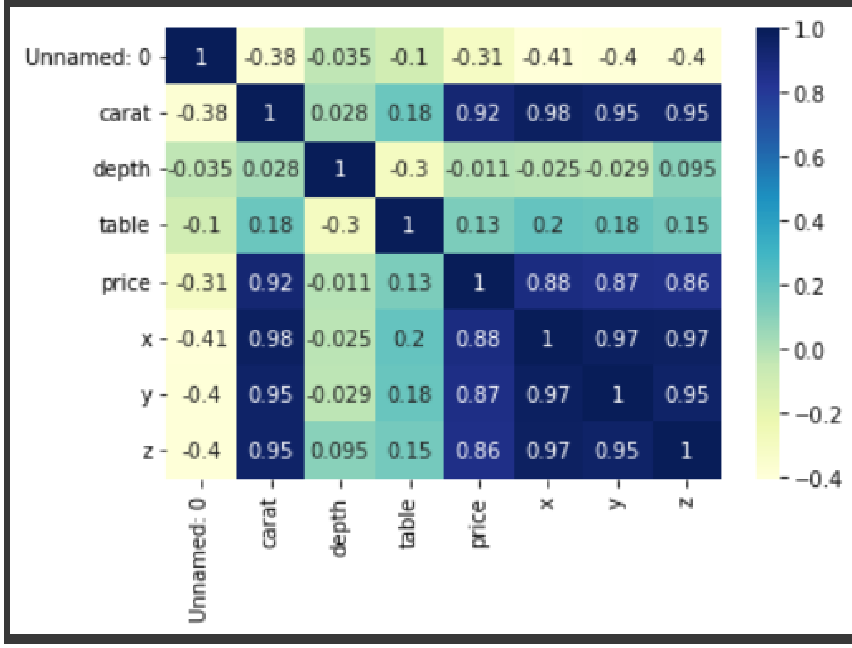


Figure 10: Matrice de corrélation

Construction du modèle grâce à la fonction LinearRegression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=3)
```

```
linreg1=LinearRegression()
```

```
# Entrainer le modèle avec la méthode fit
linreg1.fit(X_train,y_train)
linreg1.score(X_train,y_train)
```

```
0.8596825145349944
```

```
a0=linreg1.intercept_
a0
```

```
21264.40384155851
```

```
[a1,a2,a3,a4,a5,a6]=linreg1.coef_
print(a1,a2,a3,a4,a5,a6)
```

```
10769.828859118188 -205.957028722169 -104.54923160483996 -1310.500193304963 41.92745122265948 20.529401210813962
```

Finalement nous obtenons l'équation suivante :

$$y = 21264.4 + 10769.8x_1 - 205.9x_2 - 104.5x_3 - 1310.5x_4 + 41.9x_5 + 20.5x_5$$

et

$$R^2 = 0.8596$$

3 Régression logistique

Il est crucial de pouvoir prédire si un patient présente un risque de souffrir d'un infarctus, car cela permet de mettre en place des mesures de prévention et de surveillance précoce. Plus tôt un les signes sont détectés, plus les chances d'éviter un infarctus sont élevées. Dans cette perspective, Nous avons décider de construire un modèle de régression logistique afin de savoir si patient présente un risque d'avoir un infarctus. Ce modèle repose les variables explicatives suivantes : le sexe du patient (X1), son âge (X2), son niveau d'éducation (X3), si il est fumeur ou pas (X4), le nombre de cigarette qu'il consomme par jour (X5), s'il est sous traitement médical

(X6), s'il a eu des infarctus auparavant (X7), s'il souffre d'hypertension (X8), s'il est diabétique (X9), son taux de cholestérol (X10), son IMS (X11), son rythme cardiaque (X11), sa glycémie (X12)..

3.1 Rappel mathématique

Notre modèle s'ecrit comme suit :
$log\frac{p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})}{1-p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14}$
Les estimmateurs des ces coefficients s'écrivent de la sorte :
$\beta_j^* = \arg\min_{\beta_j} \sum_{i=1}^{n} [y_i log(p(x_i)) + (1 - y_i) log(1 - p(x_i))]$

3.2 Implémentation en R

Construction du modèle grâce à la fonction glm

```
data<- read.csv("framingham.csv",sep=",",dec=".")

model <- glm( TenYearCHD ~
male+age+education+currentSmoker+cigsPerDay+BPMeds+prevalentStroke+prevalentHyp+diabete
s+totChol+ + BMI + heartRate +glucose , data = data, family = binomial(link=logit))
model$coefficients
summary(model)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8608  -0.6055  -0.4275  -0.2859   2.7420

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     -7.464777   0.651523 -11.457  < 2e-16 ***
male             0.501297   0.107092   4.681 2.85e-06 ***
age              0.070186   0.006402  10.963  < 2e-16 ***
education       -0.060127   0.049021  -1.227 0.219985
currentSmoker    0.072909   0.155794   0.468 0.639797
cigsPerDay       0.018016   0.006208   2.902 0.003704 **
BPMeds           0.311561   0.227887   1.367 0.171571
prevalentStroke  0.673313   0.490305   1.373 0.169674
prevalentHyp     0.607363   0.108295   5.608 2.04e-08 ***
diabetes         0.025114   0.314283   0.080 0.936309
totChol          0.002629   0.001119   2.350 0.018762 *
BMI              0.013466   0.012349   1.090 0.275513
heartRate       -0.001238   0.004165  -0.297 0.766265
glucose          0.007637   0.002224   3.434 0.000594 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3120.5  on 3655  degrees of freedom
Residual deviance: 2776.9  on 3642  degrees of freedom
  (582 observations deleted due to missingness)
AIC: 2804.9

Number of Fisher Scoring iterations: 5
```

Figure 11: résumé

```
                         2.5 %       97.5 %
(Intercept)     -8.7528888425 -6.198035153
male             0.2917046841  0.711677611
age              0.0577098586  0.082816088
education       -0.1570343390  0.035226963
currentSmoker   -0.2346919236  0.376347787
cigsPerDay       0.0058104658  0.030165476
BPMeds          -0.1437707292  0.751745486
prevalentStroke -0.3247775081  1.621716898
prevalentHyp     0.3947771616  0.819448214
diabetes        -0.6092143351  0.626561136
totChol          0.0004260535  0.004814926
BMI             -0.0109003022  0.037540707
heartRate       -0.0094541001  0.006880125
glucose          0.0033467851  0.012082991
```

Figure 12: intervalles de confiances

Notre modèle s'écrit donc comme suit :
$log\frac{p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})}{1-p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})} = -7.4+0.5X_1+0.07X_2-0.06X_3 0.07X_4+ 0.01X_5 + 0.3X_6 0.6X_7 0.6X_8 + 0.02X_9 + 0.002X_{10} + 0.01X_{11} - 0.001X_{12} 0.007X_{13} - 0.016X_{14}$

3.3 Implémentation en Python

Importation des bibliothèques et des données

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
data = pd.read_csv("framingham.csv")
data.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYe
0	1	39	4	0	0	0	0	0	0	195	106.0	70.0	26.97	80	77	
1	0	46	2	0	0	0	0	0	0	250	121.0	81.0	28.73	95	76	
2	1	48	1	1	20	0	0	0	0	245	127.5	80.0	25.34	75	70	
3	0	61	3	1	30	0	0	1	0	225	150.0	95.0	28.58	65	103	
4	0	46	3	1	23	0	0	0	0	285	130.0	84.0	23.10	85	85	

Figure 13: Aperçu des données

Construction du modèle grâce a la fonction LogisticRegression

```
x = data [['male','age','education','currentSmoker','cigsPerDay','BPMeds','prevalentStroke','prevalentHyp','diabetes','totChol','sysBP','dia
y = data['TenYearCHD']
```

```
modele_regLog = LogisticRegression(random_state = 0,
solver = 'liblinear', multi_class = 'auto')
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 0,test_size=0.8)

modele_regLog.fit(x_train,y_train)
```

```
LogisticRegression(random_state=0, solver='liblinear')
```

```
precision = modele_regLog.score(x_test,y_test)
print(precision)
```

```
0.8441025641025641
```

```
modele_regLog.coef_
```

```
array([[-1.33910632e-01,  4.96462874e-02, -1.45095783e-01,
        -3.05623260e-01,  2.36160175e-02,  9.99727602e-01,
        -4.96195398e-01,  5.94287578e-01,  2.89169481e-01,
         2.27936191e-04,  5.97437917e-03, -2.12426865e-02,
        -2.00300497e-02, -1.64466652e-02,  6.68172984e-03]])
```

Notre modèle s'écrit donc comme suit :
$log\frac{p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})}{1-p(Y=1|X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10},X_{11},X_{12},X_{13},X_{14})} = -2.1-0.3X_1+0.04X_2-0.14X_3-0.3X_4+0.02X_5+0.99X_6-0.49X_7+0.59X_8+0.28X_9+0.0002X_{10}+0.005X_{11}-0.02X_{12}-0.02X_{13}-0.016X_{14}$
et

$$R^2 = 0.84$$

```
confusion_matrix(y_test, modele_regLog.predict(x_test))
```

```
array([[2444,   22],
       [ 434,   25]])
```

Nous avons eu 2444 vrai négatives et 25 vrai positives

4 Conclusion

En conclusion, les techniques de régression logistique et linéaire sont des outils importants pour modéliser les relations entre des variables cibles et explicatives dans les études statistiques. Ces techniques permettent de décrire les tendances générales et de prévoir les valeurs de la variable cible en fonction des valeurs des variables explicatives. Ces outils sont particulièrement utiles pour les situations où la variable cible est dichotomique ou quantitative et où il y a une ou plusieurs variables explicatives.