Université de Nouakchott Al-Aasriya
Faculté des Sciences et Techniques
Département de Mathématique-Informatique
Filière Statistique et Économetrie

Optimisation des erreurs du modèle de prédiction du prix des mobiles

Rapport présenté par :
Aichetou Maatalla Maatalla :C14690
Aicha Mohamed Bou Houbeini :C13763
Bara Babah Ahmed Babou :C14676

Sous la direction de :
Dr.Mohamed Mahmoud El Benany

Année universitaire :2022-2023

Table des matières

1 Introduction **4**

2 Les Technologies utilisée **4**

3 Objective de l'optimisation **4**

4 Présentation du données utilisées **6**

5 Analyse du donnée **7**
5.1 Visualisation de variables 7
5.2 Relation entre les variables 10

6 Prédiction et optimisation du donnée **12**
6.1 Le modèle 12
6.2 La fonction coût 13
6.3 La descente de gradient 14
6.4 Représentation du fonction coût 18
6.5 Vérification du résultat de prédiction 19
6.6 Améliorer le taux d'apprentissage en mettant à l'échelle les données 20
6.7 Vérification du résultat de prédiction après l'amélioration 21
6.8 Représentation du fonction coût après l'amélioration 22
6.9 Représentation du prix prédictive avec prix réel 23
6.10 Représentation du prix prédictive avec prix réel en utilisant la fonction SGDRegressor 24

7 Conclusion **25**

Liste des figures

1 Le données. 5
2 La data-set. 6
3 Les RAM du mobile. 7
4 La mémoire interne. 8
5 Performance du batterie. 9
6 Les corrélations entres variables. 10
7 La relation entre variable. 11
8 La Fonction cout. 13
9 La résultat du Descente de Gradient. 15
10 La descente de gradient. 16
11 La Gradient descente. 17
12 Les Eurrers. 18
13 Résultat du prédiction. 19
14 Amélioration. 20
15 Résultat du prédiction après améliore. 21
16 Fonction coût après l'amélioration. 22
17 Résultat du prédiction après améliore. 23
18 Fonction cout après l'amélioration. 24

1 Introduction

Le gradient descente est un algorithme d'optimisation qui permet de calculer le minimum local et global d'une fonction (convexe) en changeant au fur et à mesure (itérations) les paramètres de cette fonction.
En d'autres termes, le gradient descente est un algorithme permettant de trouver le minimum local et global d'une fonction différentiable. La descente de gradient est simplement utilisée pour trouver des valeurs aux paramètres d'une fonction permettant d'atteindre ce minimum local et minimum global.

2 Les Technologies utilisée

Technologies utilisées dans ce tutoriel sont :

- **Python (langage de programmation de haut niveau)**
- **Numpy (bibliothèque Python numérique)**
- **Pandas (bibliothèque Python d'analyse et de manipulation de données)**
- **Matplotlib (bibliothèque de visualisation de données Python)**
- **Seaborn (bibliothèque avancée de visualisation de données Python)**
- **Scikit-learn (bibliothèque d'apprentissage automatique Python)**
- **Bloc- notes Jupyter (environnement de développement intégré)**

3 Objective de l'optimisation

Dans ce rapport on va essayer de créer un modèle de prédiction, pour prédire les prix de téléphones et d'optimiser les erreurs pour avoir un meilleur modèle.
Le prix du mobile dépend de divers facteurs tels que la résolution, la RAM, la batterie et la mémoire interne ...
Dans cet ensemble de données, nous souhaitons estimer le prix des téléphones mobiles utilisant ces caractéristiques. Notre donnée contient 14 variables et 161 observations.

data

	Product_id	Price	Sale	weight	resoloution	ppi	cpu core	cpu freq	internal mem	ram	RearCam	Front_Cam	battery	thickness
0	203	2357	10	135.0	5.20	424	8	1.350	16.0	3.000	13.00	8.0	2610	7.4
1	880	1749	10	125.0	4.00	233	2	1.300	4.0	1.000	3.15	0.0	1700	9.9
2	40	1916	10	110.0	4.70	312	4	1.200	8.0	1.500	13.00	5.0	2000	7.6
3	99	1315	11	118.5	4.00	233	2	1.300	4.0	0.512	3.15	0.0	1400	11.0
4	880	1749	11	125.0	4.00	233	2	1.300	4.0	1.000	3.15	0.0	1700	9.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
156	1206	3551	4638	178.0	5.46	538	4	1.875	128.0	6.000	12.00	16.0	4080	8.4
157	1296	3211	8016	170.0	5.50	534	4	1.975	128.0	6.000	20.00	8.0	3400	7.9
158	856	3260	8809	150.0	5.50	401	8	2.200	64.0	4.000	20.00	20.0	3000	6.8
159	1296	3211	8946	170.0	5.50	534	4	1.975	128.0	6.000	20.00	8.0	3400	7.9
160	1131	2536	9807	202.0	6.00	367	8	1.500	16.0	3.000	21.50	16.0	2700	8.4

161 rows × 14 columns

FIGURE 1 – Le données.

On va travailler sur quatre variables seulement, qui sont les plus importants en mobile. Pour cela, on supprime tous les variables sauf les variables RAM, Prix, la batterie et la mémoire interne.

4 Présentation du données utilisées

```
df.head()
```

	Price	internal mem	ram	battery
0	2357	16	3	2610
1	1749	4	1	1700
2	1916	8	1	2000
3	1315	4	0	1400
4	1749	4	1	1700

FIGURE 2 – La data-set.

5 Analyse du donnée

5.1 Visualisation de variables

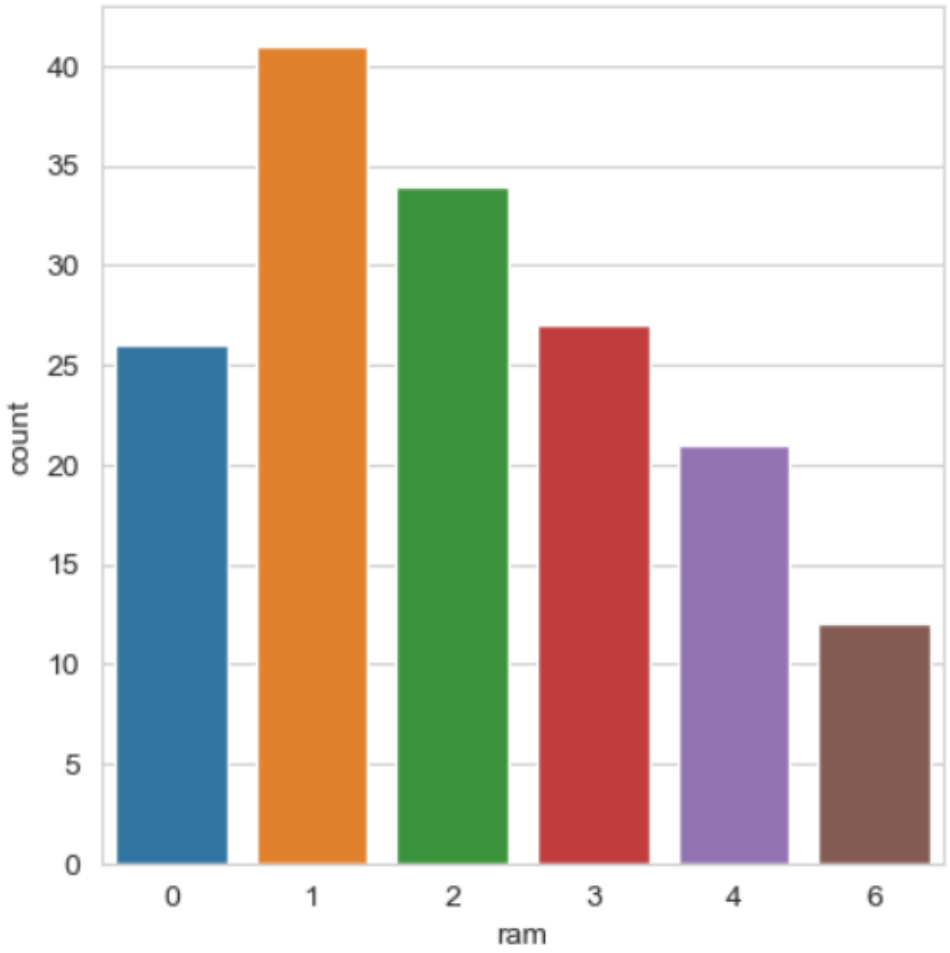

FIGURE 3 – Les RAM du mobile.

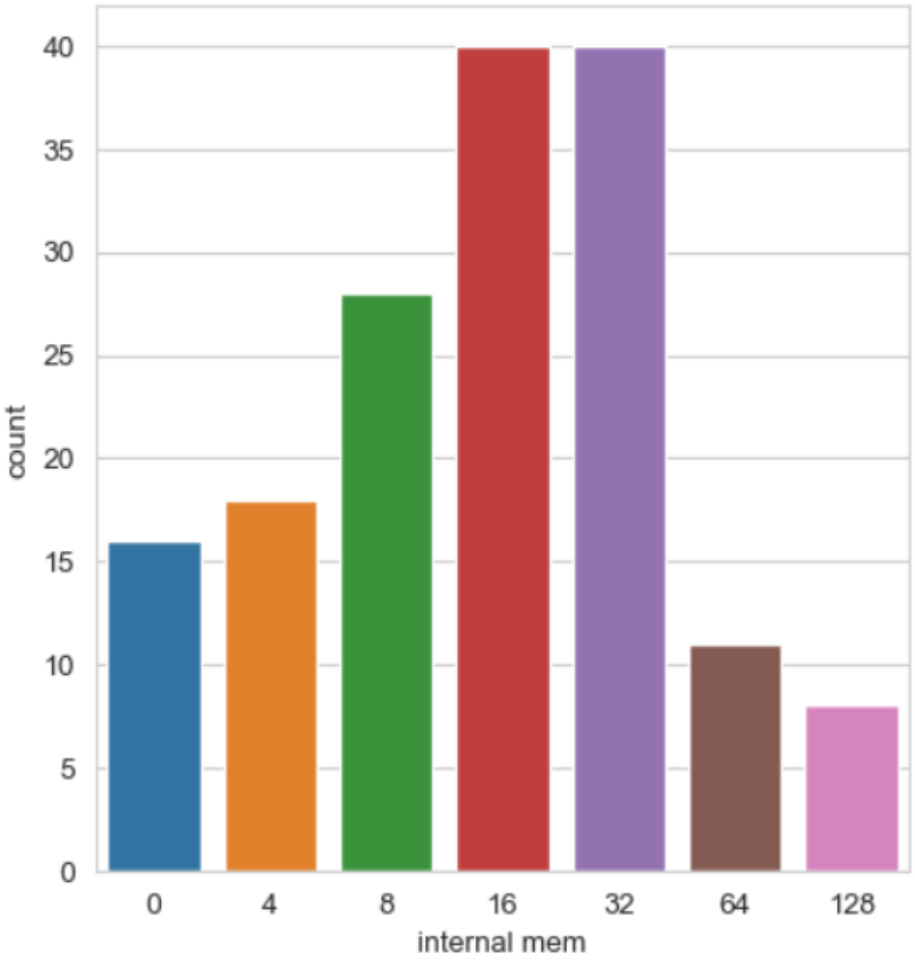

FIGURE 4 – La mémoire interne.

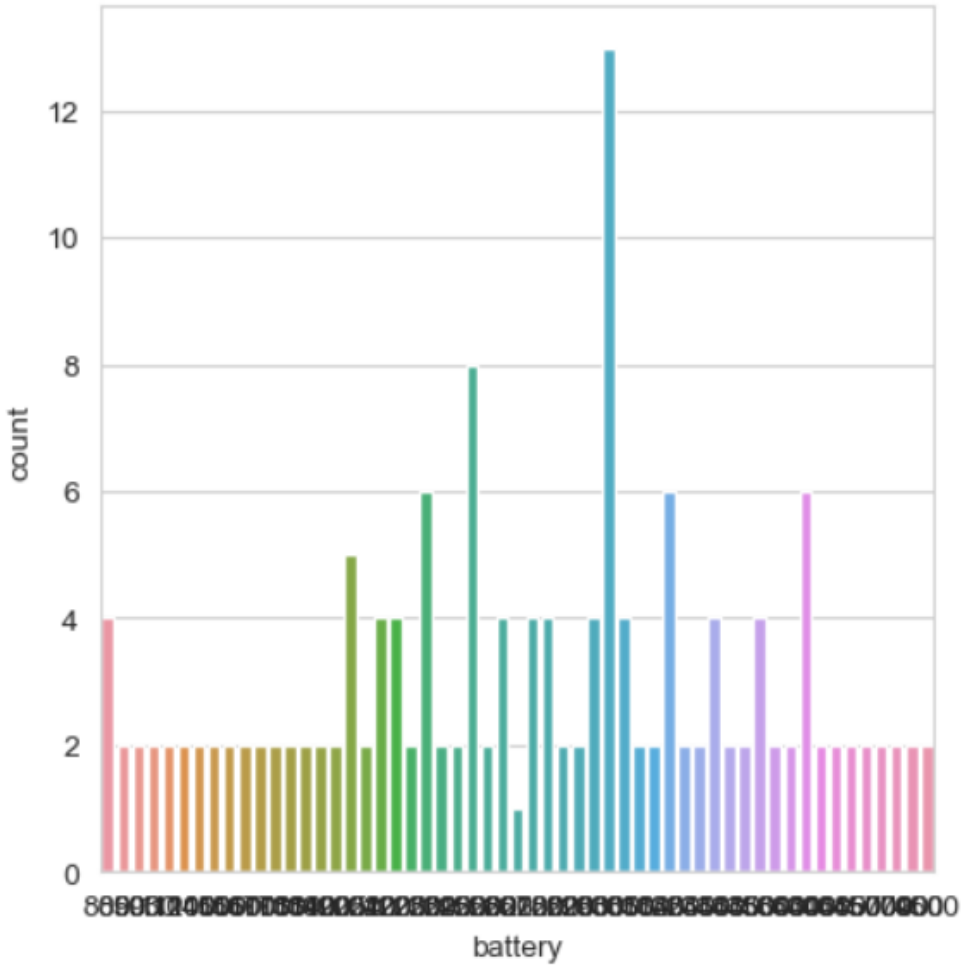

FIGURE 5 – Performance du batterie.

Interprétation :
Dans les graphes ci-dessus, on peut voir la quantité et la variété de chaque variable.

5.2 Relation entre les variables

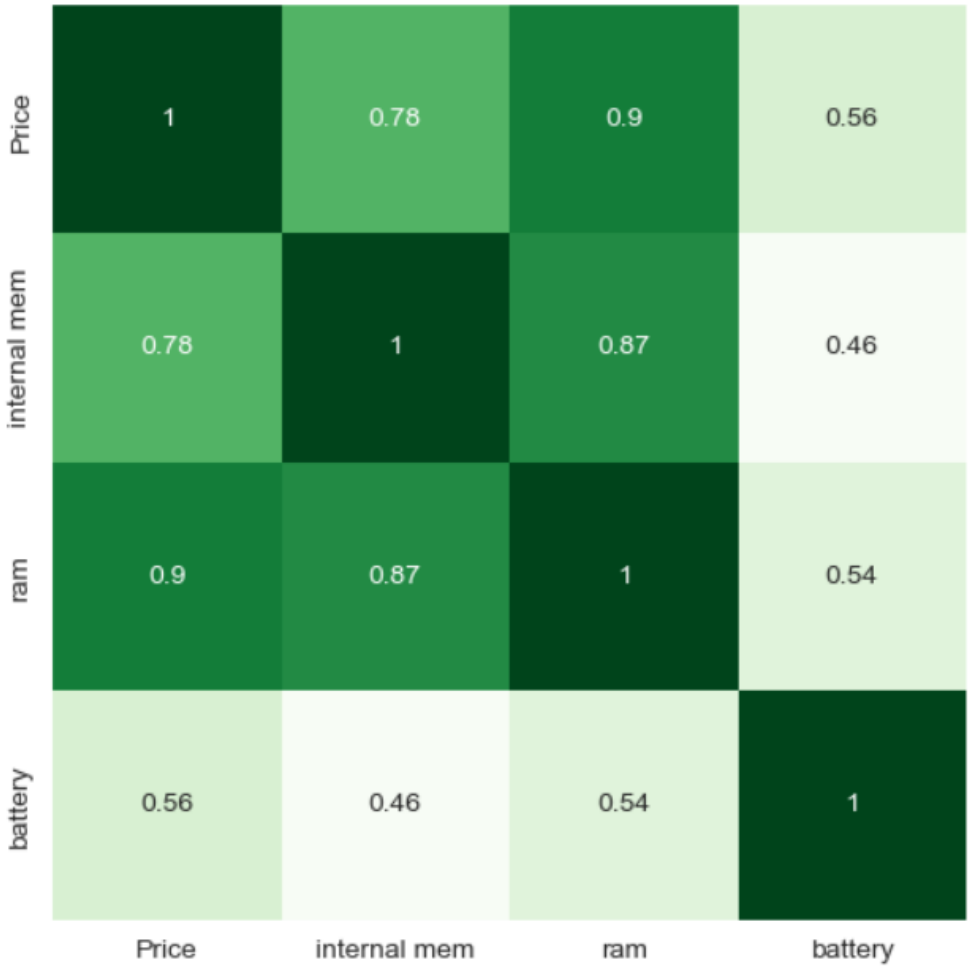

FIGURE 6 – Les corrélations entres variables.

Interprétation :
La représentation du corrélation entre les variables, montre que les corrélations entres les variables et le prix est fort, de plus les corrélations des variables entre eux est forte donc ces variables sont bien influencée entre eux.

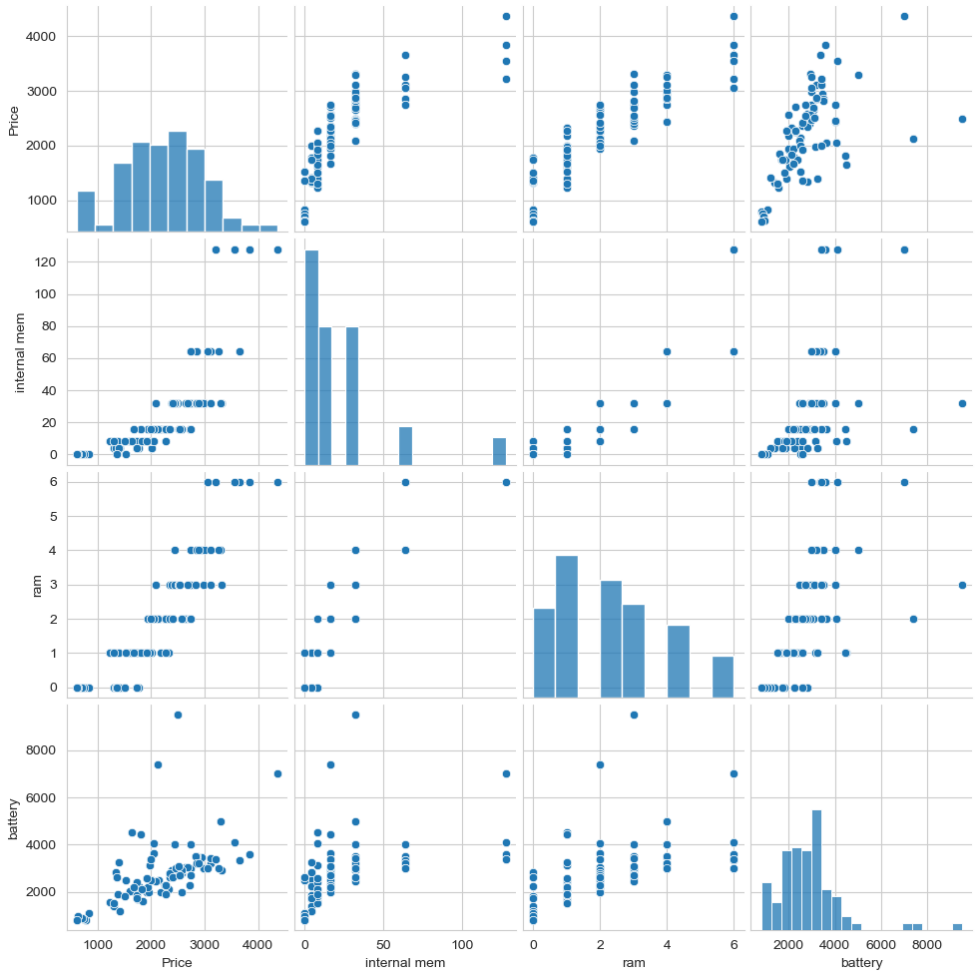

FIGURE 7 – La relation entre variable.

Interprétation :
La représentation ci-dessus montre que la relation entre les variables est linéairement fortement positive, ce qui implique qu'ils sont bien influencés le prix.

6 Prédiction et optimisation du donnée

6.1 Le modèle

D'après les résultats de l'analyse ci-dessus, on peut dire que les variables RAM, Mémoire interne et Batterie sont plus important pour prédire le prix du mobile donc on peut écrire le tableau de données précédentes sous la forme linéaire :

$$F(x) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

x_1 : Mémoire interne.
x_2 : RAM.
x_3 : Batterie.
ω_0 : Le biais.
ω_1, ω_2, et ω_3 : Sont les poids associées a chaque variable.

On va appliquer l'algorithme de régression linéaire pour la prédiction du prix de mobile.
Et on suit les étapes ci-dessous pour atteindre notre but.

6.2 La fonction coût

Reprenons la fonction de coût pour une droite détaillée dans l'introduction à la régression linéaire. La sortie de cette fonction représente les erreurs,et c'est les erreurs que l'on tentera de minimiser : la moyenne de la distance verticale entre nos observations et la droite de régression au carrée (voir la méthode des moindres carrées). Afin de garder une erreur de taille raisonnable et qui ne sera pas affectée par la taille de notre jeu de données, nous divisions par le nombre d'observations pour récupérer la moyenne des erreurs.

```
b_init = 0
w_init = np.array([0.0,0.0,0.0])

def cost_function(x,y,b,w):
    m = x.shape[0]
    error = 0.0
    for i in range(m):
        fwb = np.dot(x[i],w) + b
        error = error + (fwb - y[i])**2
    cost = error / (2*m)
    return cost

print(cost_function(x_train,y_train,b_init,w_init))
```

FIGURE 8 – La Fonction cout.

En appliquant la fonction du coût ci-dessus, elle donne la valeur 2731877.2757009347. C'est le coût associé à la ligne qui coïncide avec l'axe des abscisses.
Cette droite n'est pas la droite de régression ajustée.

6.3 La descente de gradient

Le gradient (la pente de notre fonction de coût à un point donné) représente la direction et le taux de variation de notre fonction de coût. Suivre le gradient négatif de la fonction nous permet donc de la minimiser le plus rapidement possible. Afin d'obtenir le gradient, notre fonction doit être différentiable. Prendre le carré de la distance nous assure une erreur positive et ainsi une fonction différentiable.
Tout d'abord, nous définition une fonction appelée Gradient-Descente qui prend les entrées x, y, w, b.

La fonction du gradient descente ci-dessous nous donne les valeurs des poids(w) ci-dessus est :
biais : -2215.411214953271
Poids : [-6.79798879e+04 -5.83597196e+03 -6.69438159e+06]

Puis nous définition une fonction appelée gradient-descente, qui prend les entrées x, y, poids initial, biais initial, fonction coût, Gradient-Descente, alpha et numéro d'itérations, qui donne le résultat ci-dessous

```
iteration: 0  Cost: 2687267.61148331
iteration: 1000  Cost: 168028214.67014474
iteration: 2000  Cost: 752850902.5405685
iteration: 3000  Cost: 1757155331.2229517
iteration: 4000  Cost: 3180941500.7174025
iteration: 5000  Cost: 5024209411.023836
iteration: 6000  Cost: 7286959062.142251
iteration: 7000  Cost: 9969190454.072659
iteration: 8000  Cost: 13070903586.81505
iteration: 9000  Cost: 16592098460.369417

 b,w found by gradient descent:
bias: 0.022154112149533695
weights:: [6.79798879e-01 5.83597196e-02 6.69438159e+01]
```

FIGURE 9 – La résultat du Descente de Gradient.

```python
def Gradient_Descent(x,y,w,b):
    m,n = x.shape
    dw = np.zeros(n)
    db = 0
    for i in range(m):
        Fwb = np.dot(x[i],w) + b
        error = Fwb - y[i]

        for j in range(n):
            dw[j] = dw[j] + error * x[i,j]

        db = db + error

    dw = dw / m
    db = db / m

    return dw , db

w , b = Gradient_Descent(x_train,y_train,w_init,b_init)
print('bias:',b)
print('weights:',w)
```

FIGURE 10 – La descente de gradient.

```python
def gradient_descent(x, y, w_in, b_in, cost_function, Gradient_Descent, alpha, num_iters):
    history = []
    b = b_in
    w = copy.deepcopy(w_in)

    for i in range(num_iters):
        d_w,d_b = Gradient_Descent(x, y, w_in, b_in)

        w = w - alpha * d_w
        b = b - alpha * d_b

        if i<100000:
            history.append(cost_function(x, y, b, w))

        if i% math.ceil(num_iters / 10) == 0:
            print('iteration:',i,' Cost:',history[-1])

    return w, b, history

w_final, b_final, J_hist = gradient_descent(x_train,y_train,w_init,b_init,cost_function, Gradient_Descent, 1e-9,10000)
print("\n b,w found by gradient descent:")
print('bias:',b_final)
print('weights::',w_final)
```

FIGURE 11 – La Gradient descente.

6.4 Représentation du fonction coût

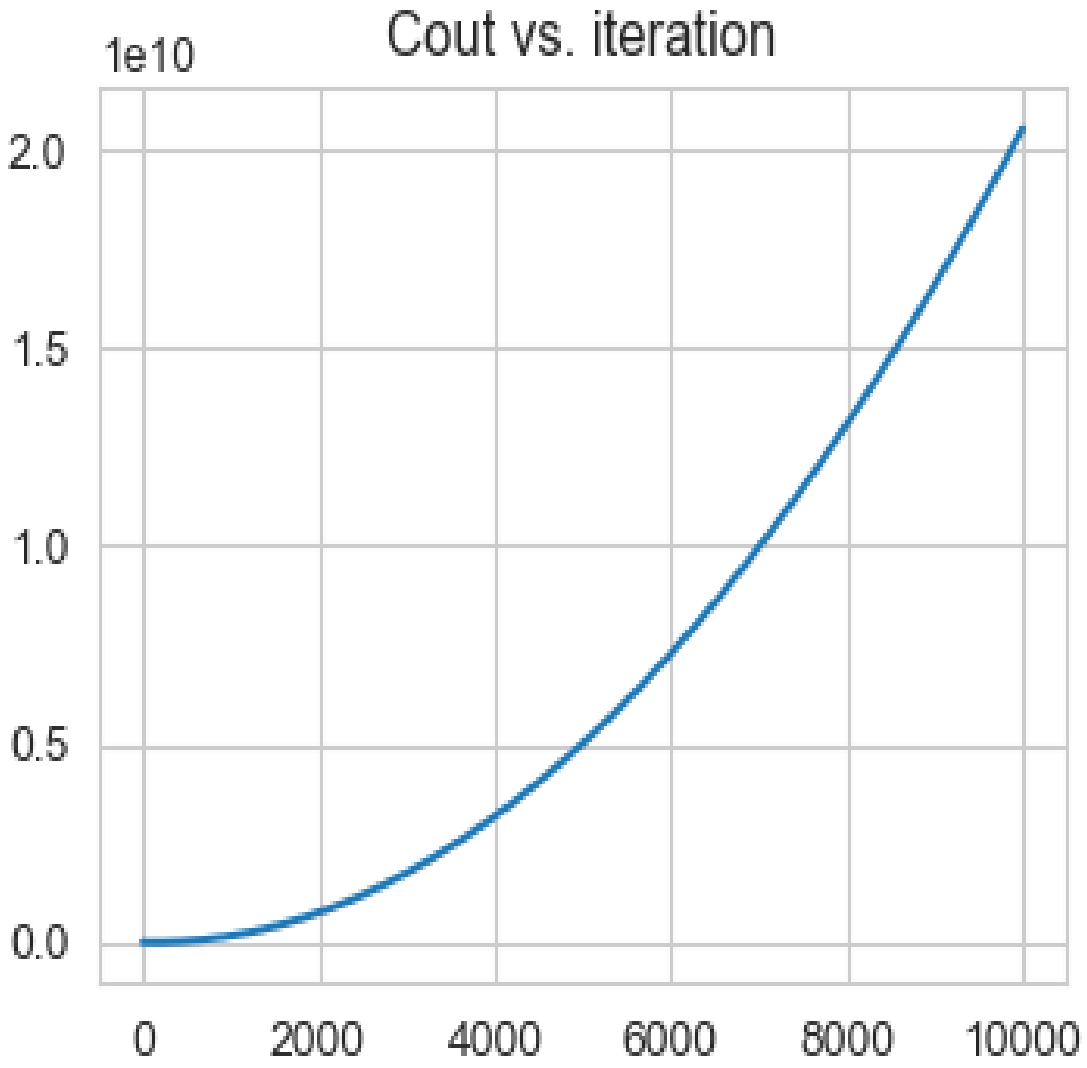

FIGURE 12 – Les Eurrers.

Interprétation :
La représentation du fonction coût montre que la fonction est positive et croissante, donc les erreurs sont augmente.

6.5 Vérification du résultat de prédiction

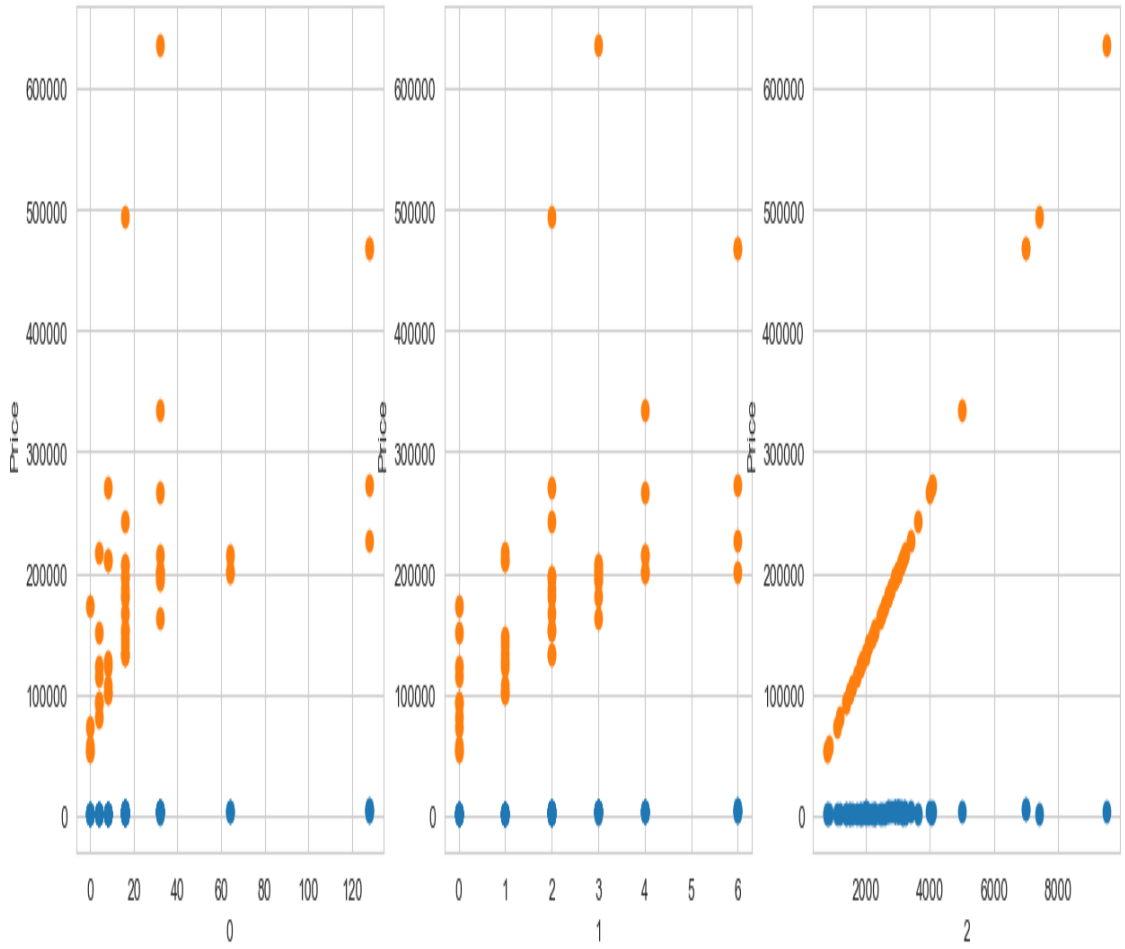

FIGURE 13 – Résultat du prédiction.

Interprétation :
La représentation du prédiction a plusieurs des valeurs aberrantes, s'implique qu'ils sont mal prédits donc cette prédiction ne pas fonction bien.

6.6 Améliorer le taux d'apprentissage en mettant à l'échelle les données

```
def standardization(col):
    mean = x_train[col].mean()
    std = x_train[col].std()
    if (std != 0):
        x_train[col] = x_train[col].map(lambda p : (p-mean)/std)

x_train = pd.DataFrame(x_train)
for i in x_train.columns:
    standardization(i)

x_test = pd.DataFrame(x_test)
for i in x_test.columns:
    standardization(i)
```

FIGURE 14 – Amélioration.

6.7 Vérification du résultat de prédiction après l'amélioration

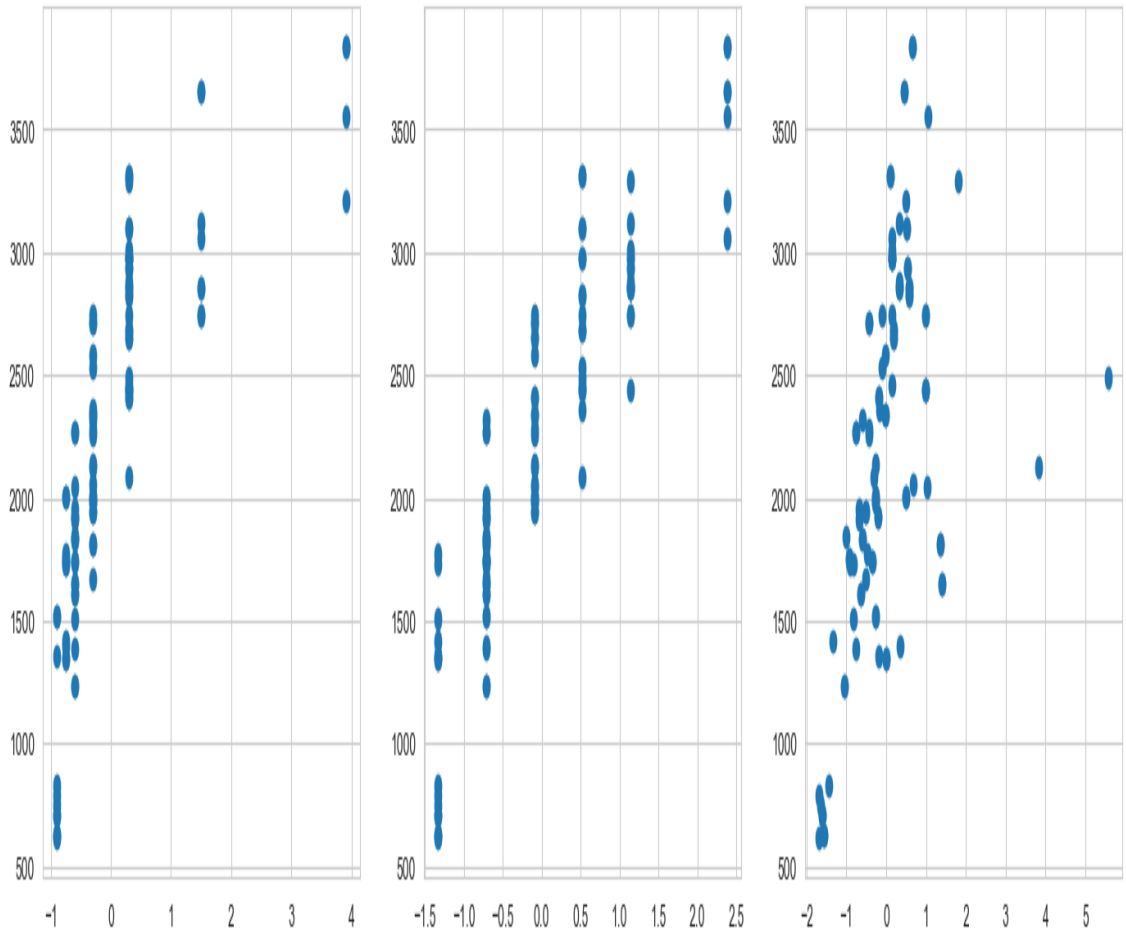

FIGURE 15 – Résultat du prédiction après améliore.

Interprétation :
On voit maintenant que la relation entre Prédiction et les variables sont linéaires positivement.

6.8 Représentation du fonction coût après l'amélioration

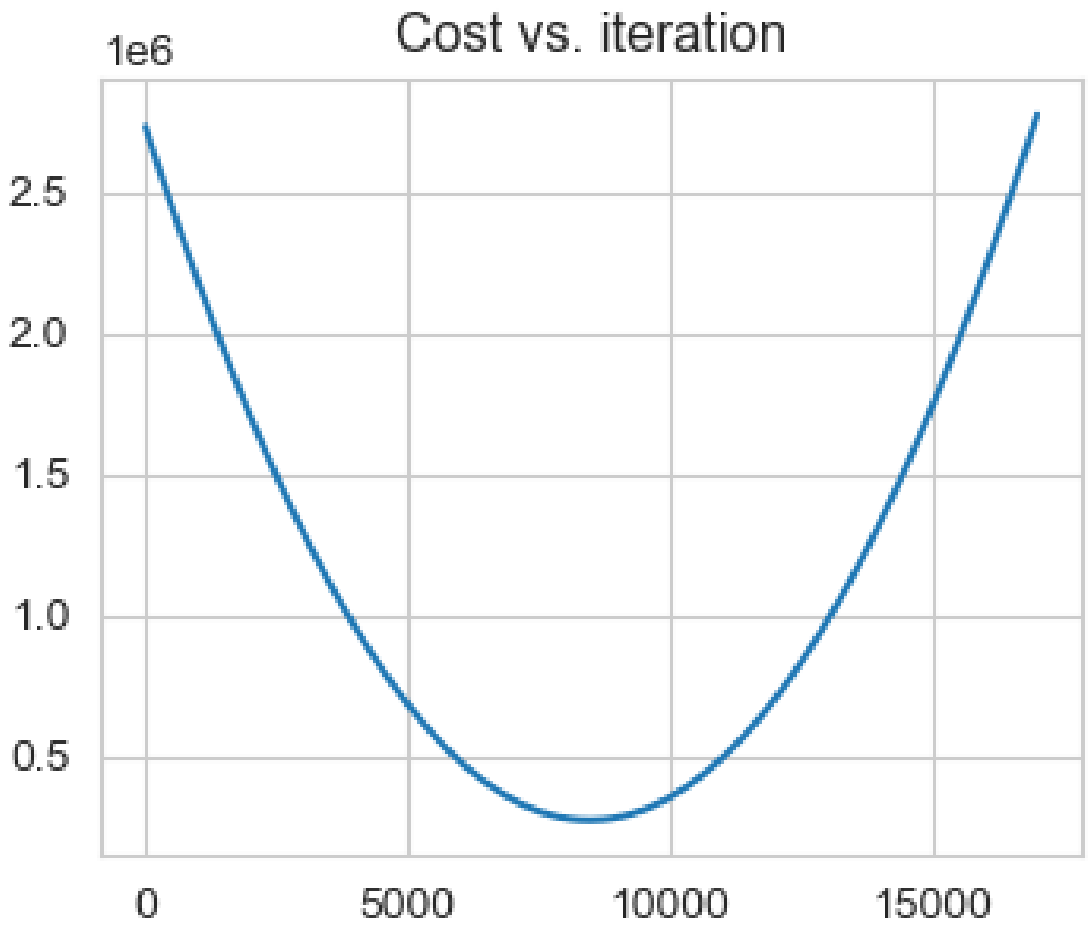

FIGURE 16 – Fonction coût après l'amélioration.

Interprétation :
La représentation du fonction coût montre que la fonction est admet un optimale a un point compris entre 5 000 et 10 000, ce qui implique qu'en ce point le gradient de la fonction est nul.

6.9 Représentation du prix prédictive avec prix réel

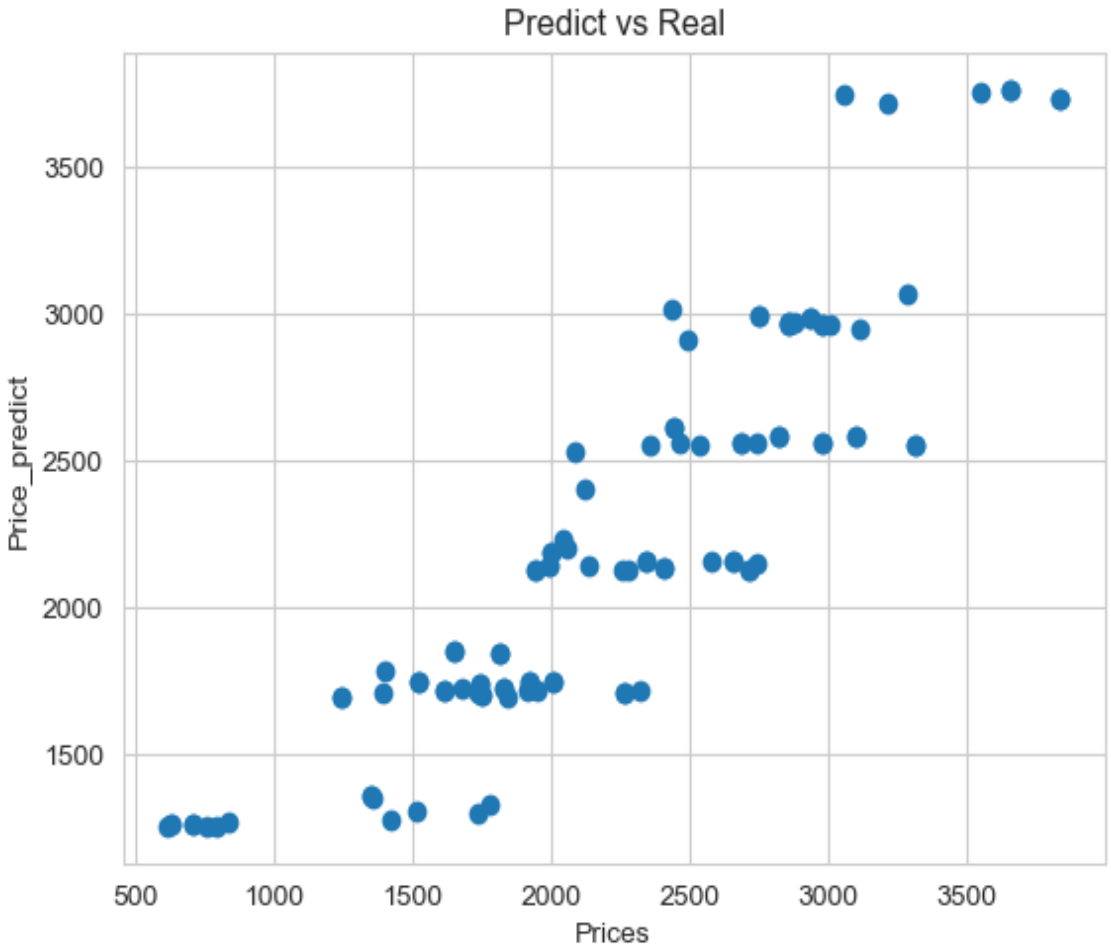

FIGURE 17 – Résultat du prédiction après améliore.

Interprétation :
La Représentation de prix prédictive en fonction du prix réel montre une relation linéaire, fortement positive. Donc la prédiction est bonne.

6.10 Représentation du prix prédictive avec prix réel en utilisant la fonction SGDRegressor

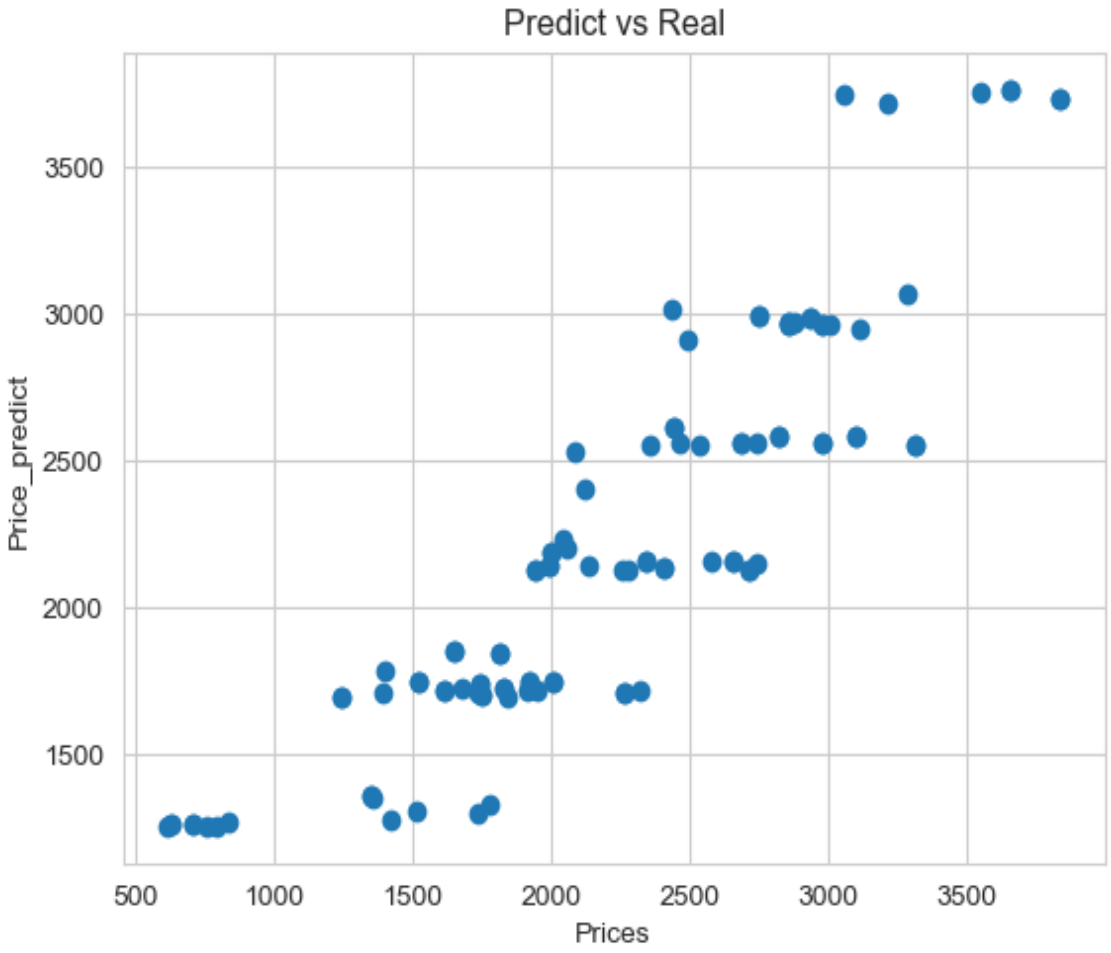

FIGURE 18 – Fonction cout après l'amélioration.

Interprétation :
La Représentation de prix prédictive en fonction du prix réel montre une relation linéaire, fortement positive. Donc la prédiction est bonne.

7 Conclusion

En conclusion, dans ce rapport, nous avons montré comment utiliser la descente de gradient pour la régression linéaire, en utilisant l'ensemble de données prédiction du prix des mobiles.
Les résultats obtenus à partir de cet exemple démontre que la descente de gradient peut être un algorithme d'optimisation efficace pour trouver le modèle la mieux adaptée à travers un ensemble de points de données.

D'après notre analyse de prédiction du prix de mobile, on peut dire que le modèle suivant est très efficace pour prédire les prix du mobile.

$$F(x) = 2215 - 19.5x_1 + 648x_2 + 64.8x_3$$