```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import numpy as np
warnings.filterwarnings('ignore')

data=pd.read_csv("50_Startups.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

retourne la dimension du matrice (50 lignes,5 colonnes)

```
data.shape
```

```
(50, 5)
```

retourne une description statistique pour la base de données en se base sur moyen, mean , quartile ,ecartype

```
data.describe()
```

```
           R&D Spend  Administration  Marketing Spend         Profit
count      50.000000       50.000000        50.000000      50.000000
mean    73721.615600   121344.639600    211025.097800  112012.639200
std     45902.256482    28017.802755    122290.310726   40306.180338
min         0.000000    51283.140000         0.000000   14681.400000
25%     39936.370000   103730.875000    129300.132500   90138.902500
50%     73051.080000   122699.795000    212716.240000  107978.190000
75%    101602.800000   144842.180000    299469.085000  139765.977500
max    165349.200000   182645.560000    471784.100000  192261.830000
```

```
data.head()
```

```
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
```

nombres de valeurs null pour chaque variable dans la base de données

```
data.isnull().sum()
```

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

data.duplicated().sum()#nombres des valeurs redondantes dans la BD
data.drop_duplicates(keep=False)#effacer les duplications et laisse une exemplaire si les duplications existe (keep=False)

```
data.duplicated().sum()
data.drop_duplicates(keep=False)
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56

```
32   63408.86       129219.61         46085.25  California   97427.84
33   55493.95       103057.49        214634.81     Florida   96778.92
34   46426.07       157693.92        210797.67  California   96712.80
35   46014.02        85047.44        205517.64    New York   96479.51
36   28663.76       127056.21        201126.82     Florida   90708.19
37   44069.95        51283.14        197029.42  California   89949.14
38   20229.59        65947.93        185265.10    New York   81229.06
39   38558.51        82982.09        174999.30  California   81005.76
40   28754.33       118546.05        172795.67  California   78239.91
41   27892.92        84710.77        164470.71     Florida   77798.83
42   23640.93        96189.63        148001.11  California   71498.49
43   15505.73       127382.30         35534.17    New York   69758.98
44   22177.74       154806.14         28334.72  California   65200.33
45    1000.23       124153.04          1903.93    New York   64926.08
46    1315.46       115816.21        297114.46     Florida   49490.75
47       0.00       135426.92             0.00  California   42559.73
48     542.05        51743.15             0.00    New York   35673.41
49       0.00       116983.80         45173.06  California   14681.40
```

print(data.Profit.std())#calcule d'ecart-type print(data.Profit.mean())#calcule de moyenne

```
print(data.Profit.std())
print(data.Profit.mean())
```

```
40306.18033765055
112012.63920000002
```

```
data.skew()
```

```
R&D Spend          0.164002
Administration    -0.489025
Marketing Spend   -0.046472
Profit             0.023291
dtype: float64
```

facteur d'applatissement , le coefficient kurtosis mesure la degré d'irrégularité d'une distribution."

```
data.kurtosis()
```

```
R&D Spend         -0.761465
Administration     0.225071
Marketing Spend   -0.671701
Profit            -0.063859
dtype: float64
```

retourne les parametres statistiques de BD, Nous remarquons que les valeurs de la médiane et de la moyenne ne sont pas proches, donc nous ne pouvons pas approximer cette distribution à une gaussienne.

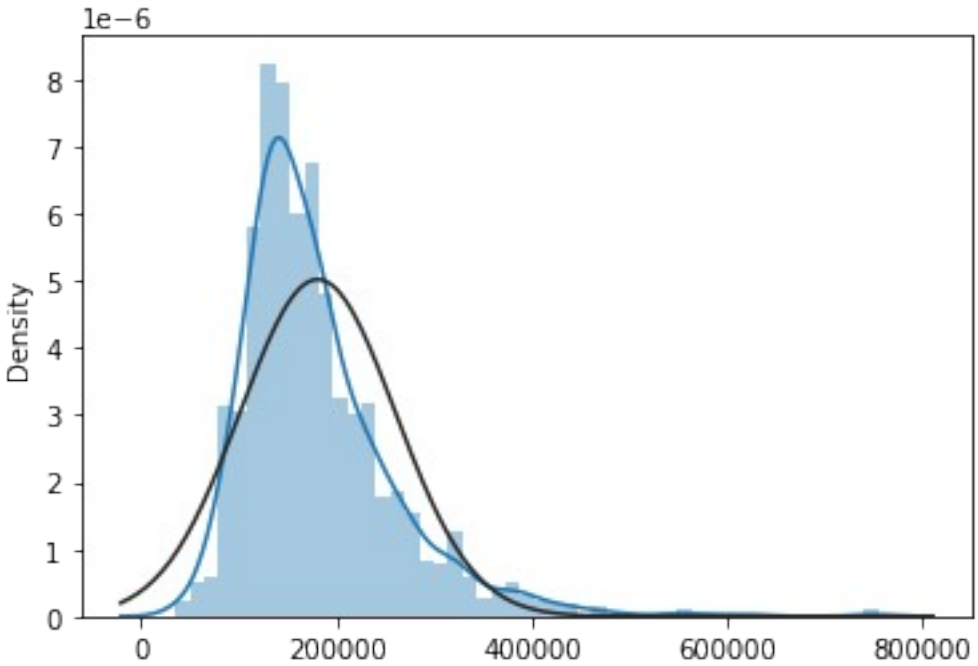
```
dt=pd.read_csv("dataFrame.csv")
dt.describe()
```

```
           SalePrice
count    1460.000000
mean   180921.195890
std     79442.502883
min     34900.000000
25%    129975.000000
50%    163000.000000
75%    214000.000000
max    755000.000000
```

Nous affichons la courbe de la distribution de la base de données et nous la comparons avec la distribution normale (fit=norm).

```
import seaborn as sb
from scipy.stats import norm
sb.distplot(dt,fit=norm)
```

```
<AxesSubplot:ylabel='Density'>
```



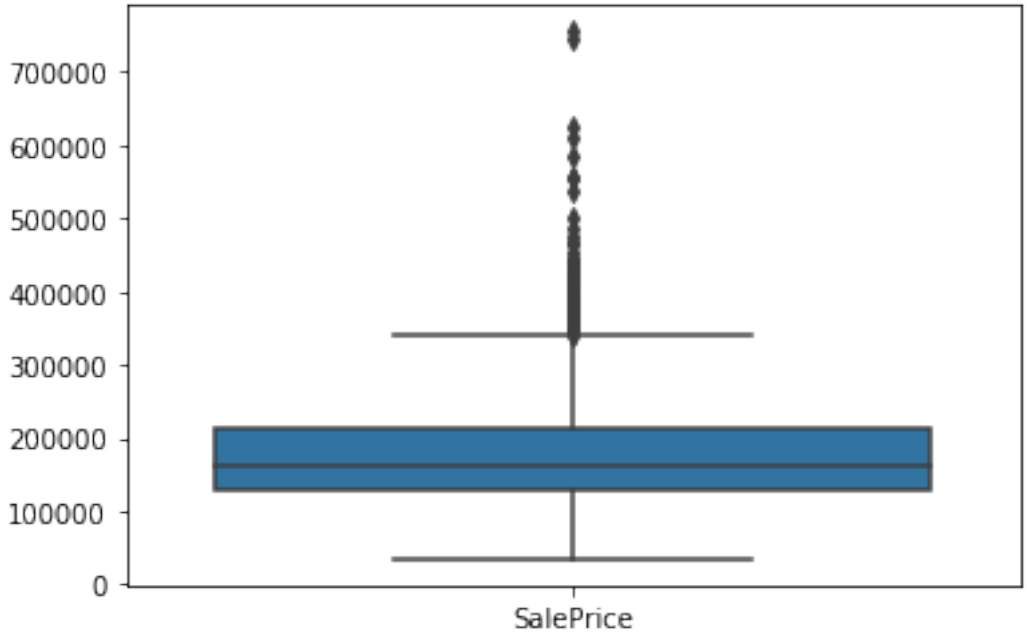

```
dt.skew()
```

```
SalePrice    1.882876
dtype: float64
```

On remarque l'apparition de plusieurs points abérrantes dans cette distribution

```
sb.boxplot(data=dt)
```

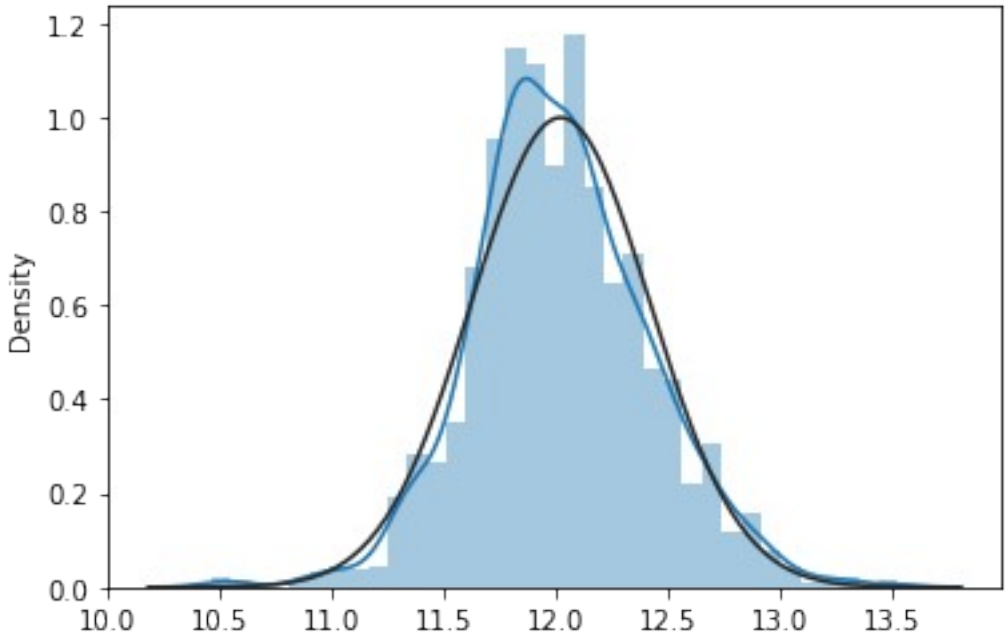
```
<AxesSubplot:>
```



En appliquant la fonction logarithme à la distribution initiale, nous remarquons qu'il y a une amélioration de la courbe de distribution et que nous pouvons l'approcher d'une gaussienne.

```
data2=np.log(dt)
sb.distplot(data2,fit=norm)
data2.skew()
```

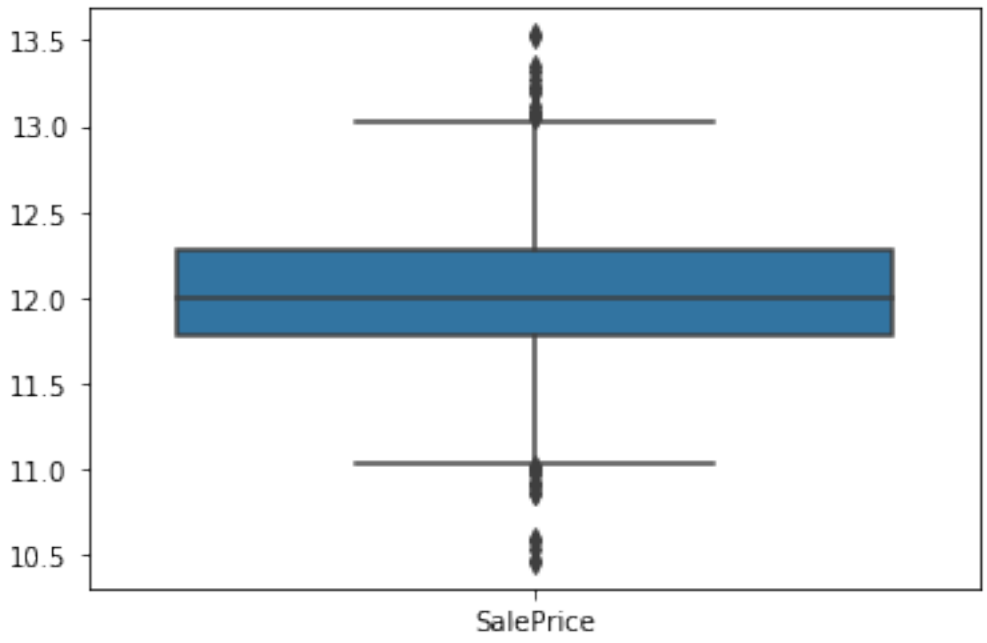
```
SalePrice    0.121335
dtype: float64
```



On peut afficher la boite de moustache d'une seul variable. Cette figure indique les informations suivantes: la ligne inférieure : 1er quantile la ligne supérieure : 3eme quantile la ligne intérieur : 2eme quantile (ou mediane) la ligne inferieur au dessous : minimum limite la ligne superieur au dessus : maximum limite les points dessous minimum limite ou dessus maximum limite : les valeurs aberants On remaeque que la distribution est uniformement distribué et presque symétrique

```
sb.boxplot(data=data2)
```

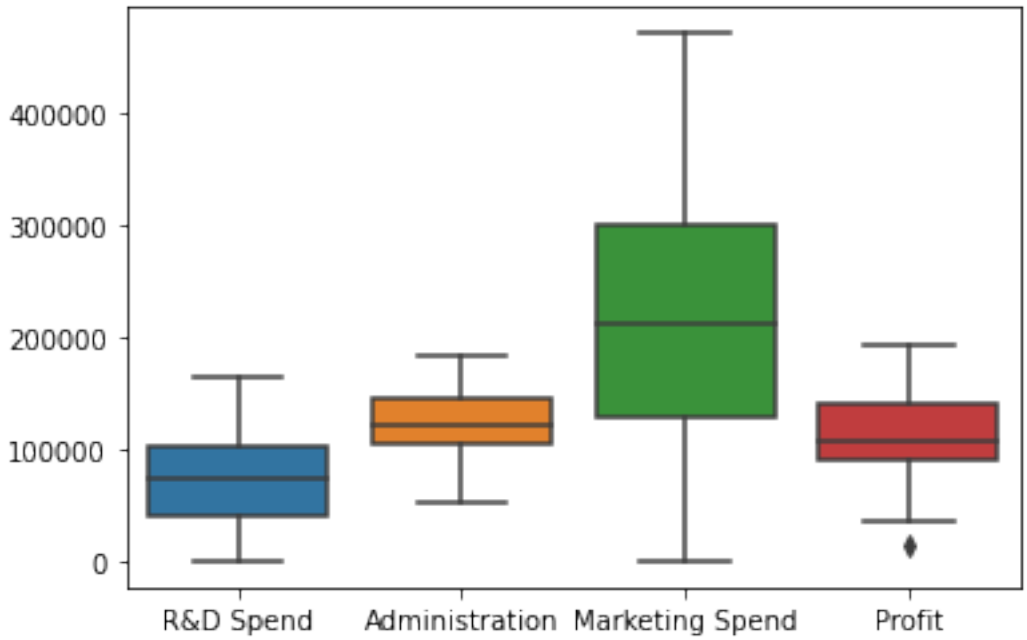
```
<AxesSubplot:>
```



La méthode boxplot fournit par la librairie seaborn affiche la boite de moustache pour toutes les variables.

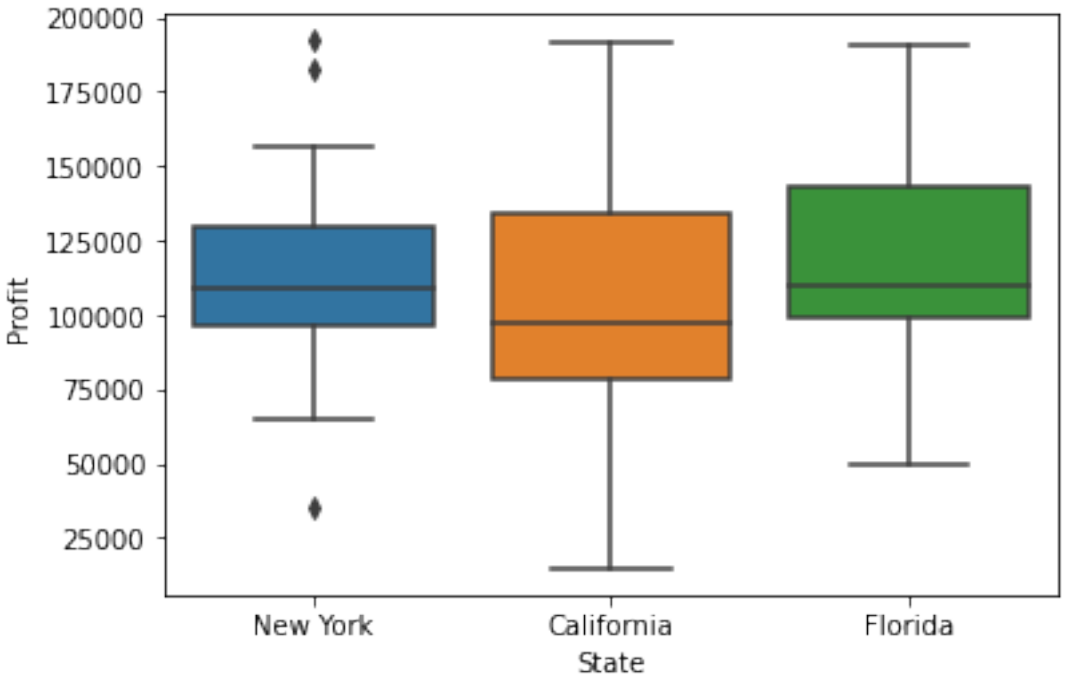
```
sb.boxplot(data=data)
```

```
<AxesSubplot:>
```




```
sb.boxplot(y=data.Profit,x=data.State)
```

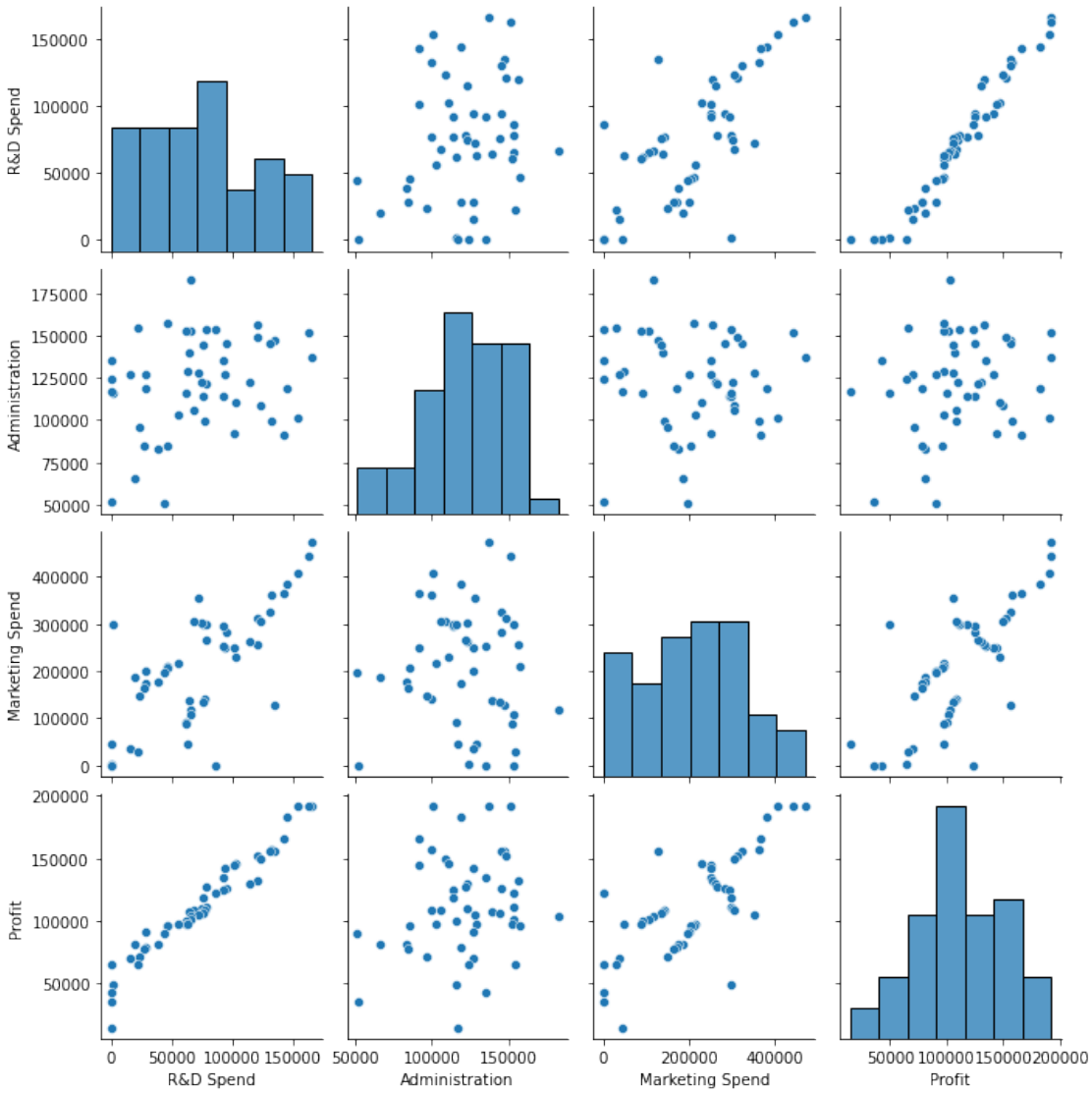
```
<AxesSubplot:xlabel='State', ylabel='Profit'>
```



A l'aide de la methode pairplot fournit par seaborn, on peut afficher la nuage des points de chaque couple de variables et des histogrammes dans la diagonale pour visualiser la distribution de chaque variable On remarque une forte correlation entre R&D Spend et Profit

```
sb.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x1fb68dc9460>
```



Calcule des coefficients de corrélations entre les differents variables en utilisant la méthode de pearson

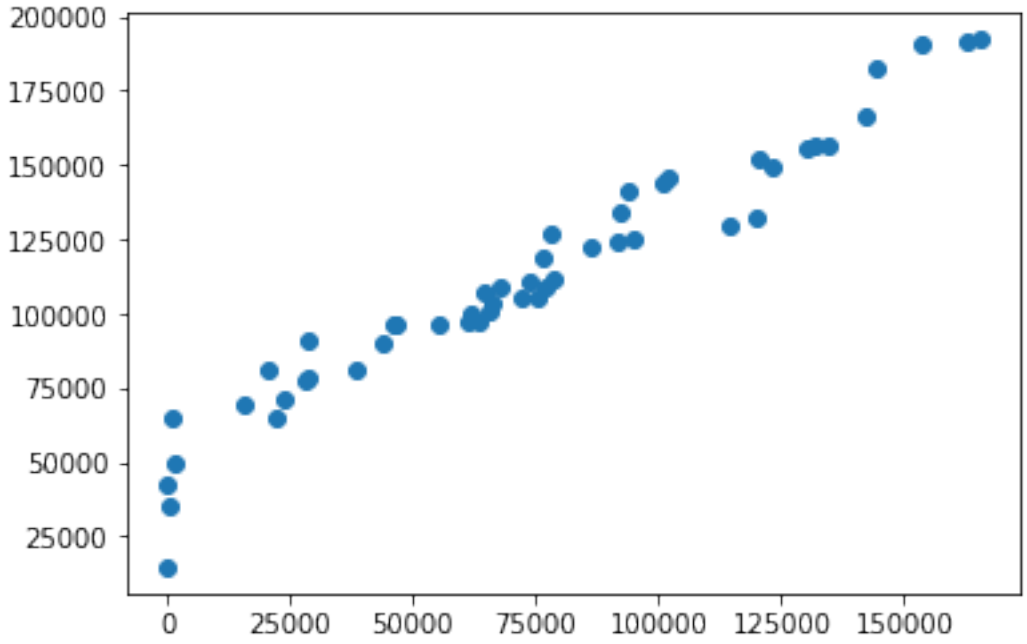
```
c=data.corr(method="pearson")
c
```

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.241955	0.724248	0.972900
Administration	0.241955	1.000000	-0.032154	0.200717
Marketing Spend	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

On peut afficher le nuage des points à l'aide de la méthode scatter de matplotlib. Cette figure présente le nuage des points de profit en fonction de R&D spend on peut constater qu'il y'a une forte correlation entre les deux variables profit et r&d spend.

```
plt.scatter(x=data["R&D Spend"],y=data["Profit"])
```

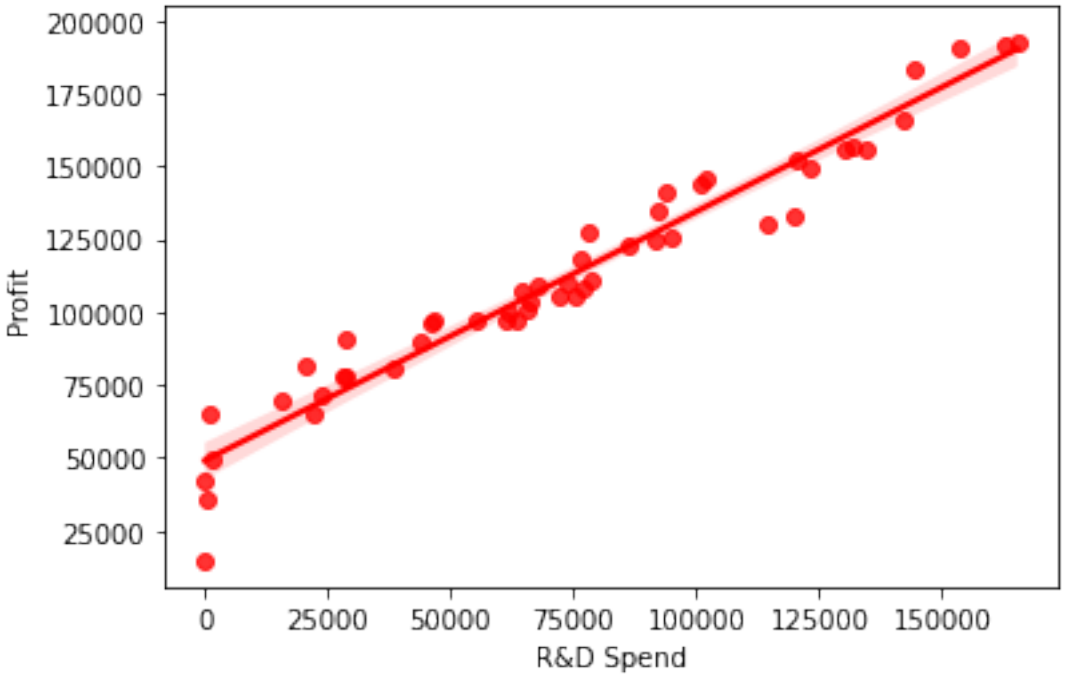
```
<matplotlib.collections.PathCollection at 0x1fb698323d0>
```



A l'aide de la méthode regplot fournit par la librarie seaborn,on peut afficher la nuage des point entre deux variables . Cette méthode est utilisée pour tracer les données et la courbe d'ajustement d'une régression linéaire

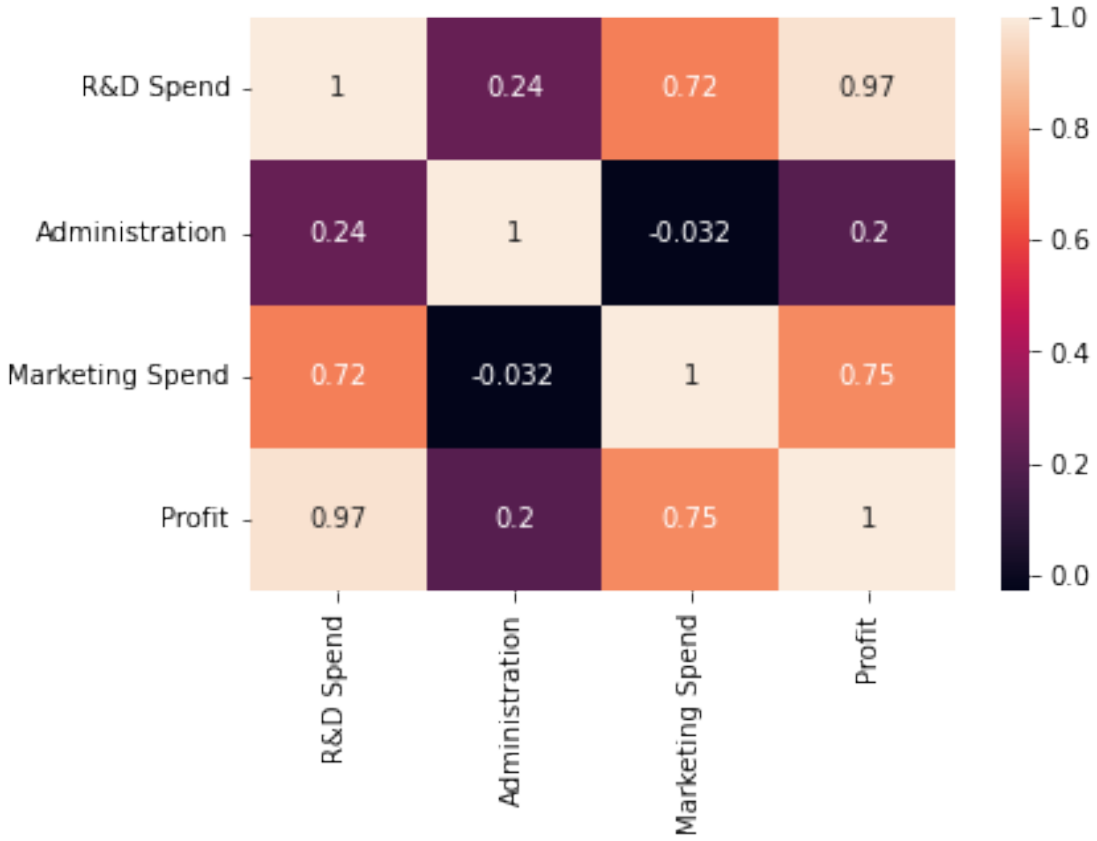
```
sb.regplot(x=data["R&D Spend"],y=data["Profit"],color="red")
```

```
<AxesSubplot:xlabel='R&D Spend', ylabel='Profit'>
```



```
sb.heatmap(c,annot=True)#R&D spend et Profit sont fortement correlé
```

```
<AxesSubplot:>
```



Préparation des parametres d'entrées

```
x=data.iloc[:,:-1]
print(x)
```

```
    R&D Spend  Administration  Marketing Spend       State
0   165349.20       136897.80        471784.10    New York
1   162597.70       151377.59        443898.53  California
2   153441.51       101145.55        407934.54     Florida
3   144372.41       118671.85        383199.62    New York
4   142107.34        91391.77        366168.42     Florida
5   131876.90        99814.71        362861.36    New York
6   134615.46       147198.87        127716.82  California
7   130298.13       145530.06        323876.68     Florida
8   120542.52       148718.95        311613.29    New York
9   123334.88       108679.17        304981.62  California
10  101913.08       110594.11        229160.95     Florida
11  100671.96        91790.61        249744.55  California
12   93863.75       127320.38        249839.44     Florida
13   91992.39       135495.07        252664.93  California
14  119943.24       156547.42        256512.92     Florida
```

```
15  114523.61       122616.84        261776.23    New York
16   78013.11       121597.55        264346.06  California
17   94657.16       145077.58        282574.31    New York
18   91749.16       114175.79        294919.57     Florida
19   86419.70       153514.11             0.00    New York
20   76253.86       113867.30        298664.47  California
21   78389.47       153773.43        299737.29    New York
22   73994.56       122782.75        303319.26     Florida
23   67532.53       105751.03        304768.73     Florida
24   77044.01        99281.34        140574.81    New York
25   64664.71       139553.16        137962.62  California
26   75328.87       144135.98        134050.07     Florida
27   72107.60       127864.55        353183.81    New York
28   66051.52       182645.56        118148.20     Florida
29   65605.48       153032.06        107138.38    New York
30   61994.48       115641.28         91131.24     Florida
31   61136.38       152701.92         88218.23    New York
32   63408.86       129219.61         46085.25  California
33   55493.95       103057.49        214634.81     Florida
34   46426.07       157693.92        210797.67  California
35   46014.02        85047.44        205517.64    New York
36   28663.76       127056.21        201126.82     Florida
37   44069.95        51283.14        197029.42  California
38   20229.59        65947.93        185265.10    New York
39   38558.51        82982.09        174999.30  California
40   28754.33       118546.05        172795.67  California
41   27892.92        84710.77        164470.71     Florida
42   23640.93        96189.63        148001.11  California
43   15505.73       127382.30         35534.17    New York
44   22177.74       154806.14         28334.72  California
45    1000.23       124153.04          1903.93    New York
46    1315.46       115816.21        297114.46     Florida
47       0.00       135426.92             0.00  California
48     542.05        51743.15             0.00    New York
49       0.00       116983.80         45173.06  California
```

Préparation de la sortie Y

```
y=data.iloc[:,-1]
print(y)
```

```
0     192261.83
1     191792.06
2     191050.39
3     182901.99
4     166187.94
5     156991.12
6     156122.51
7     155752.60
8     152211.77
9     149759.96
```

```
10    146121.95
11    144259.40
12    141585.52
13    134307.35
14    132602.65
15    129917.04
16    126992.93
17    125370.37
18    124266.90
19    122776.86
20    118474.03
21    111313.02
22    110352.25
23    108733.99
24    108552.04
25    107404.34
26    105733.54
27    105008.31
28    103282.38
29    101004.64
30     99937.59
31     97483.56
32     97427.84
33     96778.92
34     96712.80
35     96479.51
36     90708.19
37     89949.14
38     81229.06
39     81005.76
40     78239.91
41     77798.83
42     71498.49
43     69758.98
44     65200.33
45     64926.08
46     49490.75
47     42559.73
48     35673.41
49     14681.40
Name: Profit, dtype: float64
```

get_dummies permet la convertion de la colonnes state de valeur non ordinal a des colonnes de valeurs binaire . Le parametre drop_first efface la 1ere colonne transformé car la machine peut conclure les valeurs du 1ere colonne à l'aide des autres colonnes

```
x=pd.get_dummies(x,drop_first=True)
print(x)
```

```
    R&D Spend  Administration  Marketing Spend  State_Florida
State_New York
```

0	165349.20	136897.80	471784.10	0	1
1	162597.70	151377.59	443898.53	0	0
2	153441.51	101145.55	407934.54	1	0
3	144372.41	118671.85	383199.62	0	1
4	142107.34	91391.77	366168.42	1	0
5	131876.90	99814.71	362861.36	0	1
6	134615.46	147198.87	127716.82	0	0
7	130298.13	145530.06	323876.68	1	0
8	120542.52	148718.95	311613.29	0	1
9	123334.88	108679.17	304981.62	0	0
10	101913.08	110594.11	229160.95	1	0
11	100671.96	91790.61	249744.55	0	0
12	93863.75	127320.38	249839.44	1	0
13	91992.39	135495.07	252664.93	0	0
14	119943.24	156547.42	256512.92	1	0
15	114523.61	122616.84	261776.23	0	1
16	78013.11	121597.55	264346.06	0	0
17	94657.16	145077.58	282574.31	0	1
18	91749.16	114175.79	294919.57	1	0
19	86419.70	153514.11	0.00	0	1
20	76253.86	113867.30	298664.47	0	0
21	78389.47	153773.43	299737.29	0	1
22	73994.56	122782.75	303319.26	1	0
23	67532.53	105751.03	304768.73	1	0
24	77044.01	99281.34	140574.81	0	1

25	64664.71	139553.16	137962.62	0	0
26	75328.87	144135.98	134050.07	1	0
27	72107.60	127864.55	353183.81	0	1
28	66051.52	182645.56	118148.20	1	0
29	65605.48	153032.06	107138.38	0	1
30	61994.48	115641.28	91131.24	1	0
31	61136.38	152701.92	88218.23	0	1
32	63408.86	129219.61	46085.25	0	0
33	55493.95	103057.49	214634.81	1	0
34	46426.07	157693.92	210797.67	0	0
35	46014.02	85047.44	205517.64	0	1
36	28663.76	127056.21	201126.82	1	0
37	44069.95	51283.14	197029.42	0	0
38	20229.59	65947.93	185265.10	0	1
39	38558.51	82982.09	174999.30	0	0
40	28754.33	118546.05	172795.67	0	0
41	27892.92	84710.77	164470.71	1	0
42	23640.93	96189.63	148001.11	0	0
43	15505.73	127382.30	35534.17	0	1
44	22177.74	154806.14	28334.72	0	0
45	1000.23	124153.04	1903.93	0	1
46	1315.46	115816.21	297114.46	1	0
47	0.00	135426.92	0.00	0	0
48	542.05	51743.15	0.00	0	1
49	0.00	116983.80	45173.06	0	0

Préparation des parties train et test 20% partie test 80% partie train random_state=0 pour choisir la 1ere choix

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,
y,random_state=0, test_size=0.2)
```

En remarquant le grand décalage des valeurs entre les 3 premières colonnes et les valeurs des colonnes (State_Florida et State_NewYork : valeur entre 0 et 1), nous avons utilisé la méthode StandarScaler qui permet de normaliser toutes les valeurs de la base de données entre 0 et 1.Nous faisons ce traitement pour éviter la dominance d'une variable sur une autre

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_train.iloc[:,:-2])
x_train.iloc[:,:-2]=sc.transform(x_train.iloc[:,:-2])
x_train
```

```
    R&D Spend  Administration  Marketing Spend  State_Florida
State_New York
33  -0.350065       -0.785471         0.101197              1
0
35  -0.555303       -1.481174         0.027350              0
1
26   0.079358        0.801334        -0.551521              1
0
34  -0.546382        1.325058         0.070117              0
0
18   0.434854       -0.355987         0.751485              1
0
7    1.269431        0.855185         0.986031              1
0
14   1.045250        1.280770         0.440400              1
0
45  -1.529843        0.029421        -1.621875              0
1
48  -1.539763       -2.767673        -1.637297              0
1
29  -0.131152        1.144977        -0.769500              0
1
15   0.927916       -0.029921         0.483032              0
1
30  -0.209329       -0.299377        -0.899154              1
0
32  -0.178708        0.225135        -1.264016              0
0
16   0.137471       -0.069294         0.503847              0
0
42  -1.039676       -1.050767        -0.438521              0
```

20	0.099383	-0.367903	0.781818	0	0
43	-1.215802	0.154162	-1.349478	0	1
8	1.058224	0.978368	0.886701	0	1
13	0.440120	0.467547	0.409232	0	0
25	-0.151519	0.624306	-0.519831	0	0
5	1.303611	-0.910735	1.301798	0	1
17	0.497811	0.837707	0.651491	0	1
40	-0.928972	-0.187170	-0.237691	0	0
49	-1.551498	-0.247517	-1.271405	0	0
1	1.968711	1.081067	1.958181	0	0
12	0.480634	0.151771	0.386346	1	0
37	-0.597392	-2.785442	-0.041403	0	0
24	0.116490	-0.931339	-0.498672	0	1
6	1.362901	0.919649	-0.602819	0	0
23	-0.089432	-0.681423	0.831261	1	0
36	-0.930933	0.141566	-0.008215	1	0
21	0.145619	1.173615	0.790508	0	1
19	0.319472	1.163598	-1.637297	0	1
9	1.118678	-0.568313	0.832985	0	0
39	-0.716714	-1.560956	-0.219842	0	0
46	-1.523018	-0.292619	0.769263	1	0
3	1.574137	-0.182310	1.466534	0	1
0	2.028280	0.521733	2.184048	0	1
47	-1.551498	0.464915	-1.637297	0	0

```
44  -1.071354        1.213507        -1.407792              0
0

x_test.iloc[:,:-2]=sc.transform(x_test.iloc[:,:-2])
print(x_test)

    R&D Spend  Administration  Marketing Spend  State_Florida
State_New York
28  -0.121495        2.288905        -0.680323              1
0
11   0.628031       -1.220695         0.385578              0
0
10   0.654901       -0.494342         0.218855              1
0
41  -0.947621       -1.494179        -0.305121              1
0
2    1.770481       -0.859327         1.666881              1
0
27   0.009618        0.172791         1.223412              0
1
38  -1.113531       -2.218962        -0.136691              0
1
31  -0.227907        1.132224        -0.922749              0
1
22   0.050470       -0.023512         0.819521              1
0
4    1.525099       -1.236102         1.328585              1
0
```

Créer un modéle lineaire à l'aide de la classe LinearRegression fournit par sklearn. le coefficient R (score)=0.95 , donc le modèle est acceptable

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
lr.score(x_train,y_train)
```

```
0.9501847627493607
```

la valeur de l'intercept β0 Les valeurs des paramètres des vaiables (X1,X2,X3,X4,X5)==>(β1,β2,β3,β4,β5)

```
print(f"intercept: {lr.intercept_}")
print(f"slope: {lr.coef_}")
```

```
intercept: 109441.48912163253
slope: [35726.28774249   851.30163448  4519.88277698  -959.28416006
   699.36905252]
```

afficher les valeurs de R-squared et mean squared error

```
from sklearn.metrics import r2_score,mean_squared_error
y_pred=lr.predict(x_test)
print(r2_score(y_pred,y_test))
# y_predm
print(mean_squared_error(y_pred,y_test))
```

```
0.929374920931811
83502864.03257751
```