Tables des matières

Tutoriel complet Regression lineaire

Collecter data en utilisant pandas

In [1]:
```
# modules nécessaires pour le notebook
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# lire le fichier de données
#utiliser le param index_col: Column to use as the row labels of the DataFrame
df = pd.read_csv('Advertising.csv', index_col=0)
df.head()
```

Out[2]:

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

In [3]:
```
df.describe()
```

Out[3]:

	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

identification des descripteurs, cible et observations

Quels sont les descripteurs? On a 3 descripteurs dans ce dataset qui sont:

- TV
- Radio
- Newspaper

Quelle est la cible?

- Sales: vente d'un produit

Quelle est la forme ou shape du dataframe?

In [4]:
```
df.shape
```

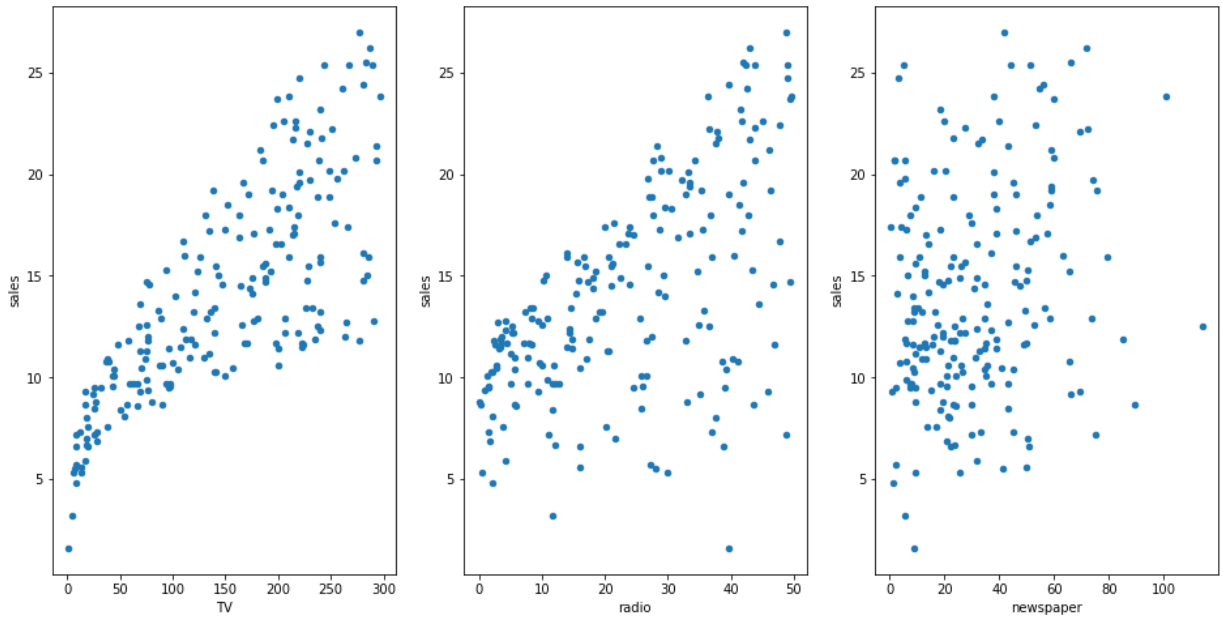
Out[4]:
```
(200, 4)
```

On voit que l'on a 200 observations avec 4 colonnes dont 3 sont des descripteurs

Tracé des relations entre les descripteurs et la cible

In [5]:
```
#utilisation d'une figure avec 3 plots aligné sur une ligne
fig, axes = plt.subplots(1,3,sharey=False)
df.plot(kind='scatter', x='TV', y='sales',
        ax=axes[0], figsize=(16,8))
df.plot(kind='scatter', x='radio', y='sales',
        ax=axes[1], figsize=(16,8))
df.plot(kind='scatter', x='newspaper', y='sales',
        ax=axes[2], figsize=(16,8))
```

Out[5]:
```
<AxesSubplot:xlabel='newspaper', ylabel='sales'>
```

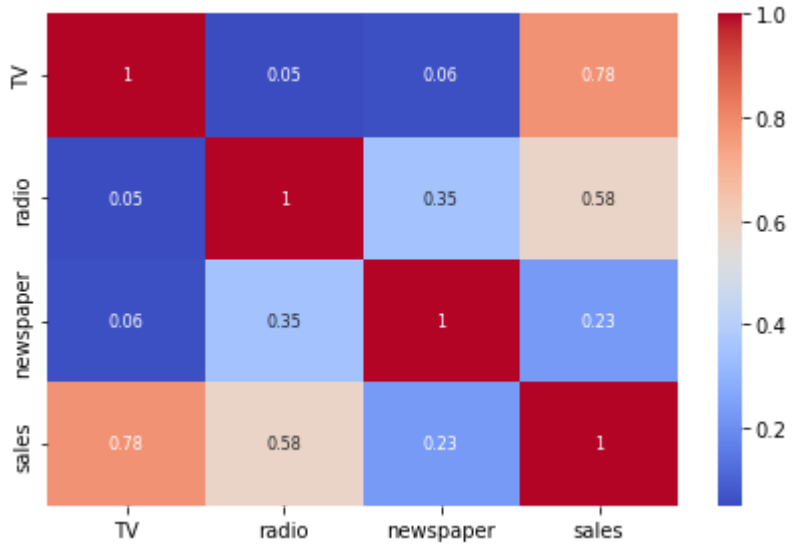


On voit au niveau des graphes qu'il existe une certaine relation linéaire entre TV et Sales ainsi que radio et Sales

Tracé des correlations entre les différents descripteurs et cible

- On utilise ici seaborn pour avoir la matrice des correlations

In [6]:
```
import seaborn as sns
#We can visualise the correlation between all the variables in the dataset using
sns.heatmap(data=df.corr().round(2),
            cmap='coolwarm', annot=True, annot_kws={"size":8})
plt.tight_layout()
plt.show()
```



On confirme qu'il n'y a pas vraiment de dépendance entre les descripteurs.

Développement du modele linear regression

In [7]:
```
from sklearn.linear_model import LinearRegression
cols_predicteurs = ['TV','radio','newspaper']
#predicteurs
X = df[cols_predicteurs]
y = df.sales
```

In [8]:
```
#Effectuer la séparation Training-Test
from sklearn import model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
                                                    y , test_size = 0.2, random_state=15)
#detail de chacun des sous-dataset
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(160, 3) (160,)
(40, 3) (40,)
```

In [9]:
```
#estimation des coeeficients du modele lineaire
lm = LinearRegression()
lm.fit(X_train,y_train)
#Afficher les coefficients
print(lm.intercept_)
print(lm.coef_)
```

```
2.657585917146074
[0.04681996 0.18857656 0.00145037]
```

In [10]:
```
#Afficher l'equation
list(zip(cols_predicteurs, lm.coef_))
```

Out[10]:
```
[('TV', 0.04681995525137476),
 ('radio', 0.18857655931851042),
 ('newspaper', 0.0014503705461097907)]
```

In [11]:
```
#Sauvegarder intercept et coefficient dans fichier
#fichier rl.csv ou rl.pkl csv ou pickle (dump et load)
```

In [12]:
```
# proceder au test
y_pred = lm.predict(X_test)
```

In [13]:
```
import numpy as np
#comparer les valeurs test et prédites
test_pred_df = pd.DataFrame( { 'Valeurs test': y_test,
                 'Valeurs prédites': np.round( y_pred, 2),
                 'residuels': y_test - y_pred } )
test_pred_df[0:10]
```

Out[13]:

	Valeurs test	Valeurs prédites	residuels
50	9.7	8.05	1.650440
192	9.9	8.24	1.662178
13	9.2	10.49	-1.286518
173	7.6	7.39	0.209698
128	8.8	6.43	2.374110
41	16.6	16.39	0.210284
31	21.4	21.77	-0.370523
171	8.4	7.21	1.187241
139	9.6	9.58	0.015291
58	13.2	12.68	0.520790

In [14]:
```
from sklearn import metrics
# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test,
                                         y_pred)))

#Calcul du R-squared
r2 = metrics.r2_score(y_test, y_pred)
print(r2)
```

```
1.664692694432902
0.9064939179490465
```

Modele avec seulement TV et radio

In [15]:
```
cols_predicteurs = ['TV','radio']
#predicteurs
X = df[cols_predicteurs]
y = df.sales

#Effectuer la séparation Training-Test


X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y ,
                                test_size = 0.2, random_state=15)
#detail de chacun des sous-dataset
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)

#estimation des coeeficients du modele lineaire
lm = LinearRegression()
lm.fit(X_train,y_train)
#Afficher les coefficients
print(lm.intercept_)
print(lm.coef_)

#Afficher l'equation
list(zip(cols_predicteurs, lm.coef_))

# proceder au test
y_pred = lm.predict(X_test)

#comparer les valeurs test et prédites
test_pred_df = pd.DataFrame( { 'Valeurs test': y_test,
                               'Valeurs prédites': np.round( y_pred, 2),
                               'residuels': y_test - y_pred } )
print(test_pred_df[0:10])
```

```
(160, 2) (160,)
(40, 2) (40,)
2.6816380062166196
[0.04683203 0.18937371]
     Valeurs test  Valeurs prédites  residuels
50            9.7              8.03   1.669627
192           9.9              8.26   1.637308
13            9.2             10.44  -1.243257
173           7.6              7.41   0.194043
128           8.8              6.44   2.362433
41           16.6             16.39   0.211843
31           21.4             21.76  -0.358015
171           8.4              7.22   1.180026
139           9.6              9.60  -0.000194
58           13.2             12.70   0.503865
```

In [16]:
```
# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

#Calcul du R-squared
r2 = metrics.r2_score(y_test, y_pred)
print(r2)
```

```
1.660481657759206
0.9069663889509988
```

In [17]:
```
#Référence: The Elements of Statistical Learning - Hastie, Tibshirani and Friedmc
voir https://web.stanford.edu/~hastie/ElemStatLearn/
```