

Projet personnel:
Prédiction des ventes liées au marketing

DÉCOUVRIR CI-DESSOUS

avec Python sur Jupyter Notebook
Marvin Hugues Laurac

INTRODUCTION

Face à une compétitivité des marchés par une guerre de la publicité, qu'elle soit papier ou numérique, il est important d'identifier la source des ventes qui contribue à la hausse du chiffre d'affaires afin d'optimiser les campagnes publicitaires.

Ce projet a pour but de créer un modèle qui peut aider à la détection des tendances afin que le pôle marketing ait plus de recul sur la fiabilité de leurs campagnes, avec des données d'analyse et d'apprentissage automatique.

SOMMAIRE

- 1 Nettoyage des données, Compréhension des tendances
- 2 Modélisation du jeu de données, test et prédiction
- 3 Validation de la prédiction
- 4 Conclusion résultat

Crédit du DataFrame Kaggle

Marvin Hugues Laurac

LOGICIEL PRIVILÉGIÉ

- Python

J'ai utilisé Python avec ces bibliothèques, comme Pandas pour l'analyse de données, Scikit-Learn pour le machine learning, et Matplotlib pour la visualisation.

PRÉDICTION DES VENTES LIÉES AU MARKETING**CONTEXTE DU SUJET**

Pour ce projet, je n'ai pas eu d'information de l'auteur du DataFrame, aucune indication et description, mais je peux en déduire une fois mon analyse rapide sur DataFrame cela représente une base de données publicitaires, avec comme information le flux de clients potentiels, par la publicité sur différentes plateformes dont: la télévision, la radio et le format papier.

DATAFRAME INFO

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column   Non-Null Count   Dtype  
 --- 
  0   TV        200 non-null    float64 
  1   Radio     200 non-null    float64 
  2   Newspaper 200 non-null    float64 
  3   Sales     200 non-null    float64 
dtypes: float64(4)
memory usage: 6.4 KB
```

LICENCE

kaggle (Database Contents License (DbCL) v1.0)

Ce projet a une deadline de 5 jours
pour me challenger, Go!

Marvin Hugues Laurac

```
Entrée [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, classification_report
%matplotlib inline
```

```
Entrée [2]: je commence par importer le DataFrame
df = pd.read_csv("/Users/octoberone/Desktop/DataFrame GitHub/DataFrame/Prédiction des ventes liées au marketing.csv")
```

```
Entrée [3]: #affichage des 5 premières lignes
df.head()
```

```
Out[3]:
      TV   Radio  Newspaper   Sales
0  230.1    37.8      69.2   22.1
1   44.5    39.3      45.1   10.4
2   17.2    45.9      69.3   12.0
3  151.5    41.3      58.5   16.5
4  180.8    10.8      58.4   17.9
```

```
Entrée [4]: #voici le nombre de lignes et de colonnes sans le détail
df.shape
```

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

PRÉDICTION DES VENTES LIÉES AU MARKETING

Entrée [5]: #voici les détails des colonnes avec beaucoup plus d'informations (variables, types, le nombre de valeurs nulles)
df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   TV        200 non-null    float64
 1   Radio     200 non-null    float64
 2   Newspaper 200 non-null    float64
 3   Sales     200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

Entrée [6]: #par souci du détail, j'ai utilisé cette commande pour être sûr qu'il n'y a pas de valeurs manquantes
df.isnull().sum()
#par conséquent, j'en déduis que ce DataFrame est plutôt agréable à utiliser dans un premier temps

```
Out[6]: TV      0
Radio    0
Newspaper 0
Sales    0
dtype: int64
```

Entrée [7]: #Afin d'avoir plus de détails statistiques, j'affiche les colonnes numériques

```
#count%:nombre de valeurs nulles
#mean%:la moyenne des valeurs
#std%:l'écart type (dispersion valeurs)
#min%:le min de la colonne
#25%:1er quartile à moins de 25
#50%:2er quartile à moins de 25
#75%:3er quartile à moins de 25
#max:le maxi de la colonne
```

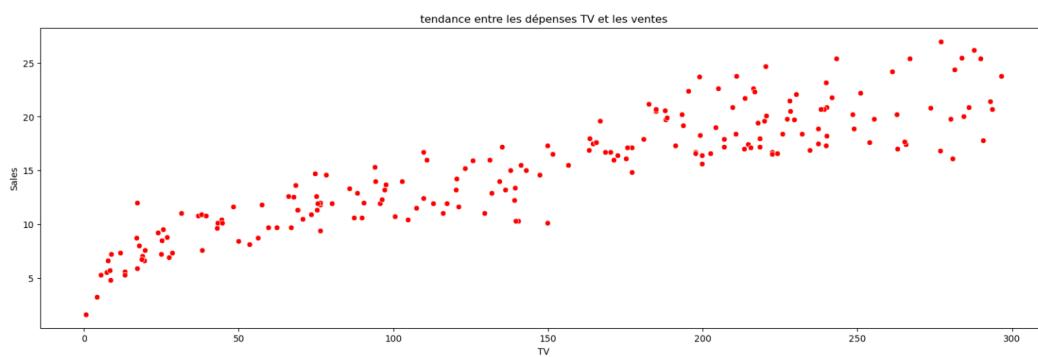
```
print(df.describe())
   TV      Radio  Newspaper  Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500  23.264000  30.554000  15.130500
std    85.854236  14.846809  21.778621  5.283892
min    0.700000  0.000000  0.300000  1.600000
25%   74.375000  9.975000  12.750000  11.000000
50%   149.750000  22.900000  25.750000  16.000000
75%   218.825000  36.525000  45.100000  19.050000
max   296.400000  49.600000  114.000000  27.000000
```

Entrée [21]: #premier affichage sous forme de nuage de points afin de visualiser la tendance et son évolution dans le temps des ventes
#conclusion : j'observe une tendance progressive avec des données progressives d'un taux d'engagement lié à la publicité

```
plt.figure(figsize=(20, 6))

sns.scatterplot(x="TV", y="Sales", color= "r", data=df)
plt.title("tendance entre les dépenses TV et les ventes")
```

Out[21]: Text(0.5, 1.0, 'tendance entre les dépenses TV et les ventes')

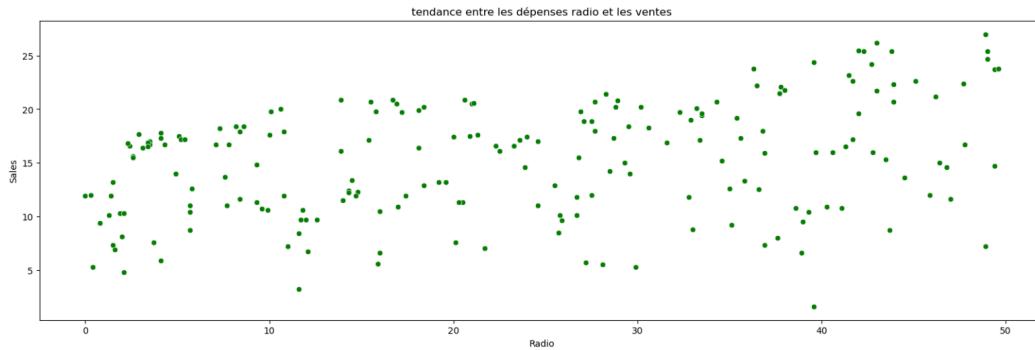


PRÉDICTION DES VENTES LIÉES AU MARKETING

Entrée [22]: *#deuxième affichage sous forme de nuage de points afin de visualiser la tendance et son évolution dans le temps des #conclusion : nous n'avons pas de tendance, mais les publicités émises ont un impact relatif sur le CA*

```
plt.figure(figsize=(20, 6))
sns.scatterplot(x="Radio", y="Sales", color= "g", data=df)
plt.title("tendance entre les dépenses radio et les ventes")
```

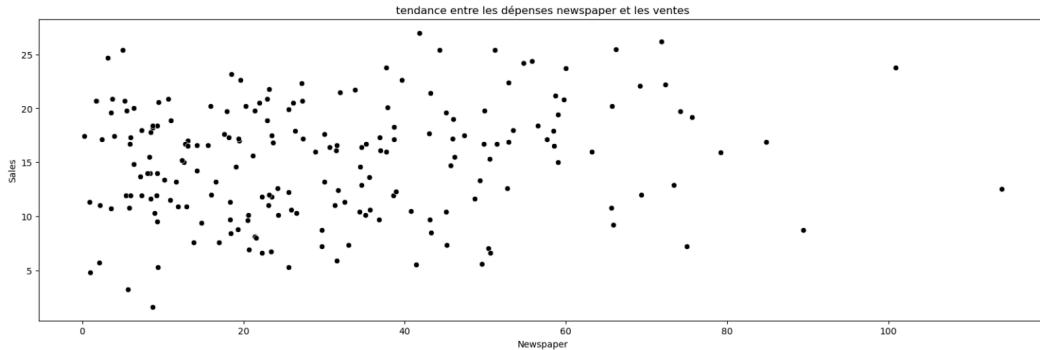
Out[22]: Text(0.5, 1.0, 'tendance entre les dépenses radio et les ventes')



Entrée [23]: *#troisième affichage sous forme de nuage de points afin de visualiser la tendance et son évolution dans le temps des #conclusion : nous n'avons pas de tendance, mais les publicités émises ont moins d'impact sur le CA*

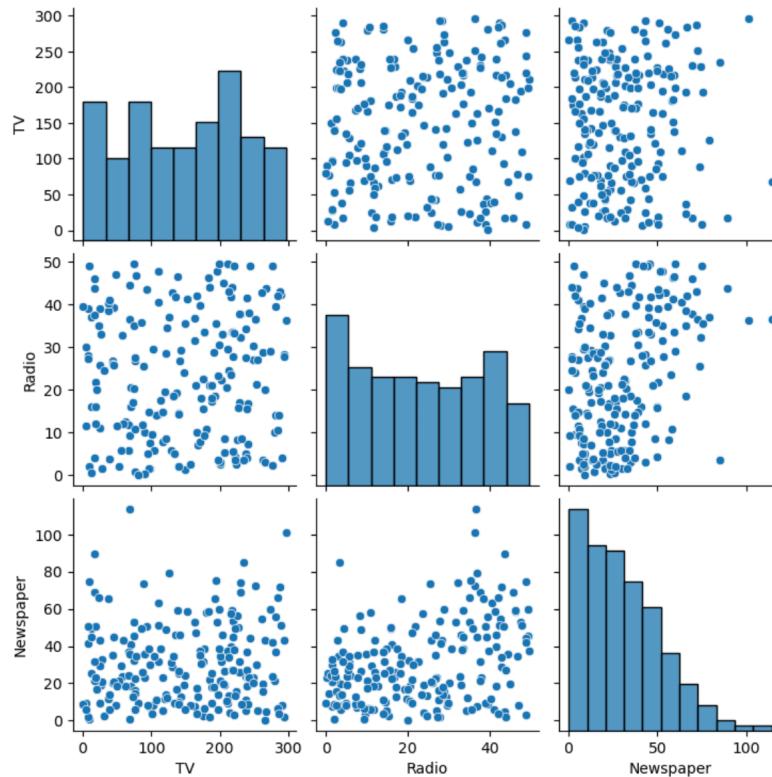
```
plt.figure(figsize=(20, 6))
sns.scatterplot(x="Newspaper", y="Sales", color= "black", data=df)
plt.title("tendance entre les dépenses newspaper et les ventes")
```

Out[23]: Text(0.5, 1.0, 'tendance entre les dépenses newspaper et les ventes')



PRÉDICTION DES VENTES LIÉES AU MARKETING

```
Entrée [11]: #afin d'avoir une visualisation plus mathématique,  
#j'ai pris la décision de regarder les tendances avec l'aide de matrices de corrélation  
#pour pouvoir avoir des données exploitables, j'ai préféré les afficher également sous forme de chiffres ci-dessous  
sns.pairplot(df, vars=["TV", "Radio", "Newspaper"])  
plt.show()  
  
/Users/octoberone/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout ha  
s changed to tight  
    self._figure.tight_layout(*args, **kwargs)
```



```
Entrée [25]: #dans cette matrice, j'observe  
#TV: a une corrélation plutôt positive avec un coefficient de += 1  
#Radio : a une corrélation plutôt négative par rapport à la TV avec un coef de -= 0  
#Newspaper : a une corrélation également plutôt négative par rapport à la TV avec un coef de -= 0  
  
corr = df[["TV", "Radio", "Newspaper"]].corr()  
print(corr)  
  
TV      TV      Radio   Newspaper  
TV  1.000000  0.054809  0.056648  
Radio  0.054809  1.000000  0.354104  
Newspaper  0.056648  0.354104  1.000000
```

```
Entrée [13]: #premièrement, je vais séparer les colonnes dans la variable X sauf "Sales" dans Y  
#pour isoler les caractéristiques de l'entraînement  
X = df[["TV", "Radio", "Newspaper"]]  
y = df["Sales"]
```

```
Entrée [14]: #maintenant je vais fractionner le jeu de données  
#pour tester la performance de la prédiction  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
Entrée [15]: #ensuite une normalisation  
#pour améliorer les résultats  
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

```
Entrée [16]: #par la suite je crée une régression linéaire pour orienter le model avec ce paramètre  
#(un paramètre qui aide à construire une prédiction)  
linear_model = LinearRegression()  
linear_model.fit(X_train, y_train)
```

```
Out[16]: ▾ LinearRegression  
LinearRegression()
```

PRÉDICTION DES VENTES LIÉES AU MARKETING

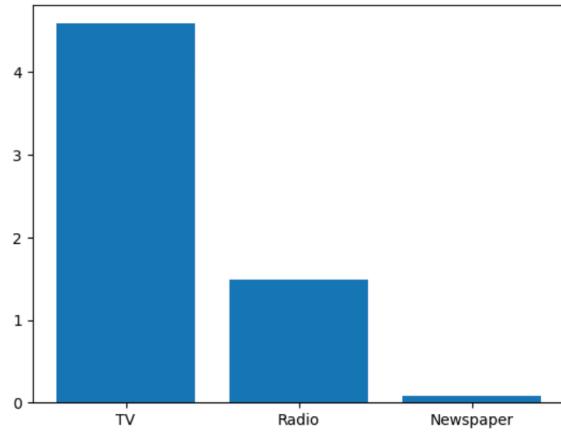
```
Entrée [17]: #et enfin de calculer la performance de la régression sur l'entraînement.  
score_train = linear_model.score(X_train , y_train)  
print(score_train)  
0.9001416005862131
```

```
Entrée [18]: #et aussi la régression sur les tests  
score_test = linear_model.score(X_test , y_test)  
print(score_test)  
  
0.9059011844150826
```

```
Entrée [19]: #afin de confirmer la prédiction, je vais calculer l'erreur quadratique moyenne et le coefficient  
y_pred_test = linear_model.predict(X_test)  
  
mse = mean_squared_error(y_test, y_pred_test)  
r2 = r2_score(y_test, y_pred_test)  
  
print(mse)  
print(r2)  
#j'observe une performance MSE beaucoup trop élevée  
#j'observe une performance R2 plutôt performante avec un 90% cela se rapproche fortement de 100%  
2.907756910271091  
0.9059011844150826
```

```
Entrée [28]: #dans cette partie, je regarde les coefficients qui ont été prédits par rapport aux différents  
#canaux de campagne publicitaire  
#de plus, pour une meilleure visualisation possible des effectifs, je l'affiche sous forme d'histogramme  
features = X.columns  
coef = linear_model.coef_  
  
print(features)  
print(coef)  
plt.bar(features, coef)  
  
Index(['TV', 'Radio', 'Newspaper'], dtype='object')  
[4.58720774 1.48984025 0.08791597]
```

Out[28]: <BarContainer object of 3 artists>



PRÉDICTION DES VENTES LIÉES AU MARKETING

```
Entrée [26]: #pour terminer les résidus je décide d'observer les erreurs possibles du modèle par rapport à la prédiction  
#dans ce graphique j'observe que l'erreur est modérée avec une ligne linéaire  
#cela indique une bonne prédiction pour déterminer  
#l'axe sur lequel il est préférable de croître le marketing (TV)  
residu = y_test - y_pred_test  
  
sns.residplot(x=y_pred_test, y=residu, lowess=True)  
plt.xlabel("Valeurs prédictes")  
plt.ylabel("Le résidu")  
plt.title("Graphique des résidus")  
plt.show()
```

