École Nationale Supérieure d'Informatique
pour l'Industrie et l'Entreprise

Apprentissage statistique

Dissertation realisée par:

BOULIF Ikrame ZAARI Abdelouahab
HEDRI Souhayla ROSENBERG Avraham

Forecast Rossmann Store Sales

Encadrant :

Mr HANCZAR Blaise

31/03/2021

Session 2021/2022

Abstract

Rossmann exploite plus de 3 000 pharmacies dans 7 pays européens. Actuellement, les directeurs de magasins Rossmann sont chargés de prévoir leurs ventes quotidiennes jusqu'à six semaines à l'avance. Les ventes en magasin sont influencées par de nombreux facteurs, notamment les promotions, la concurrence, les vacances scolaires et nationales, la saisonnalité et la localité. Avec des milliers de managers individuels prédisant les ventes en fonction de leurs circonstances uniques, la précision des résultats peut être très variable. Dans cet article, nous étudions le comportement des ventes en tenant compte des changements de facteurs. Tous les facteurs sont-ils responsables des taux de vente ? À quels facteurs devrions-nous accorder plus d'attention pour comprendre l'évolution du comportement des ventes ? Notre travail consiste à trouver des réponses à ces questions en établissant un modèle théorique, afin que nous puissions être en mesure de prédire les ventes dans le futur.

Keywords: Rossmann Store sales, theoretical model, factors, predict.

Table des matières

Remerciements

Ce travail est l'aboutissement d'un travail acharné, de beaucoup de sacrifices, et de beaucoup d'efforts, c'est pourquoi nous adressons nos sincères remerciements à tous ceux qui ont contribué de près ou de loin à la réalisation de cette étude. Nos remerciements vont particulièrement à nos chers encadrants Monsieur HANCZAR Blaise qui, durant toute cette période nous ont été d'un soutien indéfectible. En effet, il était présent pour nous conseiller, répondre à nos questions, nous soutenir mais surtout nous guider.

Table des figures

Chapter 1

Présentation des données

La base de données que nous allons décrire dans ce chapitre est dédiée à l'apprentissage. Pour le test, il existe une autre base de données qui nous été fournie sans les valeurs de la variable cible. Notre travail consiste alors à prévoir la colonne des ventes. Pour cela, on appliquera, dans le chapitre suivant, sur la base de données d'apprentissage une validation croisée pour mieux entraîner les modèles de régression et donc pouvoir générer des valeurs de ventes de la base de données de test qui seront proches de la réalité.

Pour éviter toute confusion, nous appelons données d'entraînement et données de validation respectivement les deux échatillons issues de la validation croisée effectuée sur ce qu'on appelle données d'apprentissage (C'est la base de données que nous allons décrire dans ce chapitre). Alors que nous appelions données de test les donnés qui nous ont été fournies pour faire la prédiction des ventes.

1.1 Description des données

Avec **n = 1017209 observations** de **p = 17 variables**, la base de données (Figure 1.1) que nous manipulions concerne les ventes quotidiennes des *Rossman Stores* (Il s'agit d'une chaîne allemande de magasins spécialisée dans la distribution de

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
0	1	c	a	1270.0	9.0	2008.0	0	NaN
1	1	c	a	1270.0	9.0	2008.0	0	NaN
2	1	c	a	1270.0	9.0	2008.0	0	NaN
3	1	c	a	1270.0	9.0	2008.0	0	NaN
4	1	c	a	1270.0	9.0	2008.0	0	NaN

Promo2SinceYear	PromoInterval	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
NaN	NaN	5	2015-07-31	5263	555	1	1	0	1
NaN	NaN	4	2015-07-30	5020	546	1	1	0	1
NaN	NaN	3	2015-07-29	4782	523	1	1	0	1
NaN	NaN	2	2015-07-28	5011	560	1	1	0	1
NaN	NaN	1	2015-07-27	6102	612	1	1	0	1

Figure 1.1: Aperçu de la base de données

drogueries).

Voici une brève description de quelques champs de données :

- **Store** : clé (identifiant unique) pour chaque magasin
- **Open** : indicateur de la disponibilité du magasin; 0 = closed, 1 = open
- **StateHoliday** : indicateur des jours fériés; a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- **SchoolHoliday** : indique si le tuple (Store, Date) a été infecté par la fermeture des écoles
- **StoreType** : différencie entre 4 types de magasins ; a, b, c, d
- **Assortment** : décrit ue niveau de gamme ; a = basic, b = extra, c = extended
- **CompetitionDistance** : distance en mètres au plus proche magasin concurrent
- **CompetitionOpenSince[Month/Year]** ; donne une approximation de l'an et du mois auxquels le plus proche magasin concurrent a été ouvert
- **Promo** : indique si le magasin offre une promotion ce jour-là
- **Promo2** : promotion continue et consécutive pour quelques magasins ; 0 = le magasin ne participe pas, 1 = le magasin particpe
- **Promo2Since[Year/Week]** : donne l'année et la semaine auxquels le magasin

commence à participer à Promo2

- **PromoInterval** : décrit les intervalles consécutives auxquels Promo 2 a commencé . E.g. ”Feb,May,Aug,Nov” signifie que chaque série de promotion commene à Février, Mai, Août, Novembre de chaque année donnée de ce magasin

Nous soulignons la présence d'occurences de valeurs manquantes qui peuvent être traitées de plusieurs manières. Durant ce travail, nous avons combiner plusieurs méthodes d'imputations que nous présentions dans le chapitre suivant.

1.2 Description du problème

Nous nous intéressons à la prédiction de la colonne ”Sales” sur de nouveaux exemples d'observations. Notre travail consiste alors à identifier les relations éventuelles entre la variable cible qui est les ventes et les autres variables explicatives. Ces relations sont les piliers d'**un modèle théorique** pour la prévision des ventes dans le futur.Il s'agit plus précisément d'un **modèle de régression** puisque nous avons affaire avec une variable de sortie continue.

1.3 Etude univariée

Dans cette section, nous nous intéressons plus précisément à l'analyse statistique de la variable cible.

D'après l'histogramme normé des ventes (Figure 1.2), nous identifions que le mode de la série est autour de la valeur 0. Cela veut dire que, les périodes où les magasins réalisent très peu de ventes sont plus récurrents que les périodes de prospérité des ventes.

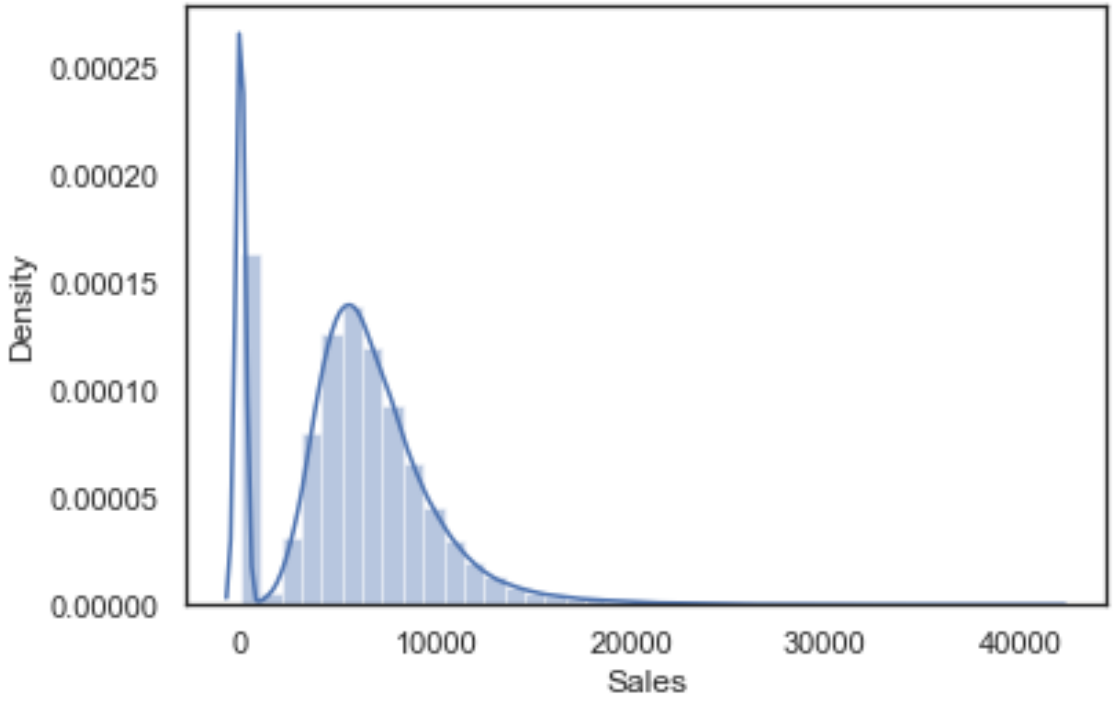

Figure 1.2: Densité empirique des ventes

Toutefois, il apparaît qu'il exite un deuxième mode (2ème pic à droite de la Figure 1.2) de la distribution qui puisse représenter la moyenne d'une loi normale asymétrique approchant la densité empirique des ventes (Table 1.1).

En dehors des périodes de stagnation, les ventes peuvent atteindre un score de 41500 articles vendus (Table 1.1), mais il reste beaucoup moins probable (au sens de densité) de réaliser ce nombre de ventes que de presque ne rien vendre (Figure 1.1).

L'écart-type de la série (Table 1.1) est important, ce qui laisse conclure que les ventes sont dispersés autour de leur moyenne. Néanmoins, à peu près 50% des ventes dépassent la moyenne.

SYNTHESE : Les périodes de stagnation des ventes sont plus répétitives que les périodes de prospérité, alors que 50% du temps, les magasins vendent moins

mean	std	min	25%	50%	75%	max
5.77e+03	3.85e+03	0	3.73e+03	5.740e+03	7.86e+03	4.15e+04

Figure 1.3: Indicateurs statistiques des ventes

que la moyenne.

CONCLUSION : Les ventes sont caractérisées par une certaine saisonnalité. Effectivement, il ne s'agit pas du seul facteur imprégnant les ventes, d'autres facteurs peuvent se révéler lors de l'étude bivariée que nous représentions dans la section prochaine.

1.4 Etude bivariée

Dans cette section, nous élaborons une étude sur les ventes mais cette fois-ci en prenant compte des relations éventuelles qui peuvent exister entre la variable de sortie et les variables explicatives et bien aussi entre ces dernières elles mêmes.

Commençons tout d'abord par visualiser la matrice de corrélation (Figure 1.3). On remarque qu'il exite peu de variables explicatives qui sont liées entre elles. En effet, on identifie parmi ces variables Promo2 et Promo2Since Week, et on trouve aussi Customers et Open qui sont fortement corrélés entre eux.

D'autre part, on remarque aussi que la variable de réponse Sales est très liée à Customers et Open et Promo donc on peut prévoir dès maintenant que ces deux variables contribueront bien à la prédiction des ventes.

Voyons maintenant plus concrètement d'autres relations entre les ventes et quelques prédicteurs sous forme de graphiques.

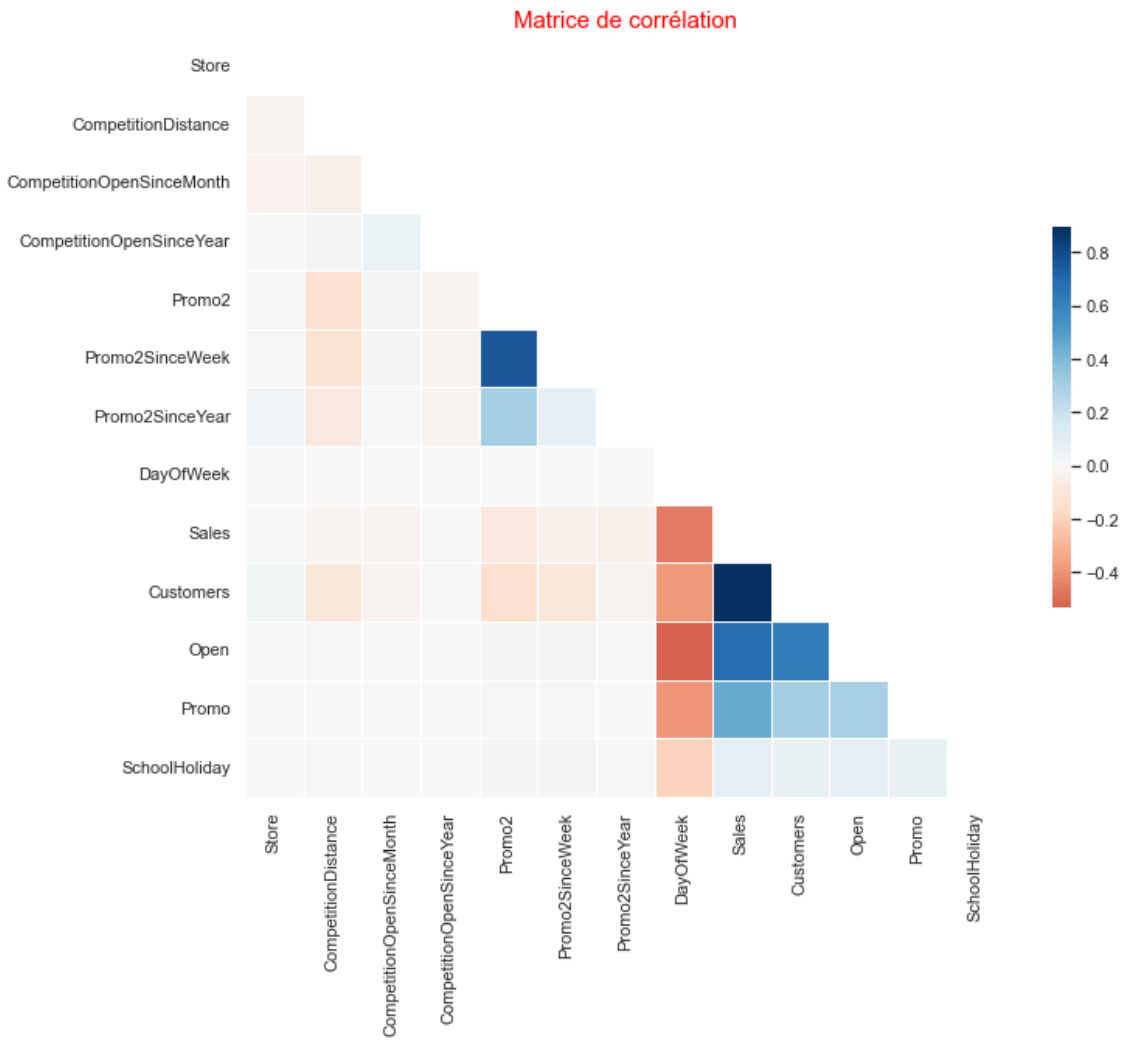

Figure 1.4: Matrice de corrélation

D'après la Figure 1.4, on voit que le maximum des ventes est atteint le mois 12, et pendant la semaine il est atteint le lundi alors que pendant le dimanche on ne vend relativement très peu. Et bien évidemment, durant la période de promotion, les ventes se multiplient. Dans le dernier graphe, on repère le magasin de type b qui enregsitre plus de ventes que les autres types de magasins qui réalisent à peu près le même nombre de ventes.

La figure 1.5 montre que la variation des ventes et celle des clients ont la même tendance ce qui met plus en évidence la relation de causalité entre les ventes et les clients que nous avions déduite plus haut à partir de la matrice corrélation.

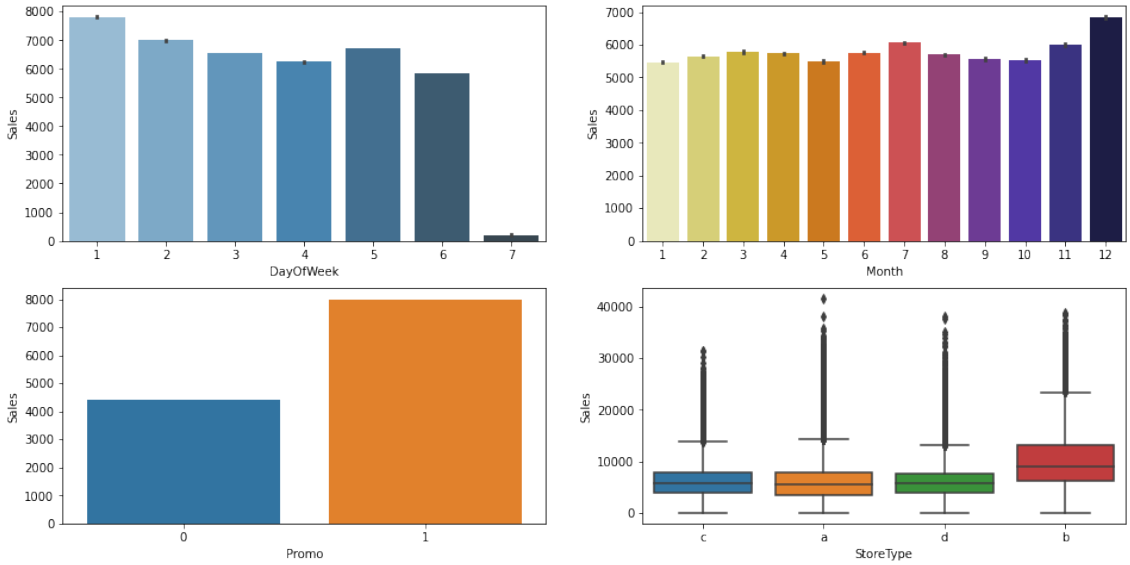

Figure 1.5: Sales = f(DayOfWeek, Month, Promo, StoreType)

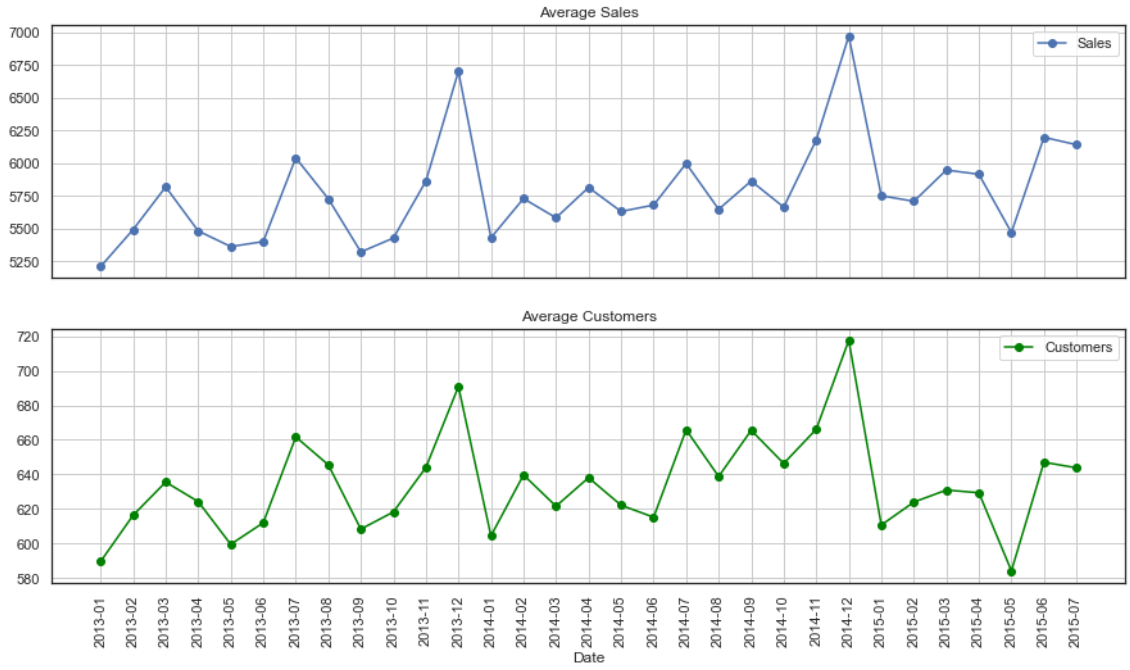

Figure 1.6: Moyenne des ventes et des clients au cours du temps

Chapter 2

Modélisation

listings

Durant la modélisation, nous avons testé une panoplie de méthodes de régression afin de retenir à la fin le modèle le plus performant.
Nous retrouvons le code de modélisation en Python en annexe A.

2.1 Préparation des données

Avant de se lancer dans la modélisation, il est primordial d'effectuer des changements sur les données.

La première chose à faire est le traitement des données manquantes.
Dans notre base de données d'apprentissage, les variables "Promo2SinceWeek", "Promo2SinceYear" et "PromoInterval" présentent un grand nombre de valeurs manquantes; problème qu'on a traité avec la méthode de suppression de données. Par contre, on a remplacé les valeurs manquantes d'autres variables qualitatives par la méthode d'imputation par le mode, en l'occurence 'CompetitionOpenSinceMonth' et 'CompetitionOpenSinceYear'. Et pour la seule donnée quantitative 'CompetitionDistance', on a adopté l'imputation par la moyenne.

La deuxième étape était d'encoder les variables explicatives pour les données d'apprentissage ainsi que pour les données de test (Figure 2.1).

Nous avons aussi supprimer la colonnes des clients pour que cela n'affecte pas l'entraînement des modèles et donc la prédiction des ventes car, comme nous l'avons vu au chapitre 1, les ventes et les clients sont très corrélés entre eux et présentent la même tendance, ce qui est naturel puisque l'un reflète l'état de l'autre.

Dans ce qui suit, nous présenterons les différentes modèles que nous avions entraînés sur la base des données d'apprentissage pour enfin choisir celui le plus performant avec lequel on va faire la prédiction des données test.

2.2 Régression Linéaire

Dans ce modèle, on cherche à établir une relation linéaire entre les ventes Y et les prédicteurs $X = (X_1, X_2, ..., X_p)$: $Y = \beta + \alpha X + \varepsilon$, avec :
$\varepsilon = (\varepsilon_i)_{(1 \leq i \leq n)}$ un terme d'erreur de prédiction
α et β sont les coefficients de la droité affine qui minimise l'erreur quadratique $\sum_{i=1}^{n} (\varepsilon_i)^2$.

2.3 Arbre de décision

Ce type d'apprentissage consiste à construire un arbre depuis un ensemble d'apprentissage constitué de n-uplets étiquetés. Chaque nœud interne décrit un test sur une variable d'apprentissage, chaque branche représente un résultat du test, et chaque feuille contient la valeur de la variable cible (une valeur numérique dans notre cas).

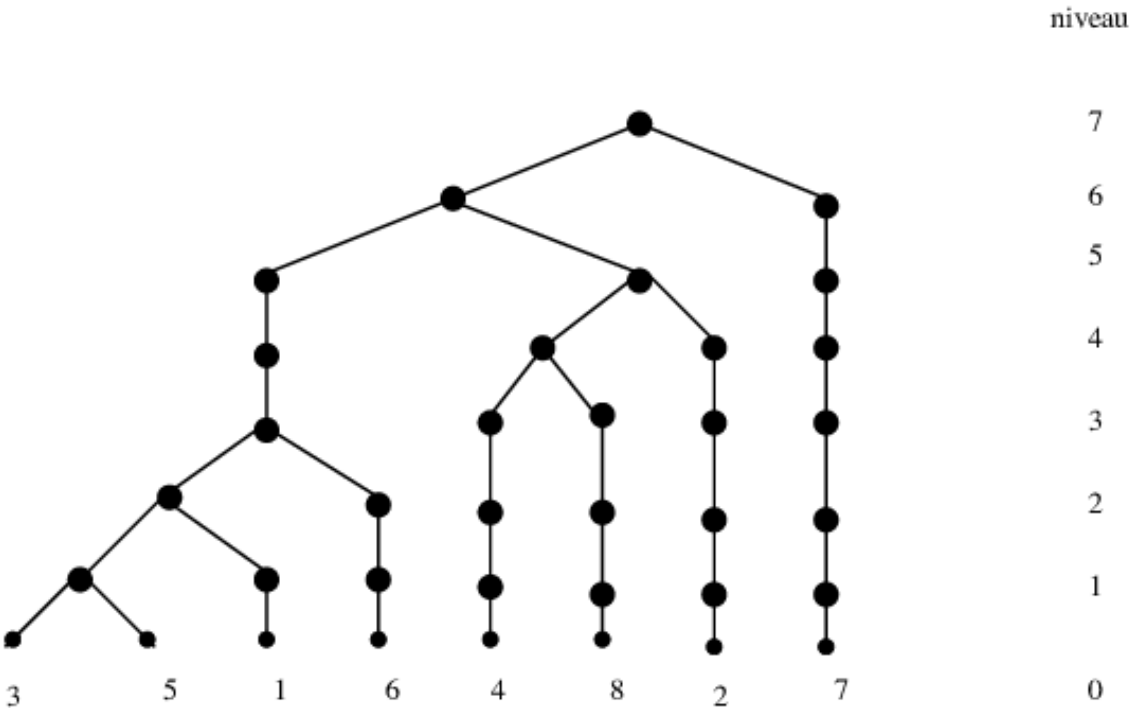

Figure 2.1: Exemple d'arbre de décision

2.4 Forêt d'arbre de décision

Cet outil de prédiction n'est qu'une combinaison de plusieurs arbres de décision (d'où le terme forêt).

2.5 ANNEXE A

Lecture des données

```
store_data = pd.read_csv(r'C:/Users/iquer/OneDrive/Documents/ML/rossmann-store-sales/store.csv')
train_data = pd.read_csv(r"C:/Users/iquer/OneDrive/Documents/ML/rossmann-store-sales/train.csv")
test_data= pd.read_csv(r'C:/Users/iquer/OneDrive/Documents/ML/rossmann-store-sales/test.csv') ## Test
data = pd.merge(store_data,train_data,on='Store')
data_test=pd.merge(store_data,test_data,on='Store') #Données tests
data.head(5)

def Funct_Date(data):
    data['Date'] = pd.to_datetime(data['Date'])
    data['Year'] = data.Date.dt.year
    data['Month'] = data.Date.dt.month
    data['Day'] = data.Date.dt.day

Funct_Date(data)
Funct_Date(data_test) ##Données tests

data.head(5)
```

Imputation des valeurs manquantes

```
#Method for manipulating with missing values
data.drop(columns=["Promo2SinceWeek", "Promo2SinceYear","PromoInterval"], inplace=True)
data_test.drop(columns=["Promo2SinceWeek", "Promo2SinceYear","PromoInterval"], inplace=True)

data['CompetitionDistance'] = data['CompetitionDistance'].fillna(data['CompetitionDistance'].mean())
data['CompetitionOpenSinceMonth'] = data['CompetitionOpenSinceMonth'].fillna(data['CompetitionOpenSinceMonth'].mode().iloc[0])
data['CompetitionOpenSinceYear'] = data['CompetitionOpenSinceYear'].fillna(data['CompetitionOpenSinceYear'].mode().iloc[0])

data_test['CompetitionDistance'] = data_test['CompetitionDistance'].fillna(data['CompetitionDistance'].mean())
data_test['CompetitionOpenSinceMonth'] = data_test['CompetitionOpenSinceMonth'].fillna(data_test['CompetitionOpenSinceMonth'].mode().iloc[0])
data_test['CompetitionOpenSinceYear'] = data_test['CompetitionOpenSinceYear'].fillna(data_test['CompetitionOpenSinceYear'].mode().iloc[0])
data_test['Open'] = data_test['Open'].fillna(data_test['Open'].mode().iloc[0])

#data.info()
```

Etude univariée

```
from scipy.stats import norm
from scipy.stats import skewnorm


sns.distplot(data['Sales'],hist = True, kde=True)

#Knowing informations about the different levels of store type and their distribution among sales
data['Sales'].describe()
```

Etude bivariée

Matrice de corrélation

```
# Plotting a diagonal correlation matrix
sns.set(style="white")
data_modif = data.pop('Sales')
corr = data.corr()
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=np.bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
plt.title("Matrice de corrélation", color = "red", size = 15)


# Generate a custom diverging colormap
cmap = "RdBu"

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(
    corr,
    mask=mask,
    cmap=cmap,
    center=0,
    square=True,
    linewidths=0.5,
    cbar_kws={"shrink": 0.5},
)
```

```
fig, axes = plt.subplots(2,2, sharex=False, figsize=(16,8))
sns.barplot(ax=axes[0,0],x="DayOfWeek", y="Sales", data=data, palette="Blues_d")
sns.barplot(ax=axes[0,1],x="Month", y="Sales", data=data, palette="CMRmap_r")
sns.barplot(ax=axes[1,0],x="Promo", y="Sales", data=data)
sns.boxplot(ax=axes[1,1],x="StoreType", y="Sales", data=data)
```

Moyenne des ventes et des clients au cours du temps

```
data['Date'] = data['Date'].apply(lambda x: (str(x)[:7]))

# group by date and get average sales, and customers among months
fig, (axis1,axis2) = plt.subplots(2,1,figsize=(15,8),sharex=True)

average_sales= data.groupby('Date')["Sales"].mean()
average_customers= data.groupby('Date')["Customers"].mean()

# Plot average customers and Sales over the time
# it should be correlated with the average sales over time

ax1 = average_sales.plot(legend=True,marker='o',ax=axis1 ,title="Average Sales",grid=True)
ax1.set_xticks(range(len(average_sales)))
ax1.set_xticklabels(average_sales.index.tolist(), rotation=90)

ax2 = average_customers.plot(legend=True,marker='o', ax=axis2,title="Average Customers",color='green',grid=True )
ax2.set_xticks(range(len(average_customers)))
ax2.set_xticklabels(average_customers.index.tolist(), rotation=90)
```

Encodage des variables

```
data['StateHoliday'].unique()
data['StateHoliday'] = data['StateHoliday'].map({'0':0, 0:0,'a':1,'b':2,'c':3}).astype(int)
data['StoreType'] = data['StoreType'].map({'c':0, 'a':1,'d':2,'b':3}).astype(int)
data['Assortment'] = data['Assortment'].map({'a':0, 'c':1,'b':2}).astype(int)

data_test['StateHoliday'].unique()
data_test['StateHoliday'] = data_test['StateHoliday'].map({'0':0, 0:0,'a':1,'b':2,'c':3}).astype(int)
data_test['StoreType'] = data_test['StoreType'].map({'c':0, 'a':1,'d':2,'b':3}).astype(int)
data_test['Assortment'] = data_test['Assortment'].map({'a':0, 'c':1,'b':2}).astype(int)
```

Modélisation

```
#Fonction pour calculer RMSPE
def ToWeight(y):
    w = np.zeros(y.shape, dtype=float)
    ind = y != 0
    w[ind] = 1./(y[ind]**2)
    return w

def RMSPE(y_predict, y):
    w = ToWeight(y)
    rmspe = np.sqrt(np.mean( w * (y_predict - y)**2 ))
    return rmspe

#Fonction pour faire l'apprentissage des modèles ainsi donner les performances de chaque modèle
def train_and_predict(name, algorithm, train_data, test_data):

    from sklearn.metrics import r2_score, mean_squared_error

    algorithm.fit(train_data['x'], train_data['y'])
    print(name, " model")
    y_pred= algorithm.predict(test_data['x'])
    y_pred = np.array(list(map(lambda x: 0 if x < 0 else x, y_pred)))
    rmspe = RMSPE(y_pred, test_data['y'])

    print('MSE score :  %.4f' % mean_squared_error(y_pred,test_data['y']))
    print("RMSPE : ", rmspe)
    print("Model score : ",algorithm.score(test_data['x'], test_data['y']))
    print('R2 score : %.4f' % r2_score(y_pred,test_data['y']))

    pred_df=pd.DataFrame({'Predictions':y_pred,'Actual':test_data['y']}).reset_index(drop=True)
    print(pred_df.head())
    print('-'*100)
    return rmspe

#valisation croisée
from sklearn.model_selection import train_test_split
import numpy as np
X = data.drop(['Sales','Date', 'Customers'],axis=1)
y = data['Sales']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size=0.3, random_state=1)
data_test=data_test.drop(['Date'],axis=1)

data_test.set_index('Id', inplace = True)

#Modelisation with Linear regression method
from sklearn.linear_model import LinearRegression
model_Linear = LinearRegression().fit(X_train, y_train)

#Modelisation with Decision tree regressor method
from sklearn.tree import DecisionTreeRegressor
model_decis_tree = DecisionTreeRegressor(max_depth=11)

#Modelisation with Random forest regressor method
from sklearn.ensemble import RandomForestRegressor

model_random_forest = RandomForestRegressor(n_jobs=-1, random_state=42)

models = {
    "LinearRegression": model_Linear,
    "DecisionTreeRegression": model_decis_tree,
    "RandomForestRegression": model_random_forest
}

models_rsmpe = {}
for model in models:
    models_rsmpe[model] = train_and_predict(model, models[model], {'x': X_train, 'y': y_train}, {'x': X_test, 'y':y_test})
```

Prédiction

```
test = pd.read_csv('test.csv')
store_data = pd.read_csv('store.csv')

test.fillna(1,inplace=True)
store_data['CompetitionDistance'] = store_data['CompetitionDistance'].fillna(store_data['CompetitionDistance'].mean())
store_data['PromoInterval'] = store_data['PromoInterval'].fillna(data['PromoInterval'].mode().iloc[0])
store_data.fillna(0,inplace=True)
```

```
Funct_Date(test)
test.drop(["Date"],axis=1,inplace=True)

mappings = {'0':0, 'Jan,Apr,Jul,Oct':1, 'Feb,May,Aug,Nov':2, 'Mar,Jun,Sept,Dec':3}
store_data.PromoInterval.replace(mappings, inplace=True)
store_data['PromoInterval'] = store_data['PromoInterval'].astype(int)

test = pd.merge(test,store_data,on='Store')
test.reset_index(drop=True, inplace=True)
test.set_index('Id', inplace=True)
test.reset_index()

test['StoreType'] = test['StoreType'].map({'c':0, 'a':1,'d':2,'b':3}).astype(int)
test['Assortment'] = test['Assortment'].map({'a':0, 'c':1,'b':2}).astype(int)
test['StateHoliday'] = test['StateHoliday'].map({'0':0, 0:0,'a':1,'b':2,'c':3}).astype(int)

model_random_forest.fit(X_train,y_train)
model_random_forest.predict(data_test)
data_test['Sales'] = model_random_forest.predict(data_test)

submission = data_test['Sales']
submission = submission.reset_index()
submission.head()
```

Chapter 3

Validation et Prédiction

Dans ce chapitre, nous verrons les résultats des différentes méthodes citées plus haut. Pour l'évaluation, nous nous sommes reposés sur 3 métriques : MSE, RMSPE et le coefficient R^2

3.1 Discussion des résultats

Après avoir entraîné les modèles sur l'échatillon d'apprentissage, nous les avons testé sur le base de données de test (celle de la valisation croisée), et nous avons noté les résultats en annexe B.

Si on retient la model score fourni par python plutôt que le R2 score que nous avons calculé à l'aide de la formule mathématique, nous parvenons à dire que les 3 modèles ont permis d'expliquer respectivement 86%, 94% et 98% de la variance totale.

En ce qui concerne les valeurs de l'erreur quadratique moyenne MSE, on y retient qu'avec le modèle de forêt d'arbre de décision, on est capable de réaliser de bonnes prédictions avec une faible dispersion relativement aux autres modèles. Toutefois, il reste à vérifier les valeurs du RMSPE qui réside un meilleur indicateur de la qualité de prédiction. En effet, l'erreur quadratique de prédiction présentent (respectivement pour les 3 modèles) en moyenne 21%, 13% et 5%

	Id	Sales
0	1	4352.41
1	857	4611.90
2	1713	4813.34
3	2569	5232.99
4	3425	0.00

Figure 3.1: Caption

des ventes réelles.

CONCLUSION : Comme les résultats le prouvent clairement, le modèle de forêt d'arbre de décision performe plus rigoureusement que les autres en terme d'exactitude et de réduction de variance inexpliquée.

3.2 Prédiction

On a fait la prédiction de ventes des données de test à l'aide du modèle de forêt d'arbre et on a obtenu une valeur de RMSPE de l'ordre de 0.179 ce qui n'est pas trop loin de la RMSPE du même modèle testé sur les données de validation. Cette valeur reste bonne tant qu'on est pas proches de 1. Une partie de la prédiction est donnée dans la figure 3.1.

3.3 ANNEXE B

```
DecisionTreeRegression  model
MSE score :  4254869.6645
RMSPE :  0.3890654598046343
Model score :  0.713437687759
R2 score : 0.6009
   Predictions  Actual
0  5295.060870    6610
1  7071.928070    8634
2  5747.169266   10509
3  5119.344000    3763
4  8710.200387    8956
------------------------------
```

```
LinearRegression  model
MSE score :  6586145.5376
RMSPE :  0.4563842942870771
Model score :  0.554039525312
R2 score : 0.1362
   Predictions  Actual
0  6818.337378    6610
1  8348.729662    8634
2  6126.955121   10509
3  5681.775637    3763
4  7227.770701    8956
------------------------------
```

```
RandomForestRegression  model
MSE score :  766419.4735
RMSPE :  0.14197621148193124
Model score :  0.948382217599
R2 score : 0.9454
   Predictions  Actual
0      6206.13    6610
1      8513.92    8634
2     10673.82   10509
3      4224.71    3763
4      8713.95    8956
------------------------------
```