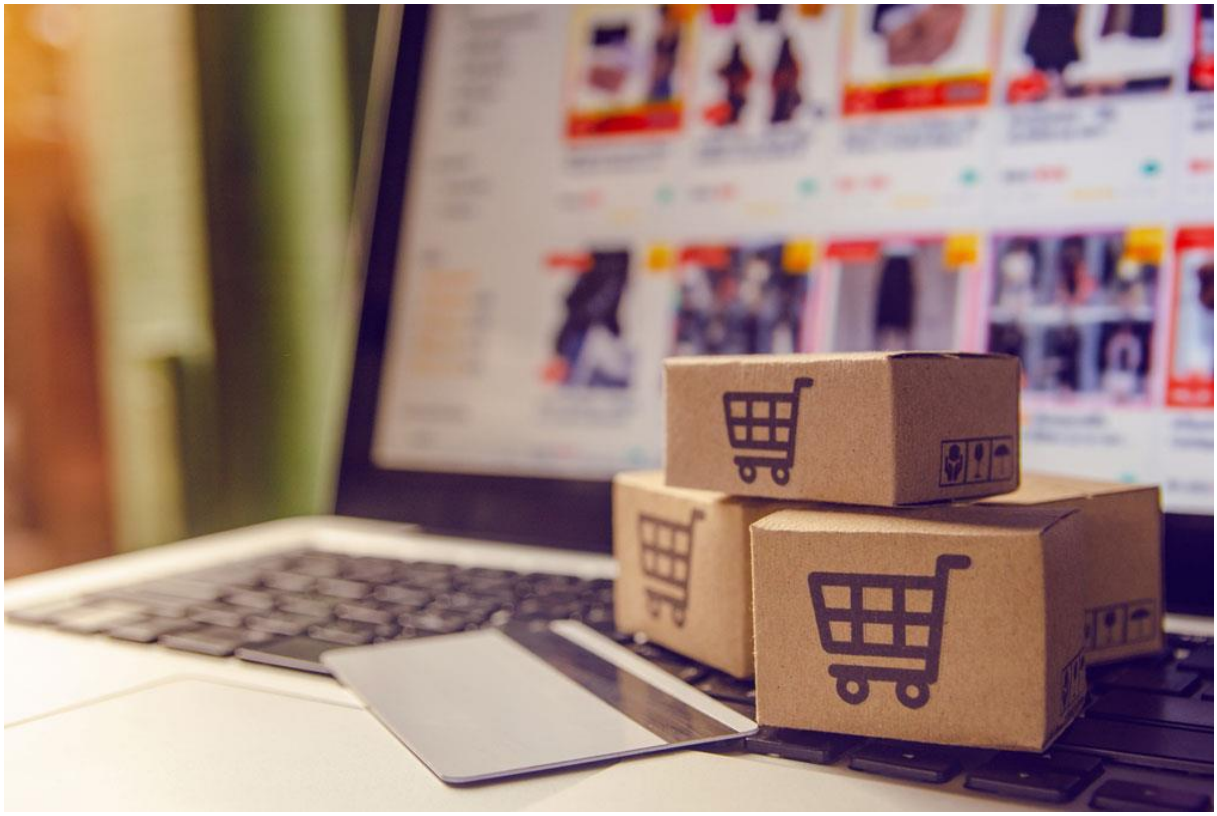


Advanced Machine Learning

TD 1 : Feature engineering

Sujet : Prédiction de la date d'expédition d'une commande

Yanis FADILI & Lies HAOUAS
Professeur : Denis OBLIN

Sommaire

1. Introduction
2. Importation des librairies et du dataset
3. Data exploration
4. Feature engineering
5. Data cleaning
6. Calcul de délais
7. Data visualization
8. Mise en place de modèles de machine learning

1. Introduction

L'objectif du projet est ici de travailler sur un jeu de données très simple composé de seulement 3 colonnes (dates de commande et d'expédition et expéditeur) afin de prédire la date d'expédition d'une commande selon les différents paramètres selon lesquels elle a été effectuée.
Le projet est développé en Python 3 sur Jupyter Notebook, majoritairement avec la bibliothèque Pandas.

2. Importation des librairies et du dataset

```
import pandas as pd
import numpy as np
import math
import datetime
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score as accuracy
from sklearn.metrics import average_precision_score as precision
```

```
df = pd.read_csv('/Users/mavericks/Downloads/logistique.csv',parse_dates=['datedecreationdec
```

3. Data exploration

```
df
```

	datedecreationdecommande	providerservice_id	dateexpe
0	2019-11-01 11:40:35.993	48	02/11/2019
1	2019-11-01 11:40:35.970	48	02/11/2019
2	2019-11-01 10:24:40.893	48	02/11/2019
3	2019-11-01 14:24:21.177	48	02/11/2019
4	2019-11-01 13:28:35.790	48	02/11/2019
...	...	...	...
572836	2019-12-16 06:12:25.217	48	17/12/2019
572837	2019-12-16 06:12:25.463	48	17/12/2019
572838	2019-12-15 22:32:36.320	48	17/12/2019
572839	2019-12-15 20:06:55.067	48	17/12/2019
572840	2019-12-15 19:17:10.123	48	17/12/2019

572841 rows × 3 columns

Aperçu du dataset ainsi que les types des variables

```
df.dtypes
```

```
datedecreationdecommande    datetime64[ns]
providerservice_id                   int64
dateexpe                            object
dtype: object
```

4. Feature engineering

Il est question dans cette section de créer des colonnes nous permettant par la suite de pouvoir faire de la dataviz, des calculs, ou encore de tester des modèles de machine learning sur le jeu de données.

```
df["datedecreationdecommande"] = df["datedecreationdecommande"].astype(str)
df["dateexpe"] = df["dateexpe"].astype(str)
```

```
df["year"] = df["datedecreationdecommande"].str[0:4]
df["month"] = df["datedecreationdecommande"].str[5:7]
df["day"] = df["datedecreationdecommande"].str[8:10]
df["hour"] = df["datedecreationdecommande"].str[11:16]
df["dateCreationdecommande"] = df["datedecreationdecommande"].str[0:10]
```

```
df["year"] = df["year"].astype(int)
df["month"] = df["month"].astype(int)
df["day"] = df["day"].astype(int)
df["dateCreationdecommande"] = df["dateCreationdecommande"].str.replace('-','/')

df['dateCreationdecommande'] = pd.to_datetime(df['dateCreationdecommande'],format='%Y/%m/%d'
df["dateexpe"] = pd.to_datetime(df['dateexpe'],format='%d/%m/%Y')
```

```
df['day_of_week'] = df['dateCreationdecommande'].dt.day_name()
```

On décide ici de convertir les données en chaines de caractères et de les splitter pour en extraire les informations relatives à la date, puis de les convertir en entiers et au format DateTime.

```
df.tail()
```

	datedecreationdecommande	providerservice_id	dateexpe	year	month	day	hour	dateCreationdecommande	day_of_week
6	2019-12-16 06:12:25.217	48	2019-12-17	2019	12	16	06:12	2019-12-16	Monday
7	2019-12-16 06:12:25.463	48	2019-12-17	2019	12	16	06:12	2019-12-16	Monday
8	2019-12-15 22:32:36.320	48	2019-12-17	2019	12	15	22:32	2019-12-15	Sunday
9	2019-12-15 20:06:55.067	48	2019-12-17	2019	12	15	20:06	2019-12-15	Sunday
0	2019-12-15 19:17:10.123	48	2019-12-17	2019	12	15	19:17	2019-12-15	Sunday

```
df.dtypes
```

```
datedecreationdecommande            object
providerservice_id                   int64
dateexpe                    datetime64[ns]
year                                 int64
month                                int64
day                                  int64
hour                                object
dateCreationdecommande      datetime64[ns]
day_of_week                         object
dtype: object
```

5. Data cleaning

Parfois, le jeu de données que l'on nous fournit n'est pas toujours « clean » et peut présenter des erreurs. Par exemple, lorsque la commande est expédiée avant qu'elle ne soit créée, on pensait que c'était une erreur alors que pas du tout car toutes ces commandes on était envoyé avant 19h, ce qui reste cohérent.

```
Entrée [95]: #On a remarqué qu'il y avait des lignes qui avait la même date de création et d'expédition
             #C'est normal car elles ont toutes été commandées avant 19heures donc c'est complètement normal, on fait l'hypothèse
             #que les fournisseurs travaillent jusqu'à minium 19h environ
             hourCommand = df[(df['datedecreationdecommande'] == df['dateexpe'])]["hour"].unique()
             hourCommand.sort()
             hourCommand

  Out[95]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                  17, 18])

Entrée [87]: min(df[df['datedecreationdecommande'] == df['dateexpe']]["hour"].unique())

  Out[87]: 0

Entrée [90]: max(df[(df['datedecreationdecommande'] == df['dateexpe'])]["hour"].unique())

  Out[90]: 18
```

Mais, il y a 34 lignes que l'on doit supprimer du jeu de données car elles ont des dates de créations de commandes faussaient comme des dates de création supérieur à la date d'expédition ou des dates de création datant de 1980 ce qui est complètement incohérent avec notre dateset.

	datedecreationdecommande	providerservice_id	dateexpe	year	month	day	hour	min	day_of_week	délai	isWeekend	isOnWorkShift
	cliquer pour faire défiler la sortie ; double-cliquer pour masquer	21	2019-11-02	2019	11	4	14	20	0	-2	1	1
37230	2019-11-05	21	2019-11-02	2019	11	5	11	32	1	-3	1	0
160743	2019-12-15	21	1980-01-05	2019	12	15	15	2	6	-1	0	1
373611	2019-12-15	16	1980-01-05	2019	12	15	13	48	6	-1	0	0
412989	2019-12-01	21	2019-07-26	2019	12	1	19	19	6	-1	0	0
454670	2019-12-07	21	1980-01-05	2019	12	7	23	59	5	-1	0	0
499183	2019-12-15	21	1980-01-05	2019	12	15	10	52	6	-1	0	0

6. Calculs de délais

Toujours dans la partie feature engineering, on choisit ici de faire un calcul essentiel dans la prédiction de la date d'expédition : le calcul du délai entre la date de commande et d'expédition :

```
Entrée [43]: df['délai'] = df['dateexpe'] - df['dateCreationdecommande']
             df
```

			02							days
4	2019-11-01 13:28:35.790	48	2019-11-02	2019	11	1	13:28	2019-11-01	Friday	1 days
...	...	...	...	...	...	...	...	...	...	...
572836	2019-12-16 06:12:25.217	48	2019-12-17	2019	12	16	06:12	2019-12-16	Monday	1 days
572837	2019-12-16 06:12:25.463	48	2019-12-17	2019	12	16	06:12	2019-12-16	Monday	1 days
572838	2019-12-15 22:32:36.320	48	2019-12-17	2019	12	15	22:32	2019-12-15	Sunday	2 days
572839	2019-12-15 20:06:55.067	48	2019-12-17	2019	12	15	20:06	2019-12-15	Sunday	2 days
572840	2019-12-15 19:17:10.123	48	2019-12-17	2019	12	15	19:17	2019-12-15	Sunday	2 days

558333 rows × 10 columns

Ici, la nouvelle colonne créée *délai* est une différence de temps : `délai timedelta64[ns]`. On convertit donc cette colonne en entier pour mieux exploiter cette donnée et l'utiliser comme variable à prédire :

```
df["délai"] = df["délai"].astype(str)
df["délai"] = df["délai"].str[0:2]
df["délai"] = df["délai"].astype(int)
df
```

1	2019-11-01 11:40:35.970	48	2019-11-02	2019	11	1	11:40	2019-11-01	
2	2019-11-01 10:24:40.893	48	2019-11-02	2019	11	1	10:24	2019-11-01	
3	2019-11-01 14:24:21.177	48	2019-11-02	2019	11	1	14:24	2019-11-01	
4	2019-11-01 13:28:35.790	48	2019-11-02	2019	11	1	13:28	2019-11-01	
...	...	...	...	...	...	...	...	...	
572836	2019-12-16 06:12:25.217	48	2019-12-17	2019	12	16	06:12	2019-12-16	
572837	2019-12-16 06:12:25.463	48	2019-12-17	2019	12	16	06:12	2019-12-16	
572838	2019-12-15 22:32:36.320	48	2019-12-17	2019	12	15	22:32	2019-12-15	
572839	2019-12-15 20:06:55.067	48	2019-12-17	2019	12	15	20:06	2019-12-15	

```
df["délai"].dtypes
```

```
dtype('int64')
```

On peut donc aisément calculer le délai moyen de prise en charge de la commande selon les différents expéditeurs :

```
#moyenne de délai par provider
df.groupby('providerservice_id')['délai'].mean().round(0)
```

```
providerservice_id
5     3.0
7     3.0
16    1.0
20    1.0
21    2.0
42    3.0
48    2.0
92    1.0
Name: délai, dtype: float64
```

```
#moyenne de délai par provider et par mois
df.groupby(['providerservice_id','month']).mean().round(0)
```

On peut ensuite comparer le délai d'expédition selon les différents mois, par exemple on remarque que le délai moyen en décembre est toujours supérieur ou égal à celui de novembre, dû à la période de fêtes :

```
#moyenne de délai par provider et par mois
df.groupby(['providerservice_id','month']).mean().round(0)
```

		year	day	délai
providerservice_id	month			
5	11	2019.0	26.0	2.0
	12	2019.0	7.0	3.0
7	11	2019.0	21.0	2.0
	12	2019.0	5.0	4.0
16	11	2019.0	18.0	1.0
	12	2019.0	8.0	1.0
20	11	2019.0	26.0	1.0
	12	2019.0	8.0	1.0
21	11	2019.0	19.0	1.0
	12	2019.0	6.0	3.0
42	11	2019.0	26.0	3.0
	12	2019.0	6.0	4.0
48	11	2019.0	20.0	2.0
	12	2019.0	6.0	3.0
92	11	2019.0	20.0	1.0
	12	2019.0	11.0	1.0

On remarque également que l'heure à laquelle la commande est effectuée influe sur le délai de prise en charge :

```
df.groupby(['providerservice_id','month','day','hour']).mean().round(0).head(20)
```

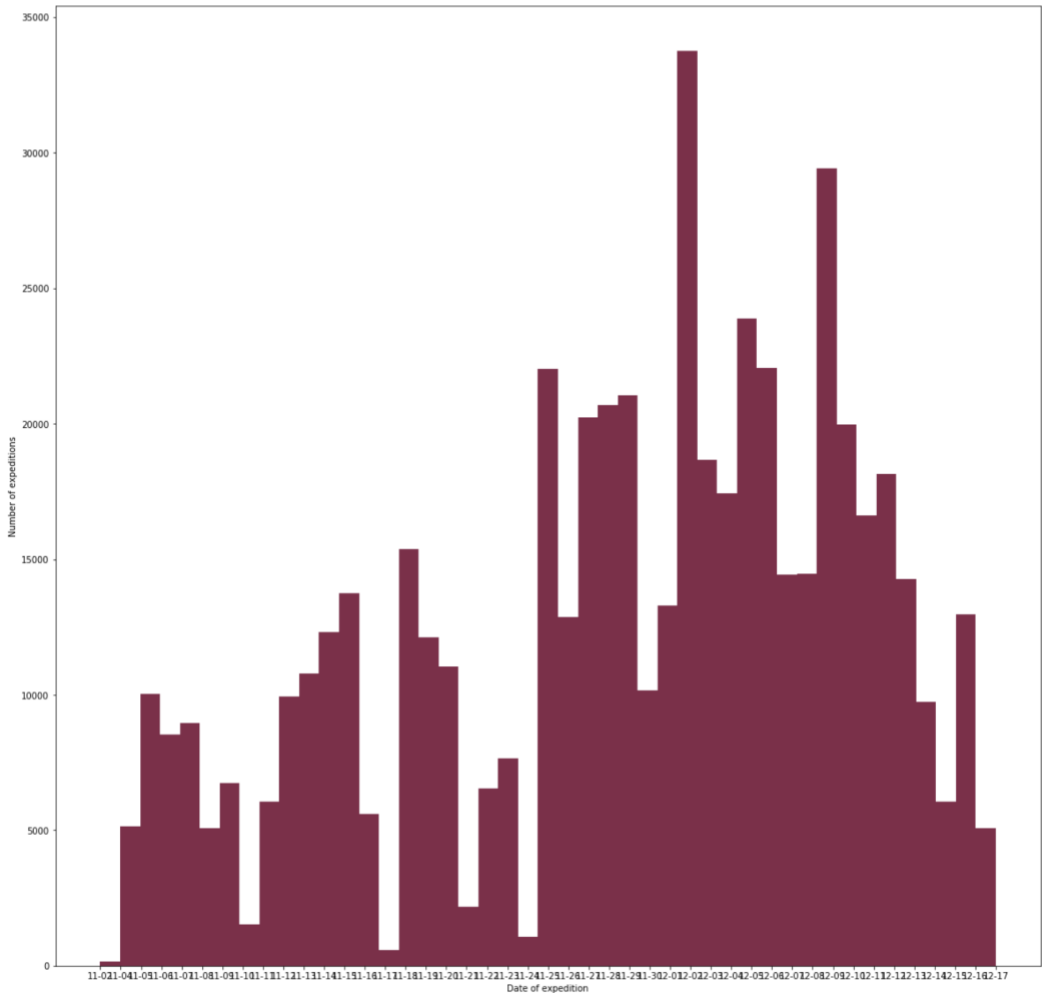
				délai
providerservice_id	month	day	hour	
5	11	23	22:32	3.0
		24	18:29	2.0
		25	16:53	2.0
		26	18:23	2.0
		28	10:00	1.0
		29	05:01	2.0
		30	16:32	3.0

7. Data visualisation

Outre les traitements informatiques, l'une des meilleures façons d'analyser la donnée est de la visualiser. Ne dit-on pas qu'une image vaut mille mots ?
On obtient ainsi le nombre de commandes pour chaque date :

```
plt.hist(df['dateexpe'], bins=45,color="#842949")
#plt.xticks(df['dateexpe'].unique())
plt.xticks(df['dateexpestr'].unique())
plt.xlabel("Date of expedition")
plt.ylabel("Number of expeditions")
```

```
Text(0, 0.5, 'Number of expeditions')
```



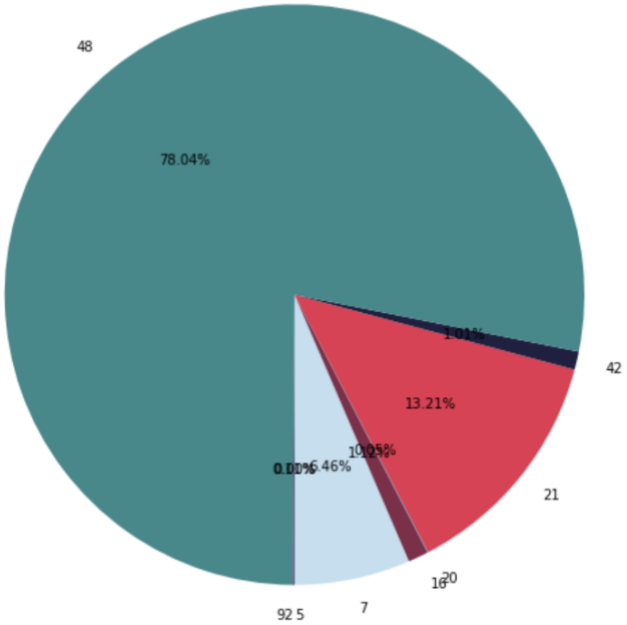
On remarque qu'il y a beaucoup de commande mi-décembre et fin décembre (période de fêtes).

Ensuite, on peut remarquer que les commandes sont majoritairement expédiées par le provider ayant l'id 48 (potentiellement Colissimo) :

```
nbcommandepro = df.groupby('providerservice_id').size()
nbcommandepro
```

```
providerservice_id
5         18
7      36058
16      6264
20       263
21     73783
42      5620
48    435713
92       614
dtype: int64
```

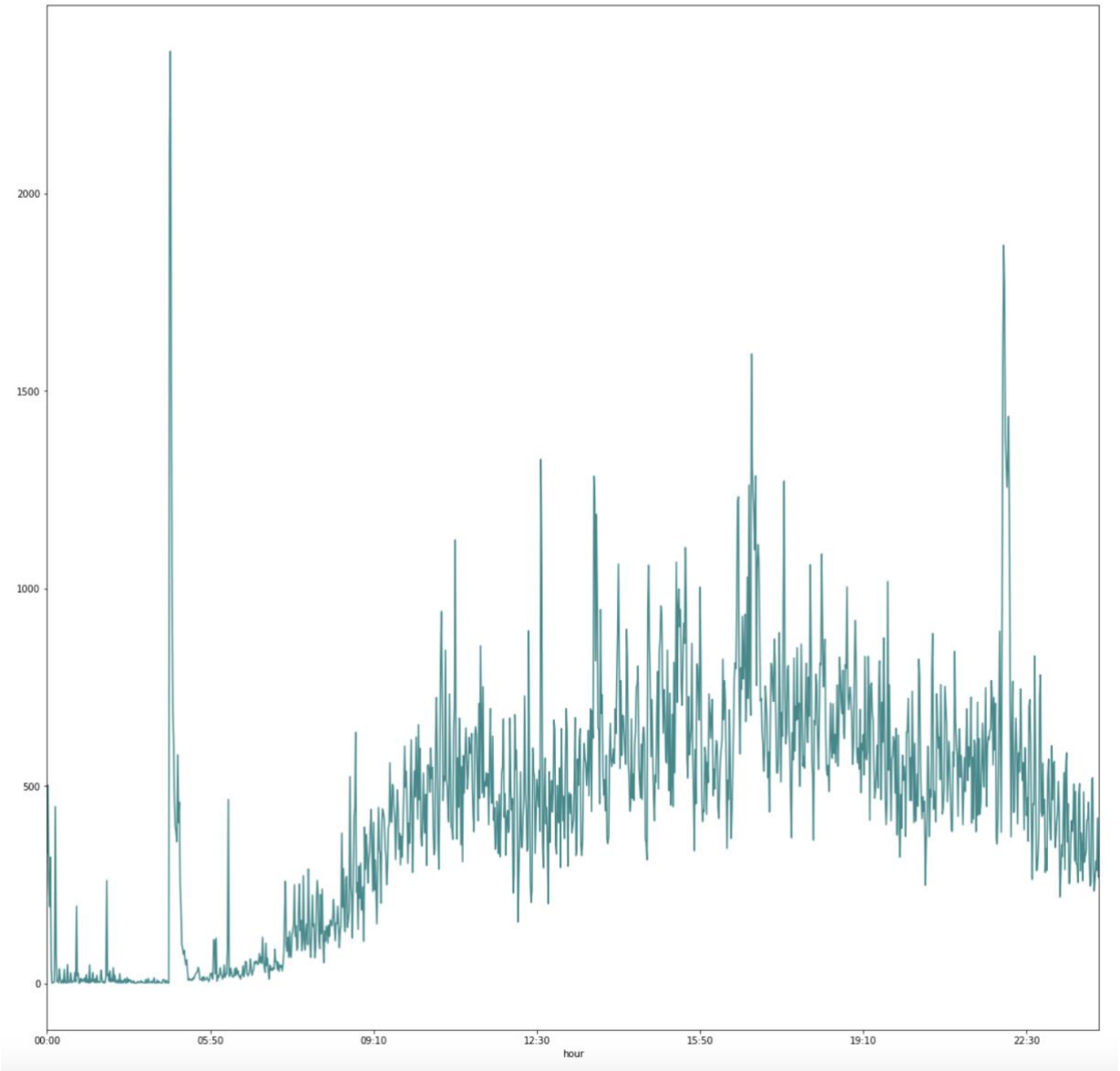
```
plt.figure(figsize = (7,7))
colors = ["#88ccf1","#c1dff0","#842949","#3587a4","#e83151","#1f2041","#2d898b","#4b3f72"]

y = nbcommandepro['dateexpe'].to_numpy()
mylabels = exp_id

plt.pie(y,
        labels = mylabels,
        colors =colors, # Color of each section
        autopct = "%0.2f%%", # Show data in persentage for with 2 decimal point
        shadow = False, # Showing shadow of pie chart
        radius = 1.4, # Radius to increase or decrease the size of pie chart
        startangle = 270) # Start angle of first section
plt.show()
```



Enfin, on peut visualiser les heures à laquelle les commandes sont souvent effectuées, en l'occurrence tôt le matin et tard le soir :

```
heurecom.plot(color=("#2D898B"))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a845c73d0>
```



8. Mise en place de modèles de machine learning

Dans un premier temps, on a préparé les différentes features pour le modèle comme la feature day_of_week qui est passé dans un encodeur le one hot encoding pour lui donner plus de sens et avoir une meilleure efficacité.

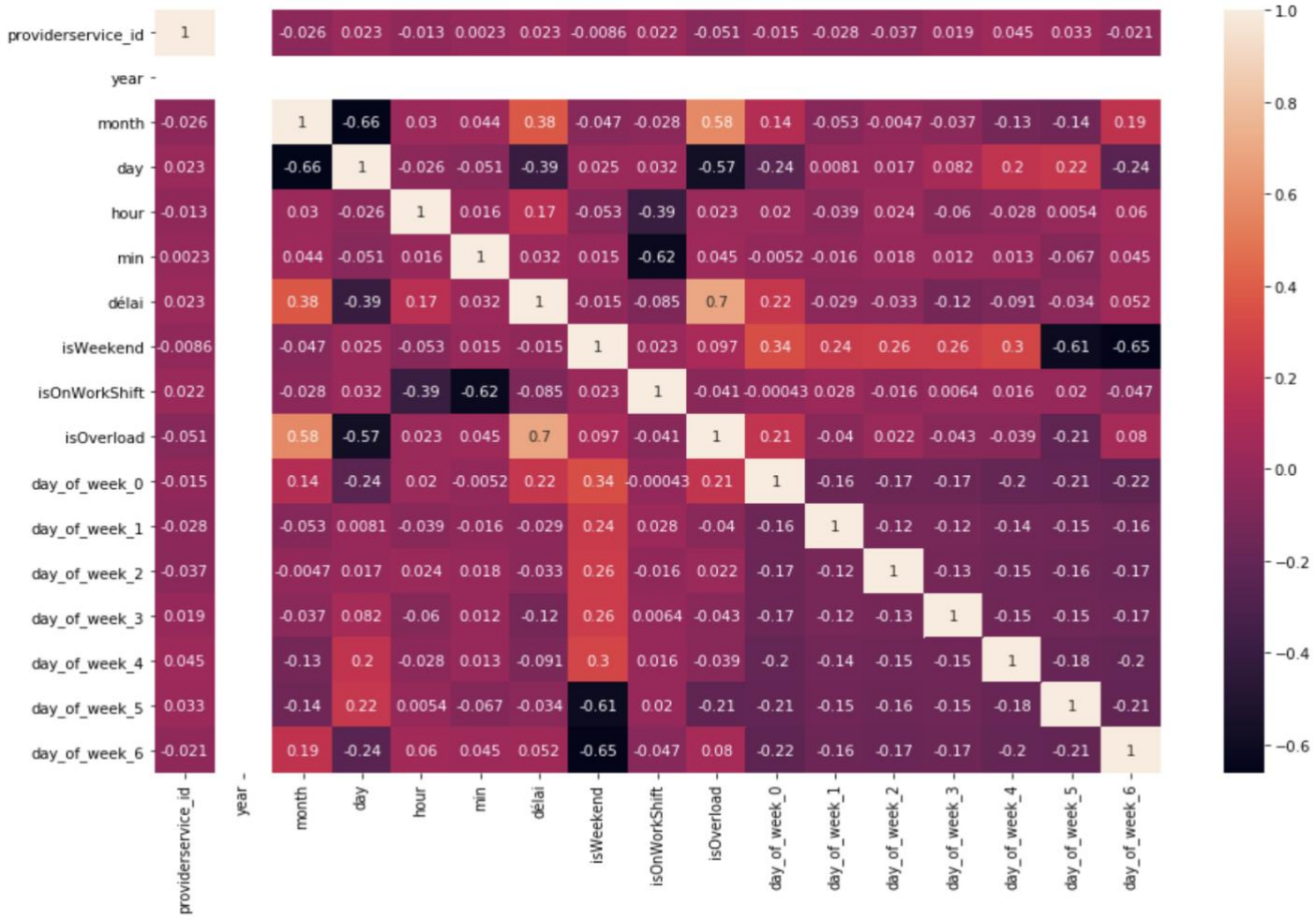
```
df['day_of_week'] =  df['day_of_week'].astype('category')
df = pd.get_dummies(data=df,columns=['day_of_week'])
df
```

day_of_week_0	day_of_week_1	day_of_week_2	day_of_week_3	day_of_week_4	day_of_week_5	day_of_w
0	0	0	0	1	0	
0	0	0	0	1	0	
0	0	0	0	1	0	
0	0	0	0	1	0	
0	0	0	0	1	0	
...	...	...	...	...	...	
1	0	0	0	0	0	
1	0	0	0	0	0	
0	0	0	0	0	0	
0	0	0	0	0	0	
0	0	0	0	0	0	

On a aussi créé 3 nouvelles features qui nous semblait intéressant de rajouter, comme si la commande a été passé un week-end, ou si elle a été passée pendant les heures de travails et si c'était pendant une période creuse (vacances, black-Friday, Noël).

isWeekend	isOnWorkShift	isOverload
1	0	1
1	0	1
1	1	1
1	1	1
1	1	1

On a ensuite fait une matrice de corrélation pour savoir quelle features influent le plus sur la variable à prédire :



Nous avons gardé ces features pour le train et le test :

```
X = df[["providerservice_id","month","day","hour","isWeekend","isOnWorkShift","isOverload","day_of_week_0","day_of_w
y = df['délai']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20)
```

Puis, nous avons réparti notre dataset en train et test avec une répartition de 80% pour le train et 20% pour le test, c'est la répartion qui nous a semblé la plus efficace.

Nous avons testé différents modèles sur les données comme le Random forest, le Bagging, Linear Regression, Decision Tree. La meilleure précision a été obtenu avec Decision Tree avec une précision de 76%.

```
from sklearn.tree import DecisionTreeRegressor
simple_tree = DecisionTreeRegressor(random_state=0)
simple_tree.fit(X_train, y_train)
simple_tree_pred = simple_tree.predict(X_test)
simple_tree_score = simple_tree.score(X_test,y_test)
simple_tree_score
```

```
.7611618394702566
```