UNIVERSITE DE KINSHASA

FACULTE DE SCIENCES ET TECHNOLOGIE
DÉPARTEMENT DE MATHEMATIQUES, STATISTIQUES ET INFORMATIQUE
P.O BOX190 KINSHASA-XI

TP D'ANALYSE DE DONNEES

REALISE PAR :

BOKAU ISALIEMA Isaac
BUMBA KALEKO Victorine
INANA KABISA Exaucée
KABUYA KABONGO Dan
KADIMA KAPIAMBA Eric
KINGOLO BENI Armel
KWEME GRACE Augustine
MAVUNGU LANDU Grace
NSINGA LUZINGU Emmanuel
PHAKA NZUZI Grace

L2 GESTION

PROFESSEUR : KASORO MULENDA
ASSISTANT : MANDIYA REAGAN

ANNEE ACADEMIQUE

2023-2024

Introduction

Le secteur de la restauration à Kinshasa connaît une croissance rapide, portée par l'évolution des habitudes de consommation des citadins. Face à un rythme de vie de plus en plus trépidant, un grand nombre de professionnels et d'étudiants préfèrent recourir aux services de livraison de repas en ligne. Ces services leur permettent de commander des plats de leurs restaurants préférés directement depuis leur smartphone, sans avoir à se déplacer.

FoodHub, une société d'agrégation de produits alimentaires, s'inscrit dans cette dynamique en mettant à la disposition des clients une plateforme qui centralise les offres de nombreux restaurants. À travers son application, FoodHub simplifie le processus de commande et de livraison de repas, facilitant l'accès des consommateurs aux restaurants tout en offrant un service de livraison fiable.

Toutefois, dans un environnement concurrentiel, FoodHub doit constamment améliorer l'expérience de ses clients pour se démarquer. Pour cela, l'analyse des données accumulées via ses interactions avec les clients devient une priorité stratégique. En tant que Data Scientist de FoodHub, notre mission est de répondre aux questions clés posées par l'équipe Data Science, afin de dégager des insights précieux pour optimiser les opérations de la société.

Ce rapport propose une analyse complète des données collectées, en suivant les différentes étapes du cycle de vie de la science des données. Nous nous concentrerons sur la demande des différents restaurants, la performance des livreurs et les préférences des clients, dans le but d'améliorer l'expérience utilisateur globale et de renforcer la compétitivité de FoodHub.

Le cycle de vie en science des données comprend :

- Définition de scope du projet et du problème
- Récolte des données (Data Analysis)
- Préparation des données (Prétraitement)
- Traitement (Exploratory Data Analysis)
- Machine Learning
- Deploiment

1) Définition de scope du projet et du problème

Objectif du projet : Analyser les commandes des clients pour identifier les tendances, améliorer la gestion de la livraison et optimiser les services. Voici les objectifs spécifiques :

- Identifier les restaurants et types de cuisines les plus populaires.
- Analyser les performances des restaurants basées sur le temps de préparation et de livraison.
- Évaluer l'impact du jour de la semaine sur les commandes.
- Comprendre la relation entre le coût, la note, et les délais de préparation/livraison.

2) Récolte des données (Data Analysis)

La récolte des données en data science est une étape cruciale où l'on collecte des informations pertinentes pour résoudre un problème spécifique. Dans ce cas, les données proviennent d'un fichier CSV, un format structuré souvent utilisé pour stocker des tableaux de données.

Les outils courants utilisés pour cette tâche incluent Python (avec des bibliothèques comme pandas) ou SQL pour manipuler et charger ces données.

Entrée [1]:
```
#Importation des bibliothèques
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
C:\Users\hp\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required fo
r this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Entrée [3]:
```
data = pd.read_csv("C:\\Users\\hp\\3D Objects\\foodhub_order.csv")
```

Entrée [4]:
```
#Affichage de 5 lignes au choix
data.sample(5)
```

Out[4]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
128	1476826	53543	CafÎ© China	Chinese	24.30	Weekend	Not given	20	23
1106	1477250	41409	Nobu Next Door	Japanese	19.35	Weekend	4	28	30
913	1477043	142356	Blue Ribbon Sushi	Japanese	6.74	Weekend	4	22	28
10	1477895	143926	Big Wong Restaurant □_¤¾Ñ¼	Chinese	5.92	Weekday	Not given	34	28
1833	1478234	102620	Balthazar Boulangerie	French	12.23	Weekend	4	35	23

```
#type des données
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   order_id               1898 non-null   int64
 1   customer_id            1898 non-null   int64
 2   restaurant_name        1898 non-null   object
 3   cuisine_type           1898 non-null   object
 4   cost_of_the_order      1898 non-null   float64
 5   day_of_the_week        1898 non-null   object
 6   rating                 1898 non-null   object
 7   food_preparation_time  1898 non-null   int64
 8   delivery_time          1898 non-null   int64
dtypes: float64(1), int64(4), object(4)
memory usage: 133.6+ KB
```

3) Préparation des données (Prétraitement)

La préparation des données, ou prétraitement, est une étape essentielle en data science qui consiste à nettoyer et transformer les données brutes pour les rendre exploitables par les modèles d'analyse. Dans ce cas, avec un fichier CSV, cela implique de traiter les valeurs manquantes, les doublons, et les erreurs de formatage. On standardise également les données en s'assurant que toutes les variables soient dans un format compatible (par exemple, transformer des dates en un format temporel standard)

a- Gestion des valeurs manquantes

```
# Gestion des valeurs manquantes
# Remplacer les 'Not given' par des NaN
data['rating'].replace('Not given', None, inplace=True)

# Conversion de la colonne 'rating' en float
data['rating'] = pd.to_numeric(data['rating'])

# Aperçu des valeurs manquantes
print(data.isnull().sum())
```

```
order_id                   0
customer_id                0
restaurant_name            0
cuisine_type               0
cost_of_the_order          0
day_of_the_week            0
rating                   736
food_preparation_time      0
delivery_time              0
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   order_id               1898 non-null   int64  
 1   customer_id            1898 non-null   int64  
 2   restaurant_name        1898 non-null   object 
 3   cuisine_type           1898 non-null   object 
 4   cost_of_the_order      1898 non-null   float64
 5   day_of_the_week        1898 non-null   object 
 6   rating                 1162 non-null   float64
 7   food_preparation_time  1898 non-null   int64  
 8   delivery_time          1898 non-null   int64  
dtypes: float64(2), int64(4), object(3)
memory usage: 133.6+ KB
```

b- Remplacement des valeurs manquantes dans 'rating' par la moyenne

```
# Remplacer les valeurs manquantes dans 'rating' par la moyenne
data['rating'].fillna(data['rating'].mean(), inplace=True)

# Vérifier la présence de valeurs manquantes
print(data.isnull().sum())
```

```
order_id                0
customer_id             0
restaurant_name         0
cuisine_type            0
cost_of_the_order       0
day_of_the_week         0
rating                  0
food_preparation_time   0
delivery_time           0
dtype: int64
```

4) Traitement (Exploratory Data Analysis)

Le traitement des données, ou **Exploratory Data Analysis (EDA)**, est une étape clé qui consiste à analyser les données de manière visuelle et statistique afin de comprendre leur structure, leurs tendances et leurs relations avant de construire des modèles. Cela inclut la génération de statistiques descriptives (moyennes, médianes, écarts-types) et l'identification des distributions des variables. Des visualisations comme les histogrammes, les graphiques en boîte (boxplots) et les nuages de points (scatter plots) sont souvent utilisées pour détecter des anomalies, des valeurs extrêmes (outliers) et des corrélations entre variables. L'EDA permet aussi de poser des hypothèses initiales et d'affiner la compréhension des données pour orienter les prochaines étapes du projet.

a- Tableau Synthétique

- Pour les valeurs numériques

```
# Statistiques descriptives de base pour les variables quantitatives
data.describe()
```

	order_id	customer_id	cost_of_the_order	rating	food_preparation_time	delivery_time
count	1.898000e+03	1898.000000	1898.000000	1898.000000	1898.000000	1898.000000
mean	1.477496e+06	171168.478398	16.498851	4.344234	27.371970	24.161749
std	5.480497e+02	113698.139743	7.483812	0.580071	4.632481	4.972637
min	1.476547e+06	1311.000000	4.470000	3.000000	20.000000	15.000000
25%	1.477021e+06	77787.750000	12.080000	4.000000	23.000000	20.000000
50%	1.477496e+06	128600.000000	14.140000	4.344234	27.000000	25.000000
75%	1.477970e+06	270525.000000	22.297500	5.000000	31.000000	28.000000
max	1.478444e+06	405334.000000	35.410000	5.000000	35.000000	33.000000

- Pour les valeurs quantitatives

```
#Selection des colonnes du type object
data_string = data.select_dtypes('object')
```

```
#Tableu synthétique pour les objects
data_string.describe()
```

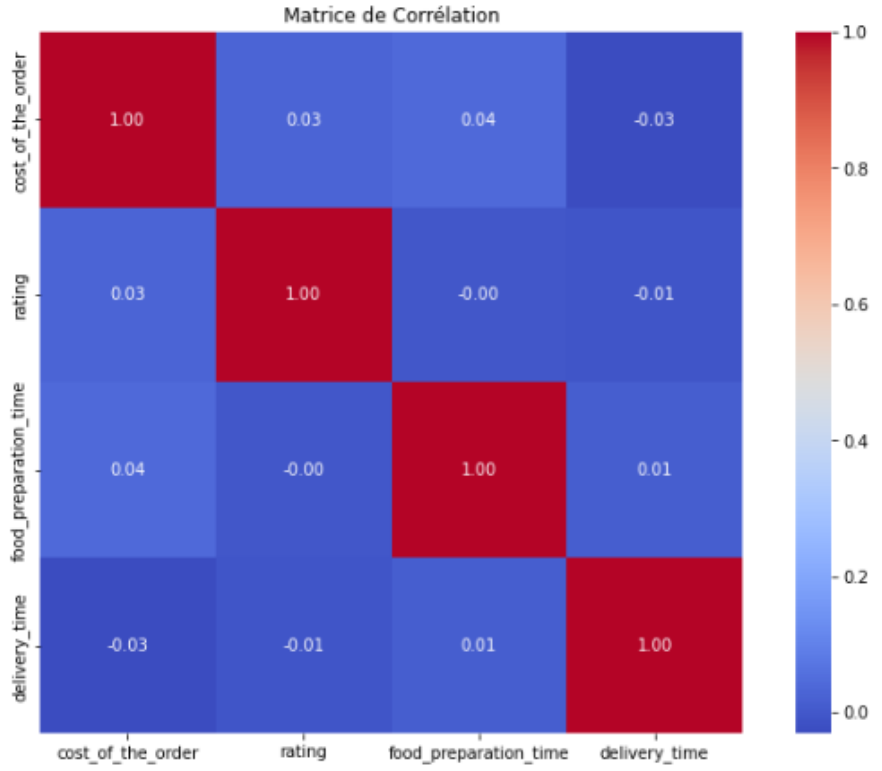
	restaurant_name	cuisine_type	day_of_the_week
count	1898	1898	1898
unique	178	14	2
top	Shake Shack	American	Weekend
freq	219	584	1351

b- Matrice de corrélation

La matrice de corrélation est un tableau qui montre les relations linéaires entre variables numériques. Chaque valeur varie de -1 (corrélation négative) à 1 (corrélation positive), avec 0 indiquant l'absence de corrélation. Cet outil permet de repérer les variables fortement corrélées, ce qui aide à réduire les redondances et à améliorer la qualité des modèles prédictifs. Un coefficient de corrélation qui vaut **0 virgule quelque chose** indique une corrélation positive faible à modérée entre deux variables. Plus précisément

- **Entre 0 et 0,3** : la corrélation est faible, signifiant que la relation entre les deux variables est peu significative.
- **Entre 0,3 et 0,7** : la corrélation est modérée, signifiant qu'il existe une relation visible, mais elle n'est pas très forte.
- **Entre 0,7 et 1** : la corrélation est forte, indiquant que les deux variables évoluent de manière similaire.

```
: # Matrice de corrélation
plt.figure(figsize=(12, 8))
corr = data[['cost_of_the_order', 'rating', 'food_preparation_time', 'delivery_time']].corr()
sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm', square=True)
plt.title('Matrice de Corrélation')
plt.show()
```



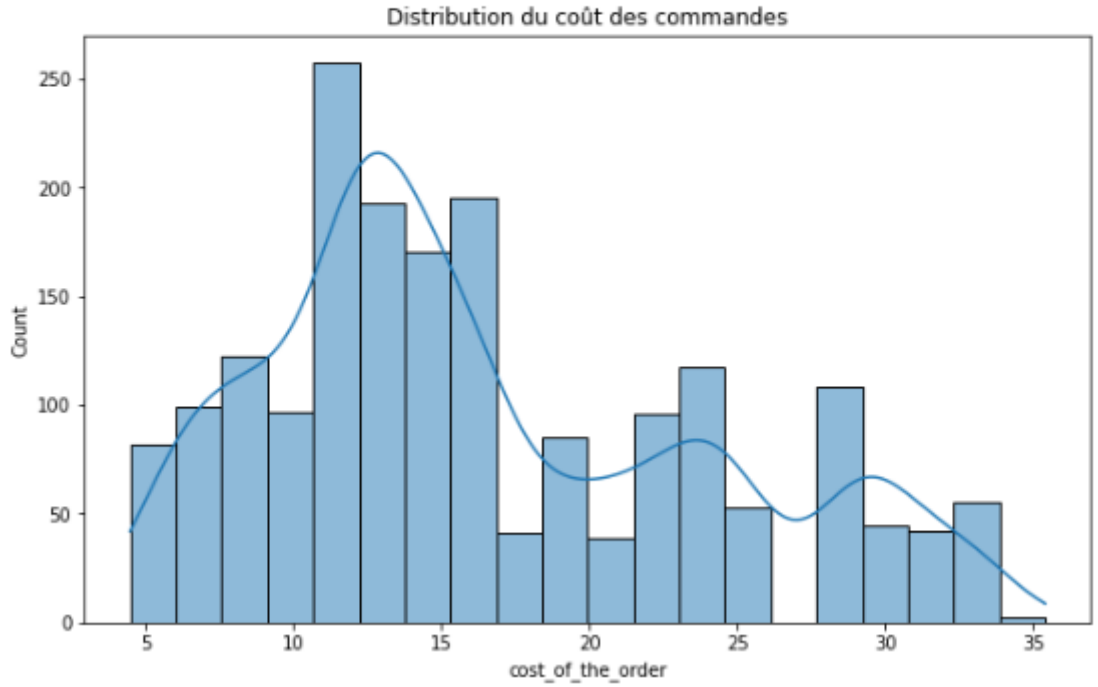
c- Histogramme

Un histogramme est un graphique utilisé pour visualiser la distribution d'une variable numérique. Il se compose de barres verticales où chaque barre représente la fréquence (ou le nombre d'occurrences) d'une plage de valeurs dans le dataset. L'axe horizontal (x) montre les intervalles ou classes de la variable, tandis que l'axe vertical (y) indique le nombre d'observations dans chaque intervalle.

Les histogrammes permettent de comprendre la répartition des données (par exemple, si elles sont symétriques, asymétriques, normales) et d'identifier des caractéristiques importantes comme la dispersion, les valeurs extrêmes ou les lacunes dans les données.

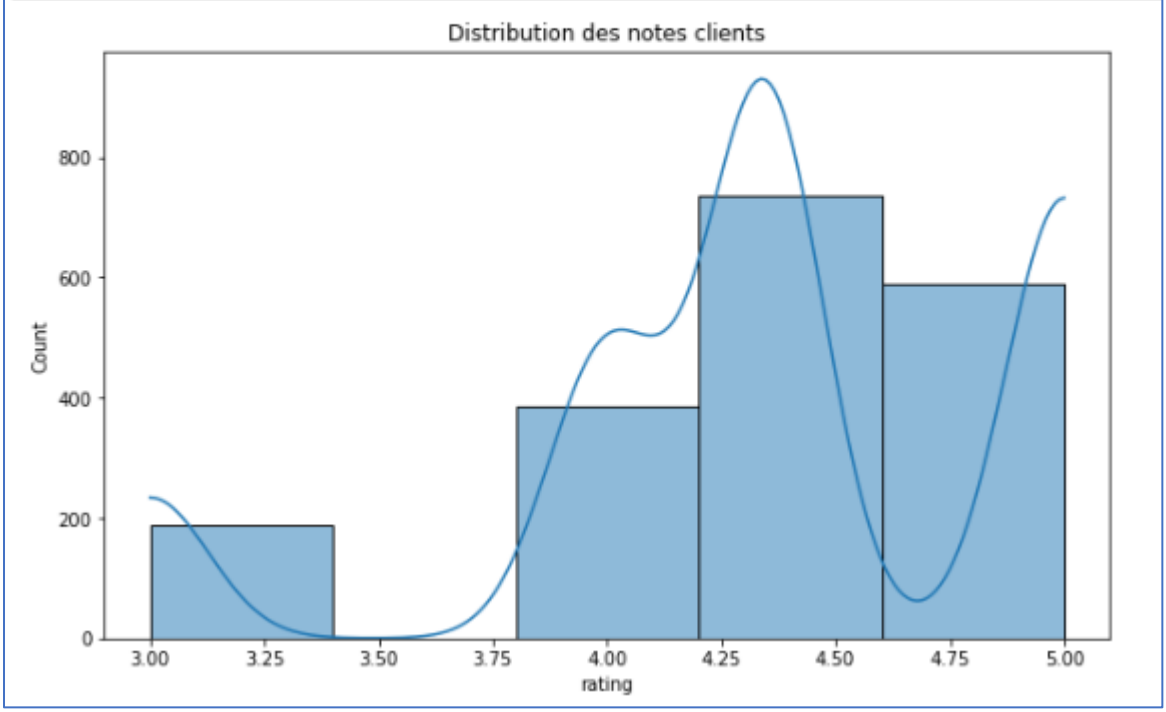
- Distribution du coût des commande

```
: # Distribution du coût des commandes
plt.figure(figsize=(10,6))
sns.histplot(data['cost_of_the_order'], bins=20, kde=True)
plt.title('Distribution du coût des commandes')
plt.show()
```



- Distribution des notes données par les clients

```python
# Distribution des notes données par les clients
plt.figure(figsize=(10,6))
sns.histplot(data['rating'], bins=5, kde=True)
plt.title('Distribution des notes clients')
plt.show()
```



d- Boxplot

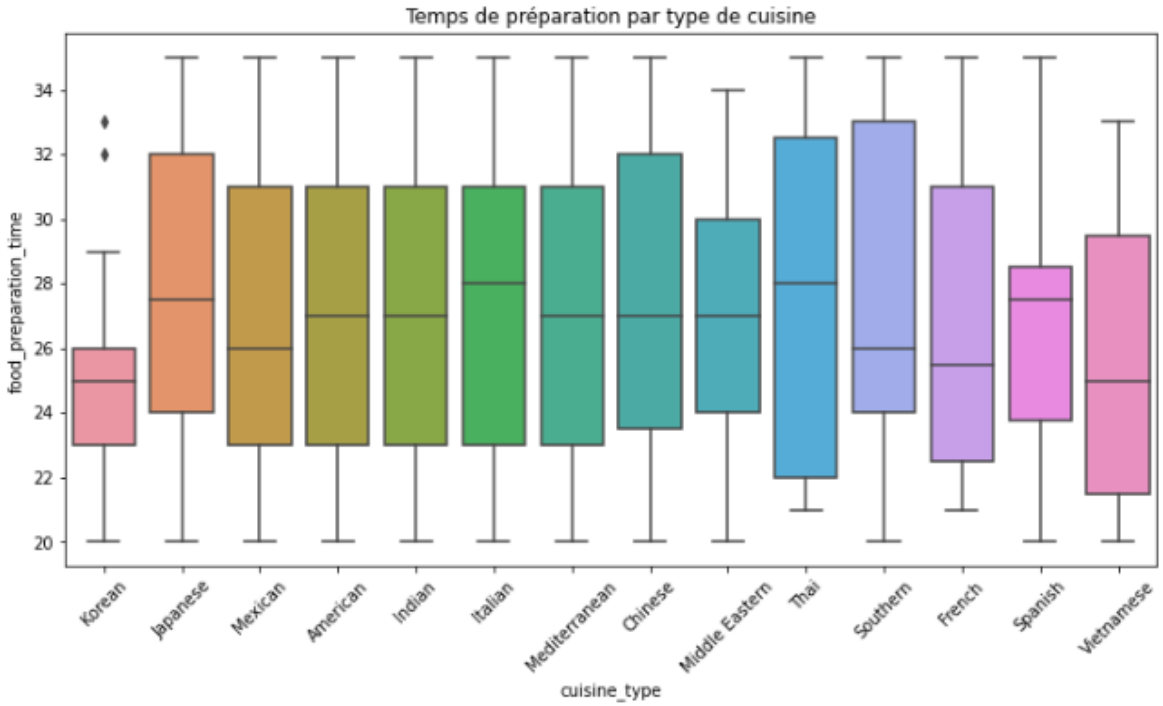
Un **boxplot**, ou diagramme en boîte, est une représentation graphique qui résume la distribution d'une variable numérique en mettant en évidence ses quartiles, ses valeurs extrêmes et ses éventuels outliers (valeurs aberrantes).

Éléments clés d'un boxplot :

- **Boîte** : Représente l'intervalle interquartile (IQR), qui va du premier quartile (Q1, 25e percentile) au troisième quartile (Q3, 75e percentile). La longueur de la boîte indique la dispersion des valeurs centrales.
- **Médiane** : Une ligne à l'intérieur de la boîte indique la médiane (Q2, 50e percentile) de la distribution.
- **Moustaches** : Des lignes s'étendant à partir des côtés de la boîte (moustaches) montrent l'étendue des données, souvent jusqu'à 1,5 fois l'IQR.
- **Outliers** : Les points situés en dehors des moustaches sont représentés par des cercles ou des étoiles, indiquant des valeurs aberrantes.

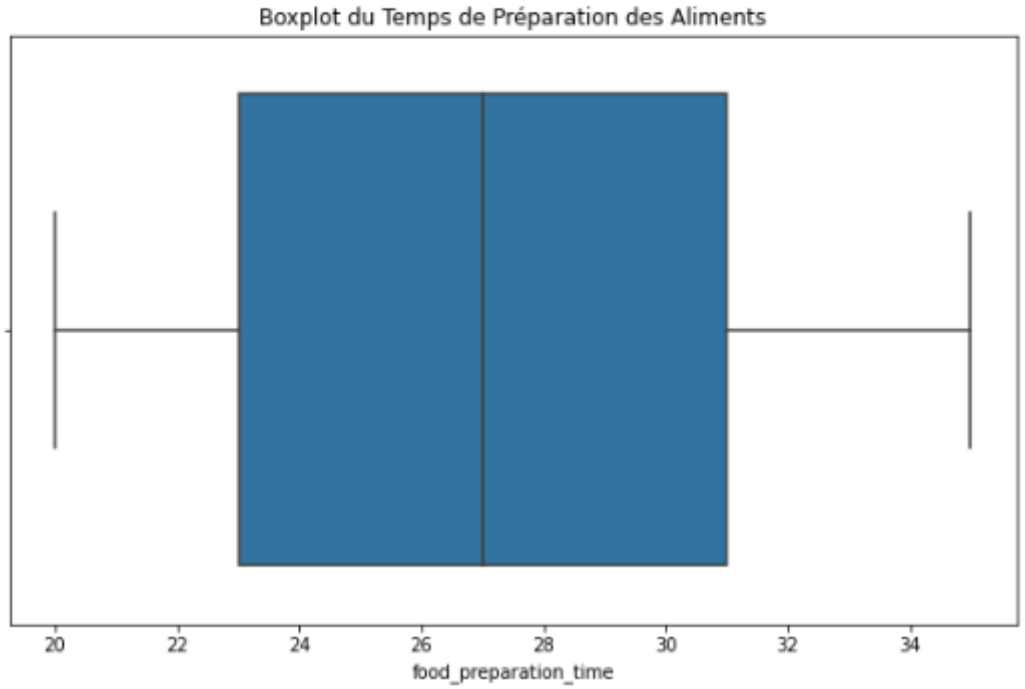
- Boxplot du temps de préparation en fonction du type de cuisine

```
: # Boxplot du temps de préparation en fonction du type de cuisine
  plt.figure(figsize=(12,6))
  sns.boxplot(x='cuisine_type', y='food_preparation_time', data=data)
  plt.title('Temps de préparation par type de cuisine')
  plt.xticks(rotation=45)
  plt.show()
```



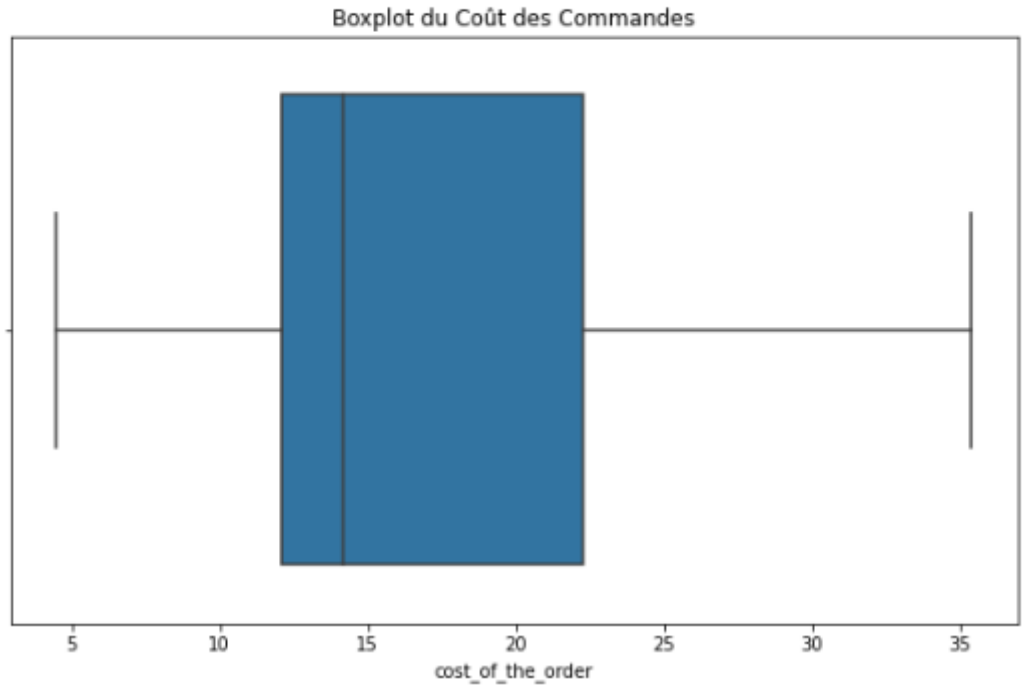
- Boxplot pour le temps de préparation des aliments

```
# Boxplot pour le temps de préparation des aliments
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['food_preparation_time'])
plt.title('Boxplot du Temps de Préparation des Aliments')
plt.show()
```



- Boxplot pour le coût

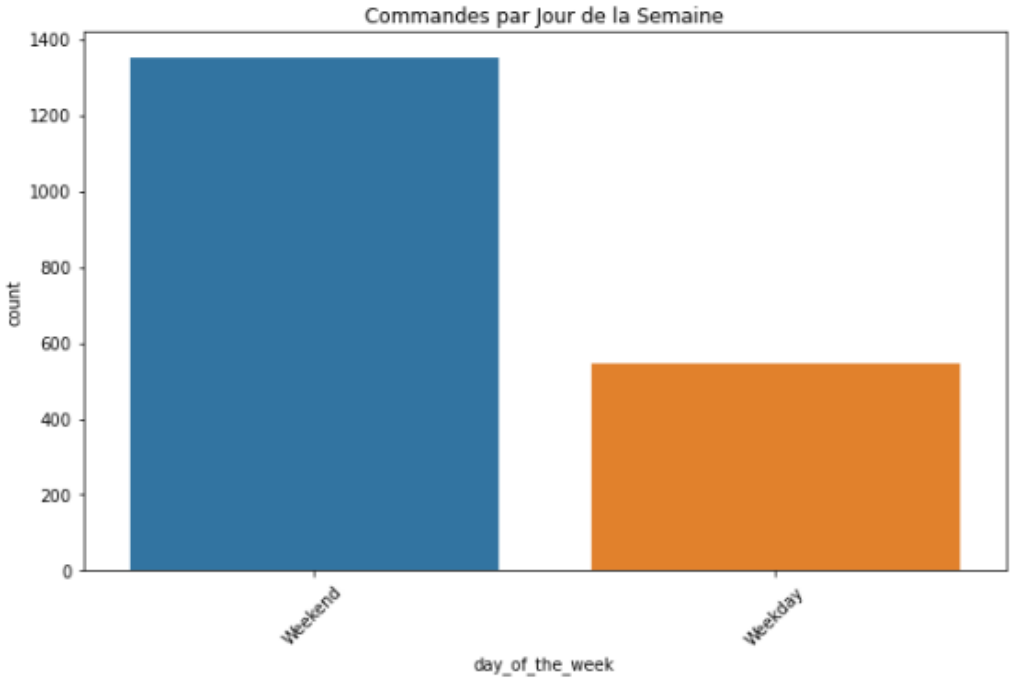
```
]: # Boxplot pour le coût
   plt.figure(figsize=(10, 6))
   sns.boxplot(x=data['cost_of_the_order'])
   plt.title('Boxplot du Coût des Commandes')
   plt.show()
```



- Graphique en barres

Un graphique en barres est un outil visuel qui permet de comparer des quantités à l'aide de barres rectangulaires. Chaque barre représente une catégorie ou un groupe, et sa longueur ou sa hauteur est proportionnelle à la valeur qu'elle représente.

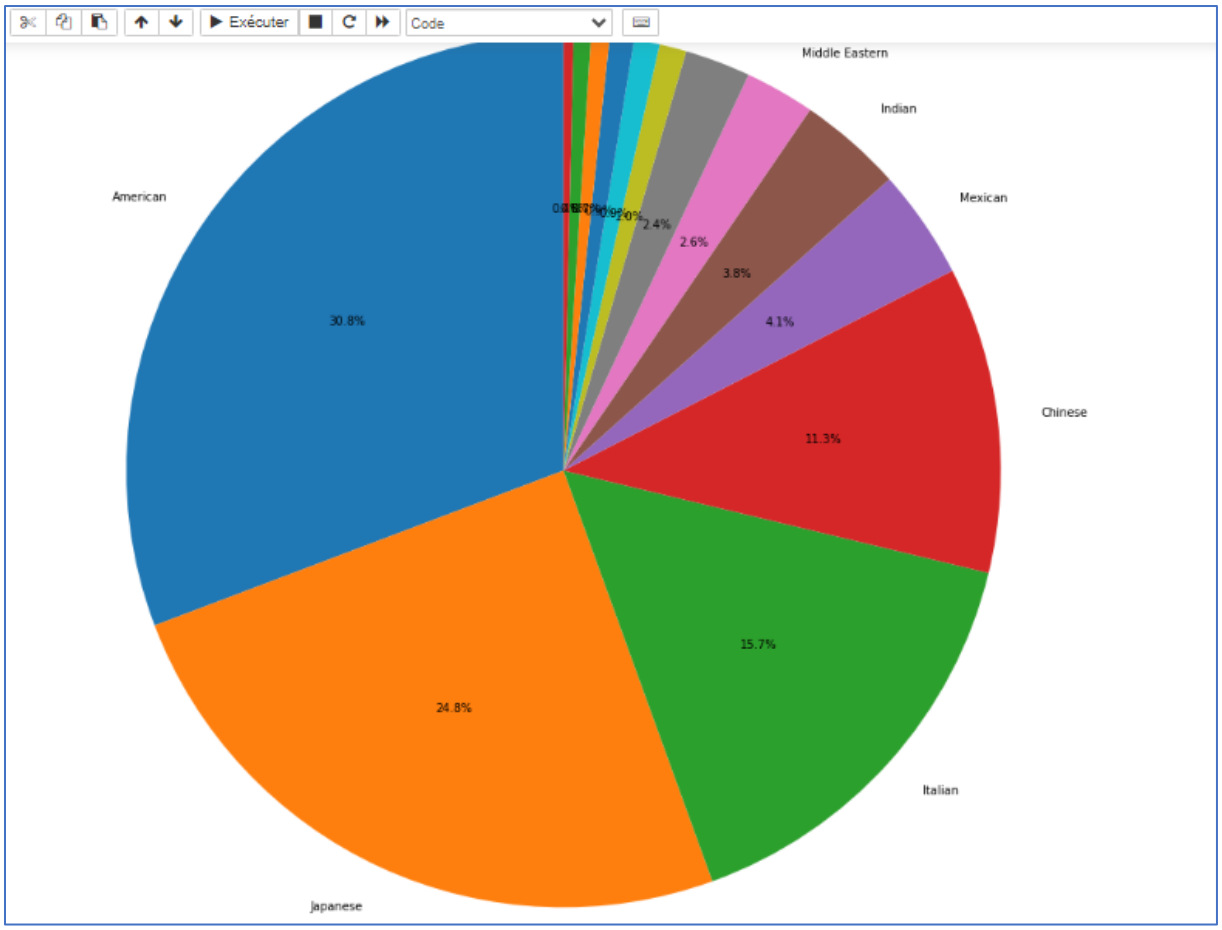
```
: # Graphique en barres pour le jour de la semaine
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='day_of_the_week')
plt.title('Commandes par Jour de la Semaine')
plt.xticks(rotation=45)
plt.show()
```



- Diagramme circulaire

Un diagramme circulaire (ou **camembert**) est un graphique circulaire utilisé pour montrer la répartition ou la proportion des différentes parties d'un ensemble. Chaque portion du cercle représente une catégorie, et la taille de chaque segment est proportionnelle à sa contribution au total.

```
# Création du diagramme circulaire
cuisine_counts = data['cuisine_type'].value_counts()
plt.figure(figsize=(20, 20))
plt.pie(cuisine_counts, labels=cuisine_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution des Types de Cuisine')
plt.axis('equal')  # Assure que le diagramme est circulaire
plt.show()
```



5) Machine Learning

Nous avons utilisé K-Means pour segmenter le dataset en 2 groupes.

Les résultats du K-Means sur le dataset **FoodHub** ont permis de séparer les commandes en deux clusters :

- **Cluster 0** : Ce groupe correspond à des commandes généralement moins coûteuses, avec des temps de préparation variés (souvent plus longs) et des temps de livraison parfois plus élevés. Par exemple, des commandes avec des coûts autour de 5 à 16 unités, comme celles du restaurant Barbounia ou Anjappar Chettinad.
- **Cluster 1** : Ce groupe regroupe des commandes plus coûteuses, avec des temps de livraison plus rapides. Par exemple, des commandes coûtant plus de 24 unités, comme celles du restaurant Hangawi ou The Meatball Shop.

```
]: from sklearn.preprocessing import StandardScaler

   # Sélection des colonnes pertinentes
   features = data[['cost_of_the_order', 'food_preparation_time', 'delivery_time']]

   # Normalisation des données
   scaler = StandardScaler()
   features_scaled = scaler.fit_transform(features)
```

```
]: from sklearn.cluster import KMeans

   # Définir le nombre de clusters
   n_clusters = 2
   kmeans = KMeans(n_clusters=n_clusters, random_state=42)

   # Appliquer le modèle
   data['cluster'] = kmeans.fit_predict(features_scaled)
```

```
]: # Afficher les résultats
   cluster_summary = data.groupby('cluster').agg({
       'cost_of_the_order': ['mean', 'count'],
       'food_preparation_time': 'mean',
       'delivery_time': 'mean'
   }).reset_index()

   print(cluster_summary)
```

```
  cluster cost_of_the_order       food_preparation_time delivery_time
                     mean count                   mean          mean
0       0        17.784189   962              31.280665     24.281705
1       1        15.177810   936              23.354701     24.038462
```

```
5]: data.sample(10)
```

5]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time	cluster
1638	1476960	45577	Parm	Italian	21.83	Weekday	4.344234	23	32	1
1354	1478239	400390	Shake Shack	American	12.95	Weekend	4.000000	30	26	0
670	1477302	52832	Don's Bogam BBQ & Wine Bar	Korean	12.23	Weekend	4.344234	32	20	0
1239	1477261	76938	Sushi of Gari	Japanese	31.43	Weekend	4.344234	21	27	1
1080	1477272	143984	The Smile	American	12.27	Weekday	4.344234	32	24	0
934	1477808	123977	Vezzo Thin Crust Pizza	Italian	13.05	Weekend	5.000000	31	27	0
1133	1477092	144851	Benihana	Japanese	16.11	Weekend	4.344234	25	29	1
1819	1478285	113817	Blue Ribbon Fried Chicken	American	12.18	Weekend	4.000000	32	22	0
91	1477569	65009	Chote Nawab	Indian	16.15	Weekend	5.000000	24	30	1
198	1477290	361509	Shake Shack	American	19.35	Weekday	4.344234	35	33	0

6) Déploiement

Le déploiement en sciences des données fait référence au processus de mise en production des modèles analytiques ou prédictifs développés pour qu'ils soient accessibles et utilisés dans des applications réelles. Cela implique non seulement la conversion du code et des modèles en un format opérationnel, mais aussi leur intégration dans des systèmes informatiques existants, la mise en place d'infrastructures pour gérer les données en temps réel, et l'assurance que le modèle fonctionne

efficacement dans un environnement de production. Le déploiement inclut également la surveillance et la maintenance des performances du modèle, permettant ainsi une mise à jour continue en fonction des nouvelles données ou des changements dans le comportement des utilisateurs.

Dans notre cas, nous avons mis en place une fonction qui utilise le modèle KNN pour les nouvelles informations qui viendront.

```
]: from sklearn.preprocessing import StandardScaler
   from sklearn.neighbors import KNeighborsClassifier
   # Normaliser les données
   features = data[['cost_of_the_order', 'food_preparation_time', 'delivery_time']]
   scaler = StandardScaler()
   features_scaled = scaler.fit_transform(features)
   # Ajuster le modèle KNN
   knn = KNeighborsClassifier(n_neighbors=3)
   # Choisissez le nombre de voisins
   knn.fit(features_scaled, data['cluster'])

   def predict_cluster_knn(cost_of_order, food_prep_time, delivery_time):
       """
       Prédit le cluster pour une commande donnée en utilisant KNN.

       :param cost_of_order: Coût de la commande
       :param food_prep_time: Temps de préparation des aliments
       :param delivery_time: Temps de livraison
       :return: Cluster prédit
       """
       # Normaliser les nouvelles valeurs
       new_data = [[cost_of_order, food_prep_time, delivery_time]]
       new_data_scaled = scaler.transform(new_data)  # Normaliser les nouvelles données

       # Prédire le cluster
       cluster = knn.predict(new_data_scaled)  # Prédire le cluster
       return cluster[0]  # Retourner le cluster
```

```
# Exemple d'utilisation
cost = float(input("Entrez le coût de la commande : "))
prep_time = float(input("Entrez le temps de préparation (en minutes) : "))
delivery_time = float(input("Entrez le temps de livraison (en minutes) : "))

predicted_cluster = predict_cluster_knn(cost, prep_time, delivery_time)
print(f"La commande appartient au cluster : {predicted_cluster}")
```

```
Entrez le coût de la commande : 10
Entrez le temps de préparation (en minutes) : 15
Entrez le temps de livraison (en minutes) : 20

C:\Users\hp\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does
er was fitted with feature names
  warnings.warn(

La commande appartient au cluster : 1
```

Conclusion

L'analyse des données de FoodHub a permis de mettre en lumière plusieurs aspects critiques du service de livraison de repas. Tout d'abord, il ressort que certains types de cuisines et certains restaurants sont nettement plus populaires que d'autres. Cela représente une opportunité pour FoodHub d'optimiser son offre en s'associant davantage avec ces restaurants performants et en les mettant en avant sur la plateforme. De plus, l'analyse des temps de préparation et de livraison révèle des possibilités d'amélioration dans la chaîne logistique, en identifiant les restaurants et livreurs les plus rapides.

L'étude des notes attribuées par les clients permet également de mieux comprendre les attentes des consommateurs. L'amélioration de la qualité du service, notamment en réduisant les temps d'attente et en assurant une meilleure communication tout au long du processus de livraison, peut significativement accroître la satisfaction et la fidélité des clients.

Enfin, cette analyse démontre l'importance des données dans la prise de décision stratégique. En exploitant de manière continue les données collectées, FoodHub pourra non seulement améliorer ses opérations, mais aussi anticiper les tendances futures du marché et s’adapter en conséquence.