

QUELLE BANQUE A PLUS DE CLIENTS SATISFAITS À RABAT

Réalisé Par:

MOUMAD Hamza

AFRACHE Yassine

Encadrée Par :

Pr. Imade Benelallam

08/06/2023

Année Académique : 2022-2023

Table des matières

1 Introduction Generale	5
2 Application d'ETL	7
2.1 Introduction	7
2.2 Collecte et nettoyage de données	7
2.2.1 Extraction de données	8
2.2.2 Collect de données	10
2.3 Traitement de données	11
2.3.1 Extraction du nom de la banque a partir de l'agence	11
2.3.2 modèle de machine learning RoBERTa	12
2.4 Chargement de données	14
2.4.1 Destination de données	14
2.4.2 Code de chargement de données	16
2.5 Conclusion	17
3 Automatisation D'ETL	19
3.1 Introduction	19
3.2 Presenatation du Air-flow	19
3.3 Code Air-Flow	20
3.4 Process d'Automatisation	21
3.5 Conclusion	24
4 visualisation et Interpretations	25
4.1 Introducation	25
4.2 visualisation de données	25

4.3 Interpretation	27
---------------------------	-------	----

5 Conclusion		28
---------------------	--	----

Table des figures

2.1 Selenium	8
2.2 PostgreSQL	14
2.3 DBeaver	14
2.4 description de la table	15
2.5 La table Banques [1].	16
3.1 Air-flow	19
3.2 Scrappage de données [1].	22
3.3 Premiere tâche [1].	22
3.4 Deuxieme tâche [1].	23
3.5 Troisieme tâche [1].	23
3.6 quatrieme tâche [1].	24
4.1 Visualisation de données [1].	27

Listings

2.1	Code D'extraction	8
2.2	Code De Netwayage	12
2.3	Code De Chargement	16
3.1	Code De Air-flow	20
4.1	connection et extraction des enregistrement	25
4.2	Code de visualisation	26

1

Introduction Generale

Ce modest projet est un projet de Data Mining passionnant qui vise à créer un processus ETL (Extraction, Transformation, Chargement) pour les banques de la ville de Rabat. L'objectif principal de ce projet est d'analyser le comportement des clients envers leurs banques respectives, en automatisant le traitement des données afin de permettre une analyse régulière et en temps réel.

Le Data Mining, également connu sous le nom de fouille de données, est un domaine de l'informatique qui se concentre sur l'extraction de connaissances à partir de grandes quantités de données. Dans ce contexte, notre projet se concentre sur les données des banques de la ville de Rabat, qui sont extrêmement précieuses pour comprendre les préférences, les habitudes et les comportements des clients vis-à-vis de leurs services bancaires.

L'un des défis majeurs dans ce projet consiste à collecter, nettoyer et transformer les données provenant de différentes sources bancaires. C'est là que l'ETL entre en jeu. Le processus d'ETL permet d'extraire les données brutes des banques, de les transformer en un format cohérent et d'en charger les résultats dans une base de données centralisée. Cela facilite ensuite l'analyse et l'extraction de connaissances à partir des données collectées.

Grâce à l'automatisation du processus ETL, il devient possible d'analyser le comportement actuel des clients de manière régulière et en temps réel. Cela permet aux banques de Rabat de prendre des décisions éclairées en ce qui concerne l'amélioration de leurs services, la personnalisation de l'expérience client et l'identification de nouvelles opportunités commerciales.

Afin de garantir l'automatisation efficace du processus ETL, nous avons choisi d'utiliser Airflow, un outil open-source largement utilisé pour la gestion des workflows. Airflow offre une plateforme flexible et puissante permettant de planifier, d'orchestrer et de surveiller les tâches liées à

l'ETL. Grâce à Airflow, nous pouvons définir des flux de travail complexes, configurer des dépendances entre les différentes étapes et automatiser l'exécution des tâches.

Ce rapport détaillera les différentes étapes du projet, y compris la collecte des données, la conception et la mise en place de l'ETL, ainsi que les analyses réalisées sur les données extraites. Nous explorerons également les résultats obtenus, les tendances identifiées et les recommandations pour les banques de Rabat.

Application d'ETL

2.1 Introduction

Dans ce chapitre, nous nous concentrons sur le¹. La collecte des données consiste à extraire les données brutes des sources bancaires de Rabat, tandis que le nettoyage vise à éliminer les erreurs et à garantir la qualité des données.

Ensuite, nous passons au traitement des données, en appliquant diverses opérations et transformations pour les préparer à l'analyse. Finalement, nous téléchargeons les données traitées dans une base de données centralisée, en assurant la sécurité et la disponibilité des données pour les analyses ultérieures. Comprendre ces étapes clés est essentiel pour garantir des résultats précis et significatifs.

2.2 Collecte et nettoyage de données

Pour la collecte des données dans notre projet, nous avons opté pour le scraping de Google Maps à l'aide de la bibliothèque Selenium. Cette approche nous permet d'extraire les informations pertinentes concernant les banques de la ville de Rabat directement à partir des pages web de Google Maps.

Nous avons utilisé le scraping de Google Maps avec Selenium pour collecter des données précieuses sur les banques de la ville de Rabat, y compris les informations sur les établissements bancaires et les commentaires des clients.

1. processus de collecte, de nettoyage, de traitement et de téléchargement des données

2.2.1 Extraction de données

En utilisant Selenium, nous avons développé des scripts pour automatiser le processus de navigation sur les pages de Google Maps. Ces scripts ont été conçus pour rechercher les banques de Rabat, extraire les informations telles que les noms des banques, les adresses..., les évaluations des utilisateurs et les commentaires des clients.

- **Selenium**

Selenium est une bibliothèque populaire et puissante pour l'automatisation des tests et le scraping web. Il fournit une interface conviviale et des outils robustes pour contrôler les navigateurs web et interagir avec les pages web de manière programmable.



FIGURE 2.1: Selenium

Selenium prend en charge plusieurs langages de programmation, tels que Python, Java, C#, et fournit des pilotes spécifiques pour différents navigateurs web tels que Chrome, Firefox, Safari, etc. Cela permet aux développeurs de choisir leur langage de programmation préféré et d'utiliser Selenium avec leur navigateur favori.

```

1 from time import time,sleep
2 from selenium.webdriver.common.by import By
3 from selenium import webdriver
4 from selenium.webdriver.support.ui import WebDriverWait
5 from selenium.webdriver.support import expected_conditions as EC
6 import json
7
8
9 driver = webdriver.Chrome('~/chromedriver')
10 wait = WebDriverWait(driver, 10)
11 root = 'https://www.google.com/maps/search/Banque/'
12 driver.get(root)
13 boot_start = time()
14 boot_finish = time()
15 while boot_finish - boot_start < 60:
16     scroll = driver.find_element(By.XPATH,'//*[@id="QAOSzd"]/div/div/div[1]/div[2]/div/div[1]/div/div/div[2]/div[1]')
17     driver.execute_script("arguments[0].scrollTop = arguments[0].scrollHeight", scroll)
  
```

```
17     sleep(3)
18
19     boot_finish = time()
20
21 elements = [e.find_element(By.TAG_NAME, 'a').get_attribute('href') for e in
22             driver.find_elements(By.CSS_SELECTOR, 'div[class^=Nv2PK]')]
23
24 stat = []
25
26 for element in elements:
27
28     Bank = {'nom': '', 'adresse': '', 'rate': '', 'nombre_reaction': '', 'avis': []}
29
30     driver.get(element)
31
32     # Wait for the element to be present
33     nom_agence_element = wait.until(EC.presence_of_element_located((By.
34 TAG_NAME, 'h1'))))
35
36     Bank['nom'] = nom_agence_element.text
37
38     try:
39
40         adresse_element = wait.until(EC.presence_of_element_located((By.
41 XPATH, '//*[@id="QA0SzD"]//div/div[1]/div[2]/div/div[1]/div/div[7]/
42 div[3]/button/div/div[3]/div[1]')))
43
44         Bank['adresse'] = adresse_element.text
45
46     except :
47
48         pass
49
50     try:
51
52         rate_element = driver.find_element(By.XPATH, '//*[@id="QA0SzD"]//div/
53 div/div[1]/div[2]/div/div[1]/div/div/div[2]/div/div[1]/div[2]/div/div[1]/
54 div[2]/span[1]/span[1]')
55
56         Bank['rate'] = rate_element.text
57
58     except:
59
60         pass
61
62     try:
63
64         nombre_avis_element = driver.find_element(By.XPATH, '//*[@id="QA0SzD"]
65 //div/div/div[1]/div[3]/div/div[1]/div/div/div[2]/div/div[1]/div[2]/div/div[1]/
66 div[2]/span[2]/span/span')
67
68         Bank['nombre_reaction'] = re.sub('(\|)', ',', nombre_avis_element.text
69 )
70
71     except:
72
73         pass
74
75     print(Bank['nom'])
76
77     print(Bank['rate'])
78
79     print(Bank['adresse'])
80
81     print(Bank['nombre_reaction'])
82
83     try:
84
85         wait.until(EC.presence_of_element_located((By.XPATH, '//*[@id="QA0SzD"]')))
```

```

    "]/div/div/div[1]/div[2]/div/div[1]/div/div/div[3]/div/div/button[2']"))

48     print('existing reviews')

49     driver.find_element(By.XPATH, '//*[@id="QA0SzD"]/div/div/div[1]/div
50 [2]/div/div[1]/div/div/div[3]/div/div/button[2']).click()

51     print('get to reviews')      wait.until(EC.
52 presence_of_element_located((By.CLASS_NAME, "wiI7pd")))

53     print('some comments there')

54     start = time()
55     finish = time()
56     while finish - start < 10:
57         scroll = driver.find_element(By.XPATH, '//*[@id="QA0SzD"]/div/div
58 /div[1]/div[2]/div/div[1]/div/div/div[3]')
59         print('scrolling')           driver.execute_script("arguments
60 [0].scrollTop = arguments[0].scrollHeight", scroll)
61         print('run scrolling')
62         sleep(3)
63         finish = time()
64     Bank['avis'] = [avis.text for avis in driver.find_elements(By.
65 CLASS_NAME, 'wiI7pd')]
66     print(Bank['avis'])

67 except:
68     print('no comments')
69     stat.append(Bank)
70     driver.back()
71
72 driver.quit()
73 with open('/home/yassine/python_works/banque_info.json', 'w') as file:
74     json.dump(stat, file)

```

Listing 2.1: Code D'extraction

2.2.2 Collect de données

Après avoir extrait les données à partir de Google Maps, nous avons choisi de stocker ces données dans un fichier JSON. Le format JSON (JavaScript Object Notation) est largement utilisé pour représenter des données structurées sous forme de paires clé-valeur, ce qui en fait une option pratique pour stocker des ensembles de données.

En utilisant le format JSON, nous avons pu organiser les informations collectées à partir de Google Maps de manière hiérarchique et facilement compréhensible. Chaque établissement ban-

caire extrait a été représenté sous la forme d'un objet JSON, avec des clés représentant les différents attributs tels que le nom de la banque, l'adresse, le numéro de téléphone, les heures d'ouverture, les évaluations des utilisateurs, les commentaires, etc.

```

1  [
2    {"nom": "Agence Attijari Wafa Banque", "adresse": "", "rate": "2,9", "nombre_reaction": "", "avis": ["Un service nul avec une \u00e9quipe non professionnelle", "agence proche \u00e0 l'APESA", "Personal banking, great service, friendly people."]},
3
4    {"nom": "Cr\u00e9dit Agricole", "adresse": "X46F+R4V, Rabat", "rate": "3,0", "nombre_reaction": "", "avis": ["Je ne conseille pas cette agence, le personnel est tr\u00e8s antipathique et pas du tout serviable.", "Bonne \u00e9quipe"]},
5
6    {"nom": "Banque Populaire", "adresse": "", "rate": "", "nombre_reaction": "", "avis": []},
7
8    {"nom": "BMCI Banque", "adresse": "X4MC+PH2, Rabat", "rate": "", "nombre_reaction": "", "avis": []}

```

2.3 Traitement de données

2.3.1 Extraction du nom de la banque à partir de l'agence

Dans la phase de nettoyage des données, nous avons utilisé un script Python pour effectuer le nettoyage et la transformation des données extraites. L'objectif principal était d'extraire le nom de la banque à partir des noms d'agences récupérés.

En utilisant des techniques de traitement de texte et des expressions régulières, nous avons développé un script qui a parcouru les données extraites et extrait le nom de la banque à partir des noms d'agences. Étant donné que les noms d'agences peuvent contenir des informations supplémentaires, telles que le numéro d'agence ou la localisation, notre script a utilisé des règles spécifiques pour identifier et extraire le nom de la banque principale.

En effectuant ce nettoyage des données, nous avons simplifié et préparé les données pour les étapes ultérieures de l'analyse. En se concentrant sur le nom de la banque, nous avons créé un

ensemble de données plus facile à manipuler et à analyser, permettant de mieux comprendre le comportement des clients des banques de Rabat.

2.3.2 modèle de machine learning RoBERTa

Dans la deuxième phase de nettoyage des données, nous avons utilisé un modèle de machine learning appelé "RoBERTa Multilanguage" pour extraire les opinions des clients à partir des commentaires recueillis.

En utilisant ce modèle, nous avons créé un pipeline de traitement des commentaires des clients. Le modèle RoBERTa a été utilisé pour effectuer une classification des sentiments afin de déterminer si un commentaire est positif, négatif ou neutre. Le modèle a été entraîné sur un ensemble de données annotées manuellement pour apprendre à reconnaître les différents sentiments exprimés dans les commentaires.

Le pipeline de traitement a permis d'extraire les opinions des clients à partir des commentaires, en identifiant les sentiments exprimés et en les associant aux établissements bancaires correspondants. Cela nous a fourni des informations précieuses sur la satisfaction des clients, les problèmes rencontrés et les aspects positifs des services bancaires dans la ville de Rabat.

En utilisant un modèle de machine learning comme RoBERTa, nous avons pu automatiser le processus d'analyse des commentaires des clients, ce qui a permis d'obtenir rapidement des informations pertinentes à grande échelle. Cela nous a également permis de comparer les opinions des clients entre différentes banques et de détecter les tendances et les problématiques communes.

- **Code du Netwayage**

```

1 import json
2 import pandas as pd
3 import re
4 from transformers import pipeline
5
6 model_path = "cardiffnlp/twitter-xlm-roberta-base-sentiment"
7 sentiment_task = pipeline("sentiment-analysis", model=model_path, tokenizer=
    model_path)
8
9 df = pd.read_json('/home/yassine/python_works/banque_info.json')

```

```

10 df[‘nom’] = df[‘nom’].str.lower()
11
12 agences = [‘barid’, ‘populaire’, ‘cih’, ‘africa’, ‘assafa’, ‘omnia’, ‘agricole’, ‘
13 ‘bmce’, ‘g n rale’]
14 pattern = ‘(‘+|’.join(agences)+’)’
15 df[‘nom_banque’] = df[‘nom’].apply(lambda x: re.findall(pattern,x)[0] if re.
16 findall(pattern,x) else ‘unknown’)
17
18 def calcul_score(list_avis):
19     weight = 0
20     feedback = []
21     for avis in list_avis:
22         result = sentiment_task(avis)[0]
23         if result[‘label’] == ‘negative’:
24             weight -= result[‘score’]
25         else:
26             weight += result[‘score’]
27     return 0 if len(list_avis) == 0 else weight/len(list_avis)
28
29 df[‘score’] = df[‘avis’].apply(calcul_score)
30
31 df.to_json(‘/home/yassine/python_works/processData.json’, orient = ‘split’,
32 compression = ‘infer’, index = ‘true’)

```

Listing 2.2: Code De Netwayage

- Réultats après le Netwayage**

```

{"columns": ["nom", "adresse", "rate", "nombre_reaction", "avis", "nom_banque", "score"], "index
": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, ..., 119], "data": [
["agence attijari wafa banque", "", "2,9", "", ["Un service nul avec
une \u00e9quipe non professionnelle", "agence proche \u00e0 l'APESA",
Personal banking, great service, friendly people."], "unknown
", 0.2469862501], ["cr\u00e9dit agricole", "X46F+R4V, Rabat", "3,0", "", ["Je
ne conseille pas cette agence, le personnel est tr\u00e8s antipathique et
pas du tout serviable.", "Bonne \u00e9quipe"], "agricole
", -0.0281277299], ["banque populaire", "", "", "", [], "populaire", 0.0], ["bmci
banque", "X4MC+PH2, Rabat", "", "", [], "unknown", 0.0]
]
```

2.4 Chargement de données

2.4.1 Destination de données

- PostgreSQL

Dans l'étape de chargement des données, nous avons choisi d'utiliser une base de données PostgreSQL comme destination finale. PostgreSQL est un système de gestion de base de données relationnelle puissant, fiable et largement utilisé, offrant des fonctionnalités avancées pour le stockage et la manipulation des données.

En transférant les données extraites et nettoyées vers PostgreSQL, nous avons bénéficié des avantages d'une base de données relationnelle bien structurée. Nous avons créé une structure de base de données adaptée à notre projet, comprenant des tables pour stocker les informations sur les banques, les commentaires des clients, les évaluations et d'autres attributs pertinents.



FIGURE 2.2: PostgreSQL

- DBeaver-ce



FIGURE 2.3: DBeaver

Pour gérer et interagir avec la base de données PostgreSQL utilisée dans notre projet, nous avons utilisé l'outil DBeaver. DBeaver est un client de base de données universel et open source qui prend en charge plusieurs systèmes de gestion de base de données, y compris PostgreSQL.

DBeaver offre une interface conviviale et intuitive pour se connecter à la base de données PostgreSQL, exécuter des requêtes SQL, naviguer dans les schémas de la base de données, gérer les tables et les vues, ainsi que visualiser et modifier les données. Il permet également de créer des requêtes complexes, d'importer et d'exporter des données, et de générer des rapports.

- Table Destinataire

En utilisant des requêtes SQL, nous avons inséré les données nettoyées dans les tables appropriées de la base de données PostgreSQL. Nous avons pris soin de maintenir la cohérence et l'intégrité des données en appliquant des contraintes et des relations entre les différentes tables, garantissant ainsi la qualité des données et la facilité d'accès pour les analyses ultérieures.

Dans notre projet, la table de destination que nous avons utilisée s'appelle "banques". Cette table est spécifiquement conçue pour stocker les données relatives aux banques de la ville de Rabat, et elle comprend six colonnes.



FIGURE 2.4: description de la table

1. **"bank_id" (id auto-increment)** : Cette colonne sert d'identifiant unique pour chaque enregistrement dans la table. Elle utilise une fonction d'auto-incrémentation pour attribuer automatiquement une valeur unique à chaque nouvelle entrée.
2. **"agence" (varchar_100)** : Cette colonne stocke le nom de l'agence de la banque. Elle est définie comme un champ de texte de type VARCHAR avec une longueur maximale de 100 caractères.
3. **"address" (varchar_100)** : Cette colonne enregistre l'adresse de l'agence bancaire. Elle est également définie comme un champ de texte de type VARCHAR avec une longueur maximale de 100 caractères.
4. **"nombre_reaction" (int_4)** : Cette colonne stocke le nombre de réactions ou d'avis exprimés par les clients pour chaque agence bancaire. Elle est définie comme un entier de 4 octets (int) pour enregistrer les valeurs numériques.
5. **"banque" (varchar_100)** : Cette colonne contient le nom de la banque à laquelle appartient l'agence. Elle est définie comme un champ de texte de type VARCHAR avec une longueur maximale de 100 caractères.
6. **"score" (float_8)** : Cette colonne enregistre le score attribué à chaque agence bancaire, qui

est obtenu à partir de l'analyse des commentaires des clients à l'aide du modèle de machine learning RoBERTa.

Dans notre projet, la table "banques" contient 120 enregistrements, ce qui signifie que nous avons réalisé une étude sur 120 banques de la ville de Rabat. Chaque enregistrement dans cette table représente une banque spécifique et contient des informations pertinentes pour notre analyse.

Le figure suivant illustre les enregistrement de la table.

	bank_id	agence	adresse	nombre_reaction	banque	score
1	750	agence attijari wafa banque		[NULL]	unknown	0.2469862501
2	751	crédit agricole	X46F+R4V, Rabat	[NULL]	agricole	-0.0281277299
3	752	banque populaire		[NULL]	populaire	0
4	753	bmci banque	X4MC+PH2, Rabat	[NULL]	unknown	0
5	754	banque populaire	X46M+69W, Rabat	[NULL]	populaire	-0.8062218252
6	755	société générale	N° 15,Centre Commercial MAHAJ F	[NULL]	générale	-0.6133375883
7	756	banque populaire al massira	967, Al Manzah Lot Olm, Yacoub E	[NULL]	populaire	0
8	757	crédit du maroc	Av. Annakhil, Rabat	[NULL]	unknown	-0.4145423273
9	758	banque assafa ennakhil hay riad	X46G+2PH, Av. Al Haour, Rabat	[NULL]	assafa	-0.8285737634
10	759	banque islamique de développement	1 Av. Annakhil, Rabat	[NULL]	unknown	0
11	760	point de vente (autoroutes du ma	2 Rue Joullanar, Rabat 10100	[NULL]	unknown	-0.19946751
12	761	société générale	X48X+385, Rabat	[NULL]	générale	-0.7322269678
13	762	bank al yousr	X44J+PW9, rue Erroumane bd An	[NULL]	unknown	0.5438312292
14	763	banque populaire	X43G+CHR, Rabat	[NULL]	populaire	-0.8998334408
15	764	la banque mondiale	X5G4+4CX, Rue Larbi Ben Abdellâ	[NULL]	unknown	0.8845962882
16	765	banque populaire	البنك الشعبي agence al manal. 10000 Av. Al Ma	[NULL]	populaire	0
17	766	banque populaire	X4G5+C89, Av. Al Menzeh, Rabat	[NULL]	populaire	0.3435395956
18	767	banque populaire	X4J7+F4M, Av. Al Mostakbal, Raba	[NULL]	populaire	0
19	768	cih bank	av. Annakhil, imm. High Tech n°3	[NULL]	cih	-0.719598676
20	769	attijariwafa bank gab	X46P+H6G, Av. Al Arz, Rabat	[NULL]	unknown	0
21	770	bmci banque	X4MC+PH2, Rabat	[NULL]	unknown	0
22	771	banque populaire	X4WC+78G, Rabat	[NULL]	populaire	0.8617748618
23	772	crédit du maroc mahaj riad	X47P+534, Rue Joullanar, Rabat 1	[NULL]	unknown	-0.4317236543
24	773	banque populaire	X5H7+C56, Rabat	[NULL]	populaire	0.7368180454
25	774	banque islamique de développement	1 Av. Annakhil, Rabat	[NULL]	unknown	0
26	775	centre d'affaire credit agricole	X5R3+RV2, Rabat	[NULL]	unknown	0

FIGURE 2.5: La table Banques [1].

2.4.2 Code de chargement de données

```

1 import psycopg2
2 import json
3 import pandas as pd
4
5 df = pd.read_json('/home/yassine/python_works/processData.json', orient =
6   'split', compression = 'infer')
7 con = psycopg2.connect(

```

```

8     database='postgres',
9     user='afrache',
10    password='yassine',
11    host='localhost',
12    port='5432'
13 )
14 cursor_obj = con.cursor()
15
16 # Iterate over rows using iterrows()
17 for index, row in df.iterrows():
18     nom = row['nom']
19     adresse = row['adresse']
20     #rate = float(row['rate'])
21     nombre_reaction = int(row['nombre_reaction']) if row['nombre_reaction']
22     else None # Cast rate to integer, handle empty values
23     nom_banque = row['nom_banque']
24     score = float(row['score'])
25
26     # Use parameterized query to insert values
27     cursor_obj.execute("INSERT INTO banques (agence,adresse,nombre_reaction,
28                         banque,score) VALUES (%s, %s, %s, %s, %s)",
29                         (nom, adresse,nombre_reaction,nom_banque,score))
30     print('inserted')
31 con.commit()
32 cursor_obj.close()
33 con.close()

```

Listing 2.3: Code De Chargement

2.5 Conclusion

Dans ce chapitre, nous avons décrit le processus de collecte, de nettoyage, de traitement et de téléchargement des données dans le cadre de notre projet d'analyse du comportement des clients des banques de la ville de Rabat.

À l'aide de Selenium, nous avons extrait des données pertinentes à partir de Google Maps, puis les avons nettoyées en se concentrant sur l'extraction des noms de banques à partir des noms d'agences. En utilisant le modèle de machine learning RoBERTa Multilanguage, nous avons également analysé les commentaires des clients pour obtenir des scores d'évaluation.

Enfin, nous avons téléchargé les données traitées dans une base de données PostgreSQL à l'aide de DBeaver. Ce processus nous a permis d'obtenir des données fiables et structurées pour les analyses futures sur le comportement des clients vis-à-vis des banques de Rabat.

Automatisation D'ETL

3.1 Introduction

L'automatisation des données joue un rôle essentiel dans de nombreux projets de data mining et d'analyse de données. C'est ici qu'Airflow entre en jeu.

Airflow est une plateforme open-source d'automatisation des workflows qui offre des fonctionnalités avancées pour la planification, l'exécution et la surveillance des tâches liées aux données.

3.2 Présentation du Air-flow



FIGURE 3.1: Air-flow

Airflow est une plateforme open-source d'automatisation des workflows développée par Apache. Il permet de créer, planifier et gérer des flux de travail complexes et répétables pour traiter et analyser des données.

En utilisant Airflow, il devient possible de créer des pipelines de données flexibles et évolutifs, où chaque étape du processus est exécutée de manière automatisée et planifiée. Cela permet d'améliorer l'efficacité et la fiabilité du processus global, réduisant ainsi la nécessité d'interventions manuelles et minimisant les erreurs potentielles.

De plus, Airflow offre une interface conviviale pour la visualisation des workflows, la gestion des paramètres et la surveillance des tâches en cours. Il fournit également des mécanismes de notification et de gestion des erreurs, ce qui facilite la détection et la résolution des problèmes po-

tentiels.

Dans ce rapport, nous explorerons l'automatisation des données en utilisant Airflow comme un outil puissant pour orchestrer et exécuter les différentes étapes de notre projet de data mining. Nous découvrirons comment Airflow peut simplifier le processus global, améliorer l'efficacité et permettre une analyse en temps réel du comportement des clients des banques de Rabat.

3.3 Code Air-Flow

Le code suivant illustre une approche structurée pour organiser les tâches d'extraction, de transformation et de chargement des données.

Chaque tâche est représentée par une fonction spécifique, telle que l'extraction des données, la transformation des données et le chargement des données. Ces fonctions peuvent contenir le code nécessaire pour effectuer les opérations correspondantes sur les données.

Le code suivant utilise la valeur `schedule_interval='@daily'` pour planifier l'exécution du DAG. Cela signifie que le DAG sera exécuté une fois par jour. Lorsque le DAG est démarré pour la première fois, il exécutera les tâches définies à ce moment-là. Ensuite, il sera automatiquement planifié pour s'exécuter tous les jours à la même heure, en fonction de la date de début spécifiée dans le code.

```

1 from datetime import timedelta,datetime
2 from airflow import DAG
3 from airflow.operators.bash import BashOperator
4 from airflow.operators.postgres_operator import PostgresOperator
5 from time import time,sleep
6
7 default_args = {
8     'owner':'yassine',
9     'retries': 5,
10    'retry_delay':timedelta(minutes=2)
11 }
12 with DAG(
13     dag_id = 'Bank_statistics',
14     default_args=default_args,
```

```

15     description='this operator try to scrap data from RABAT banks and make
16     some data processing',
17     start_date=datetime(2023,5,24,2),
18     schedule_interval='@daily',
19     catchup = False
20 ) as dag:
21
22     task1 = BashOperator(
23         task_id = 'get_data',
24         bash_command = "python /home/yassine/python_works/scrapBank.
25 py"
26
27     )
28
29     task2 = BashOperator(
30         task_id = 'process_data',
31         bash_command = "python /home/yassine/python_works/
32 processingData.py"
33     )
34
35     task3 = PostgresOperator(
36         task_id = 'truncate_table',
37         postgres_conn_id = 'yassine',
38         sql = "DELETE FROM banques;"
39     )
40
41     task4 = BashOperator(
42         task_id = 'load_data',
43         bash_command = "python /home/yassine/python_works/load_data.
44 py"
45     )
46
47     task1 >> task2 >> task3 >> task4

```

Listing 3.1: Code De Air-flow

3.4 Process d'Automatisation

Lors du lancement d'Airflow, la première tâche à exécuter est le scrapping de données. Le scrapping de données est un processus automatisé qui consiste à extraire des informations à partir de GooglMap. Cette première tâche permet de collecter les données nécessaires pour alimenter le reste du flux de travail dans Airflow.

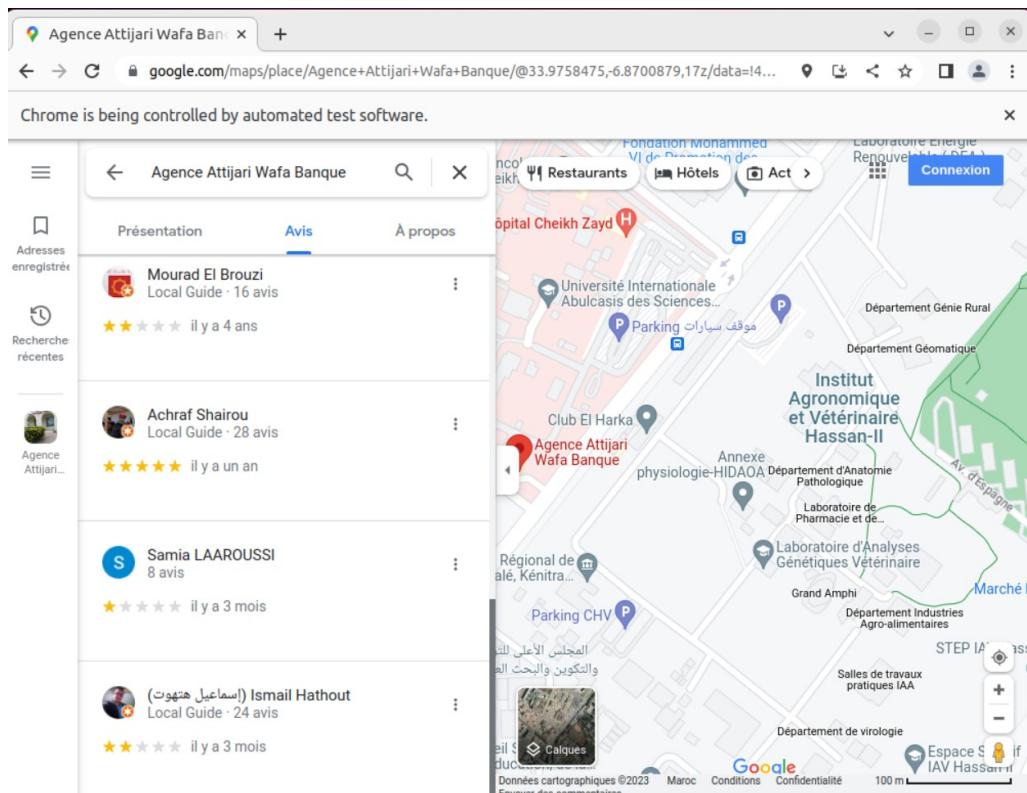


FIGURE 3.2: Scrappage de données [1].

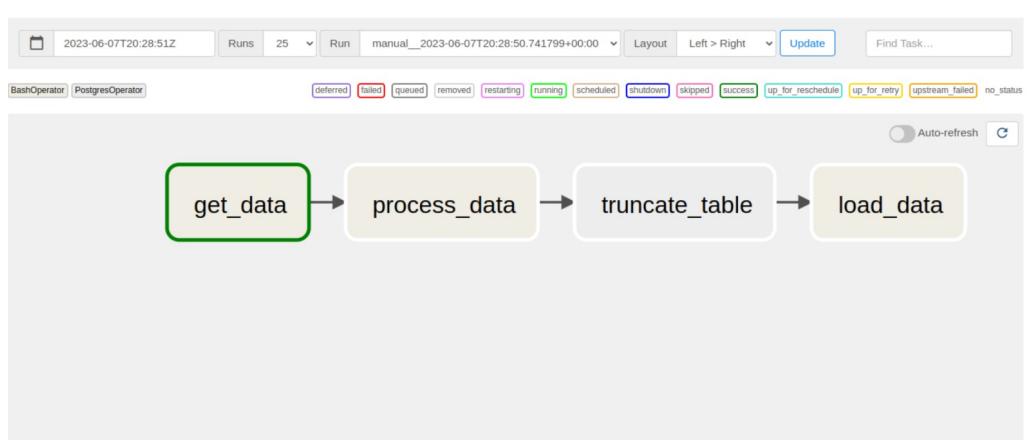


FIGURE 3.3: Première tâche [1].

Une fois que les données ont été scrappées, la tâche suivante peut consister à les transformer ou à les nettoyer pour les préparer à leur utilisation ultérieure. Cela peut inclure le filtrage, la normalisation, la fusion ou toute autre manipulation nécessaire pour garantir la qualité et la cohérence des données.

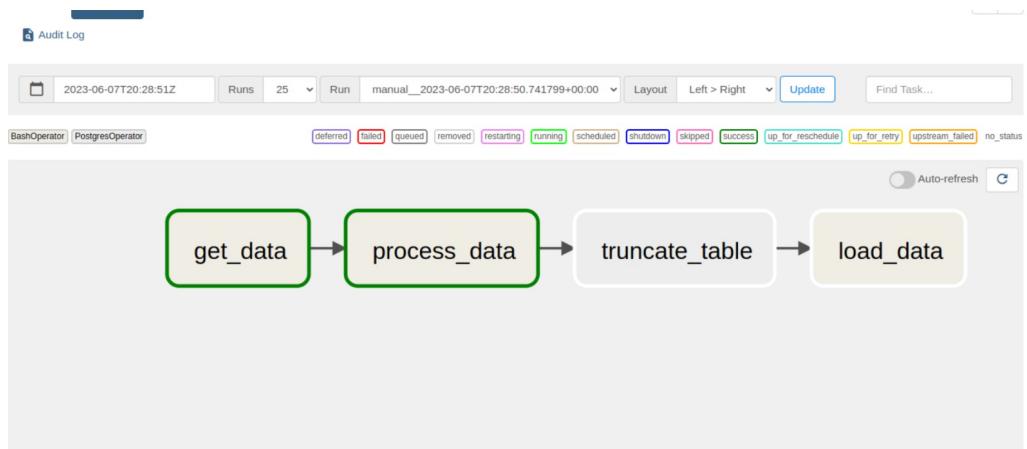


FIGURE 3.4: Deuxième tâche [1].

Une fois que les données ont été transformées, La tâche suivant est une tâche qui permet de vider la table de destination dans le flux de travail de chargement de données.

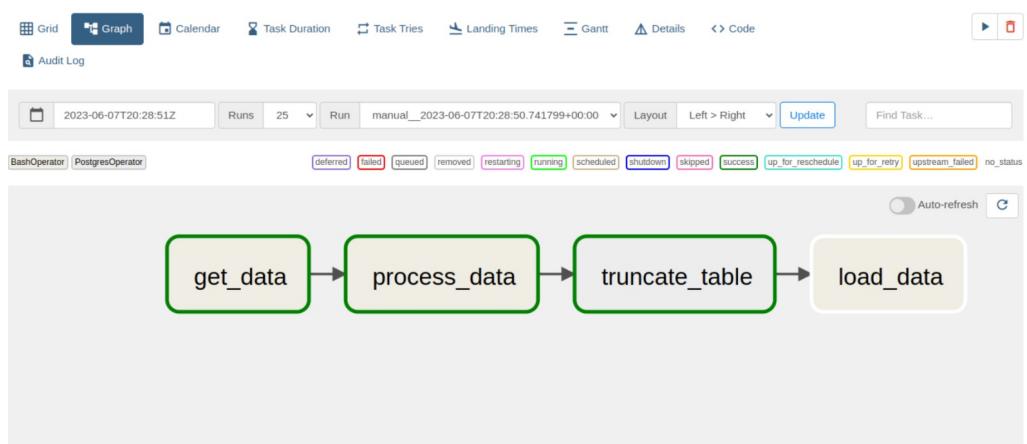


FIGURE 3.5: Troisième tâche [1].

Après avoir vider la table destinataire, les données peuvent être chargées dans la destination finale, comme une base de données, un entrepôt de données ou tout autre système de stockage approprié. Cette étape peut nécessiter une connexion aux systèmes de destination et l'utilisation d'outils ou de bibliothèques spécifiques pour l'insertion ou la mise à jour des données.

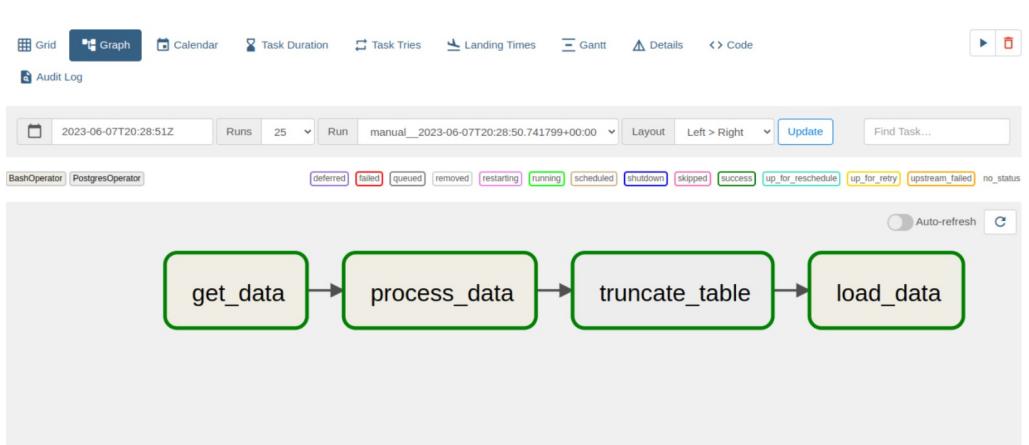


FIGURE 3.6: quatrième tâche [1].

3.5 Conclusion

En somme, l'automatisation de l'ETL a permis d'optimiser notre projet de Data Mining en réduisant les tâches manuelles et en assurant un traitement automatisé, régulier et fiable des données. Cela nous a fourni des informations précieuses sur le comportement des clients des banques de Rabat, ouvrant ainsi la voie à une prise de décision éclairée et à l'amélioration des services bancaires offerts.

visualisation et Interpretaions

4.1 Introduction

Le chapitre à venir porte sur un aspect fondamental de l'analyse de données : la visualisation et l'interprétation des données.

La visualisation des données est un moyen puissant de découvrir des schémas, des tendances et des relations cachées dans les données. Elle nous permet de voir les données d'une manière qui est plus accessible et compréhensible pour notre cerveau, en exploitant sa capacité à traiter les informations visuelles de manière rapide et intuitive.

4.2 visualisation de données

- Code de visualisation

```
1 import psycopg2
2 import pandas as pd
3
4 con = psycopg2.connect(
5     database='postgres',
6     user='afrache',
7     password='yassine',
8     host='localhost',
9     port='5432'
10 )
11 if con :
12     print('connected')
13 else:
```

```

14 print('no connection')

15

16 cursor_obj = con.cursor()
17 cursor_obj.execute("select * from banques")
18 data = cursor_obj.fetchall()

19

20 frame = pd.DataFrame(data,columns=['id','agence','adresse','rate','banque','
21 score'])
21 frame.drop('id',inplace=True, axis=1)
22 frame.drop('rate',inplace=True, axis=1)
23 df = frame[(frame['banque'] != 'unknown')]
24 df = df[(frame['score'] != 0)]

25

26 df['Average Score'] = df.groupby('banque')['score'].transform('mean')
27 banque = df.banque.unique()
28 scoreBanque = df['Average Score'].unique()

```

Listing 4.1: connection et extraction des enregistrement

```

1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 plt.bar(banque, scoreBanque)

4

5 # Set the labels for x and y axis
6 plt.xlabel('banque')
7 plt.ylabel('score')

8

9 # Display the plot
10 plt.show()

```

Listing 4.2: Code de visualisation

- **Le graphe de visualisation**

le code precedent permet d'afficher le graphe suivant :

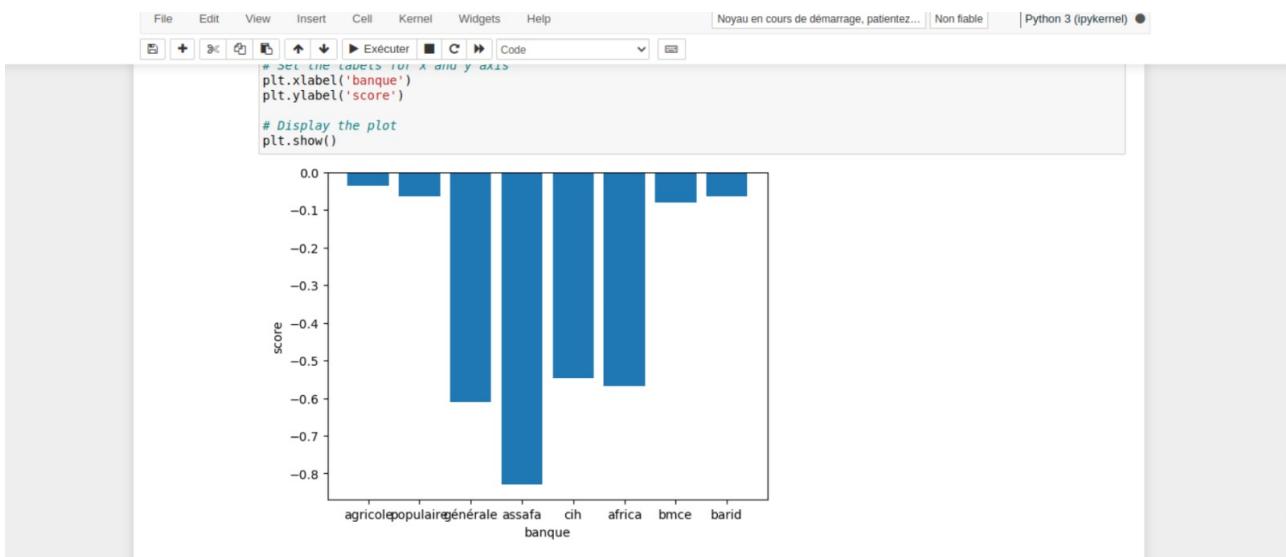


FIGURE 4.1: Visualisation de données [1].

4.3 Interpretation

le plot au-dessus designe les différentes banques meres des agences collecte et leurs scores totaux. On voit dans un premier temps que toutes les banques sans exception malheureusement un score négatif qui designe que la banque a des feedbacks négatifs plus que positifs d'après les clients commentant au service l'intensité diffère d'une banque a l'autre, comme le plot illustre présentla plot illustre presnete la banque assafa a score plus haut que les autres banques, portant al barid bank et la banque agrecole a des score tros faible ce que reflaite soit disant le niveau de satisfaction acceptable de client vis a ces banques.

finalement, ces résultats il ne faut pas les prendre en considération puisque le le nombre des commentaires aussi joue un rôle principal dans le calcul de ces scores, cela se voit dans le cas de banque assa fa qui malgré le nombre des commentaires associe a eu le score a été très élevée pourtant la banque populaire a eu des résultats plus au moins faible que assa fa et cela ne peuvent pas reflete la realite.

5

Conclusion

En conclusion, notre projet de Data Mining avait pour objectif la création d'un ETL pour les banques de la ville de Rabat, afin d'analyser le comportement de leurs clients. Grâce à l'utilisation d'outils tels que Selenium, RoBERTa multilingue et PostgreSQL, nous avons pu automatiser le processus de collecte, de nettoyage, de traitement et de chargement des données.

Nous avons commencé par recueillir les informations nécessaires en extrayant les données des sites de Google Maps grâce à Selenium. Ensuite, nous avons nettoyé les données en extrayant les noms des banques à partir des adresses des agences.

Une fois les données nettoyées, nous avons utilisé le modèle de machine learning RoBERTa multilingue pour extraire les opinions des clients à partir des commentaires. Ces informations ont été stockées dans une base de données PostgreSQL, plus précisément dans une table appelée "banques" avec des colonnes spécifiques telles que bank_id, agence, adresse, nombre_reaction, banque et score.

Le processus de collecte, de nettoyage, de traitement et de chargement des données a été organisé et automatisé grâce à l'utilisation d'Airflow. Ce framework nous a permis de définir les différentes tâches à exécuter, leur ordre d'exécution et les dépendances entre elles.

La visualisation des données nous a permis de communiquer efficacement les résultats de notre analyse à un public non technique, tels que les décideurs et les parties prenantes. Les graphiques et les visualisations ont rendu les informations plus accessibles, compréhensibles et engageantes, facilitant ainsi la prise de décision basée sur les résultats obtenus.

En conclusion, notre projet de Data Mining a permis d'analyser le comportement des clients des banques de Rabat de manière automatisée. Les informations recueillies ont été stockées dans une base de données, ce qui facilite l'analyse ultérieure et la prise de décision basée sur ces données. Ce projet représente une étape importante dans l'utilisation du Data Mining pour comprendre les comportements des clients bancaires et améliorer les services offerts par les banques.