

Ingénierie de Données / Transformation Digitale et Intelligence Artificielle

Semestre 5

Année universitaire 2025/2026

Module : Web Marketing & CRM

TP N° 03

Pr : Sara OUALD CHAIB

Objectifs

- Comprendre les leviers d'acquisition digitale du point de vue technique
- Manipuler des APIs marketing et automatiser des workflows
- Analyser des données marketing avec Python/SQL
- Optimiser la conversion avec du machine learning

Partie 1 - APIs Marketing & Automatisation

1.1 API Email Marketing

Concepts Clés

- **API REST** : endpoints pour l'envoi d'emails, gestion de listes
- **Webhooks** : notifications en temps réel (ouverture, clic, désabonnement)
- **Rate limiting** : gestion des quotas API
- **Event-driven automation** : déclencheurs basés sur le comportement utilisateur

Exercice 1.1 : Script Python d'automatisation email

Contexte : Automatiser l'envoi d'emails de bienvenue via API (Brevo/ SMTP Gmail / Mailgun...etc)

Créez un script Python qui :

1. Se connecte à l'API Brevo avec une clé API
2. Lit une liste de nouveaux inscrits depuis un fichier CSV (*vous pouvez utiliser les emails de vos collègues de classe 2 à 5 @mail*)
3. Envoie un email personnalisé à chaque utilisateur
4. Log les réponses API (*succès/échecs*)

```
# Pseudo-code Python import requests import csv API_KEY = 'votre_clé_api' URL = 'https://api.brevo.com/v3/smtp/email' with open('inscrits.csv') as f:    for row in csv.DictReader(f):        payload = {            'sender': {'email': 'hello@startup.com'},            'to': [{'email': row['email']}],            'subject': f"Bienvenue {row['prenom']} !",            'htmlContent': '<h1>Bienvenue</h1>'        }        response = requests.post(URL, headers={'api-key': API_KEY}, json=payload)        print(f"Email envoyé: {response.status_code}")
```

Données CSV exemple (inscrits.csv) :

email,prenom,date_inscription
jean@email.com,Jean,2025-01-15
marie@email.com,Marie,2025-01-15

1.2 Social Media Data Collection

Concepts Clés

- **Graph API (Meta)** : accès aux données Facebook/Instagram
- **Rate limits** : quotas d'appels API (200 appels/heure typique)
- **Métriques d'engagement** : reach, impressions, engagement_rate
- **ETL** : Extract, Transform, Load pour centraliser les données

Exercice 1.2 : Analyser des données sociales

Contexte : Analyser les performances de publications sur les réseaux sociaux.

Mission : Créez un notebook Jupyter qui :

5. Charge un dataset CSV de posts sociaux (fourni : /social_posts.csv)
6. Calcule les métriques clés (engagement rate, reach moyen)
7. Identifie les meilleurs horaires de publication
8. Visualise les tendances avec matplotlib/seaborn

Questions à répondre :

- Pourquoi utiliser first-party cookies plutôt que third-party ?
- Quelle différence entre pixel synchrone et asynchrone ?

 **Livrable Partie 1 :** Repository Git avec : script Python d'envoi email, notebook Jupyter d'analyse sociale.

Partie 2 - ML Pipeline pour Marketing

2.1 ETL Pipeline & Analytics SQL

Concepts Clés

Architecture Data Warehouse Marketing :

- **Sources** : APIs (Google Analytics, Meta, Brevo), logs serveur
- **Transformation** : nettoyage, normalisation, agrégation
- **Storage** : PostgreSQL, BigQuery, Snowflake
- **Viz** : Metabase, Superset, Looker

Schéma de base de données (PostgreSQL) :

```
-- Schema de la BDD

CREATE TABLE events (
    event_id SERIAL PRIMARY KEY,
    user_id VARCHAR(50),
    event_type VARCHAR(50), -- 'page_view', 'add_to_cart', 'purchase'
    timestamp TIMESTAMP DEFAULT NOW(),
    channel VARCHAR(50), -- 'organic', 'paid', 'email', 'social'
    revenue DECIMAL(10,2) DEFAULT 0,
    properties JSONB -- données additionnelles
);

CREATE TABLE sessions (
    session_id VARCHAR(50) PRIMARY KEY,
    user_id VARCHAR(50),
    start_time TIMESTAMP,
    end_time TIMESTAMP,
    pages_viewed INT DEFAULT 0,
    converted BOOLEAN DEFAULT FALSE,
    channel VARCHAR(50)
);

-- Index pour performance
CREATE INDEX idx_events_user ON events(user_id);
CREATE INDEX idx_events_type ON events(event_type);
CREATE INDEX idx_sessions_user ON sessions(user_id);
```

Exercice 2.1.1 : Requêtes SQL Analytics

Écrivez les requêtes SQL pour calculer :

9. Taux de conversion par canal

```
SELECT channel, COUNT(DISTINCT CASE WHEN converted THEN user_id END) * 100.0 / COUNT(DISTINCT user_id) AS conversion_rate FROM sessions GROUP BY channel;
```

10. Revenu moyen par utilisateur (ARPU)

11. Top 5 des heures avec le meilleur taux de conversion

12. Cohort analysis : rétention des utilisateurs par mois d'inscription

Exercice 2.1.2 : Pipeline ETL avec Python

Créez un script qui :

- Extrait les données de Matomo API
- Transforme les données (nettoyage, agrégation)
- Charge dans PostgreSQL
- Schedule avec cron (toutes les heures)

```
import requests import psycopg2 from datetime import datetime # Extract response =  
requests.get( 'https://matomo.com/api', params={'module': 'API', 'method':  
'VisitsSummary.get'}) data = response.json() # Transform metrics = { 'date':  
datetime.now(), 'visits': data['nb_visits'], 'conversions':  
data['nb_conversions']} # Load conn = psycopg2.connect('dbname=analytics') cur =  
conn.cursor() cur.execute( 'INSERT INTO daily_metrics VALUES (%s, %s, %s)',  
(metrics['date'], metrics['visits'], metrics['conversions'])) conn.commit()
```

2.2 ML Optimization

Concept Clé

- **Predictive models** : ML pour prédire la conversion (Random Forest, XGBoost)

Exercice 2.2.1 : Modèle prédictif de conversion

Créez un modèle ML pour prédire si un utilisateur va convertir :

- Features : temps sur site, pages vues, source de trafic, appareil
- Target : converted (0/1)
- Modèle : Logistic Regression ou Random Forest
- Métriques : Accuracy, Precision, Recall, AUC-ROC

```
from sklearn.ensemble import RandomForestClassifier from sklearn.model_selection  
import train_test_split from sklearn.metrics import classification_report,  
roc_auc_score import pandas as pd # Charger les données df =  
pd.read_csv('user_behavior.csv') # Features et target X = df[['time_on_site',  
'pages_viewed', 'source', 'device']] y = df['converted'] # Encoding des variables  
catégorielles X = pd.get_dummies(X, columns=['source', 'device']) # Train/test  
split X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,  
random_state=42) # Entraînement model = RandomForestClassifier(n_estimators=100)  
model.fit(X_train, y_train) # Prédiction et évaluation y_pred =  
model.predict(X_test) print(classification_report(y_test, y_pred)) print(f'AUC-ROC:  
{roc_auc_score(y_test, y_pred):.3f}') # Feature importance importances =  
model.feature_importances_ for feature, importance in zip(X.columns, importances):  
print(f'{feature}: {importance:.3f}')
```

Livrable Partie 2 : Fichier SQL avec requêtes + Script Python ETL + Notebook ML.

Bon travail !