TP 1 : SYSTÈME DE RECOMMANDATION INTELLIGENT

Module : Concepts et Technologies de l'IA — Master 1 Data

Contexte du Projet

*Vous êtes **Data Engineer / Scientist** dans une startup spécialisée dans l'e-commerce de produits technologiques.*
*Votre mission est de concevoir et développer un **système de recommandation intelligent** pour améliorer l'expérience utilisateur et augmenter le chiffre d'affaires.*

Objectifs Business

- ***Augmenter le panier moyen** de 15 %*
- ***Améliorer le taux de conversion** de 20 %*
- ***Réduire le taux de rebond** sur les pages produits*

Contraintes Techniques

- ***Temps réel** : recommandations en < 200 ms*
- ***Volume** : 100 000 utilisateurs actifs, 10 000 produits*
- ***RGPD** : anonymisation et respect de la confidentialité*
- ***Budget** : infrastructure optimisée (coût / performance)*

Objectifs Pédagogiques

À l'issue de ce TP, vous serez capable de :

1. ***Analyser** un besoin métier et définir une stratégie data*
2. ***Concevoir** un pipeline ML de bout en bout*
3. ***Implémenter** différents algorithmes de recommandation*
4. ***Évaluer** les performances avec des métriques adaptées*
5. ***Déployer** un modèle selon les bonnes pratiques MLOps*
6. ***Documenter** votre travail de manière professionnelle*

Structure du Projet

```
TP_Recommandation/
│
├── data/
│   ├── users.csv            # Données utilisateurs
│   ├── products.csv         # Catalogue produits
│   └── interactions.csv     # Historique interactions
│
├── notebooks/
│   ├── 01_EDA.ipynb         # Analyse exploratoire
│   ├── 02_Features.ipynb    # Feature engineering
│   ├── 03_Modeling.ipynb    # Modélisation
│   └── 04_Evaluation.ipynb  # Évaluation
│
├── src/
│   ├── data_pipeline.py     # Pipeline de données
│   ├── models.py            # Modèles de recommandation
│   └── utils.py             # Fonctions utilitaires
│
└── README.md
```

Partie 1 – Analyse Exploratoire des Données

Mission

Analyser les données fournies pour comprendre les patterns d'utilisation et identifier les opportunités.

1.1 — Chargement et Découverte

- *Charger les 3 datasets*
- *Afficher les premières lignes et types de données*
- *Identifier les valeurs manquantes*

Questions à répondre :

- *Nombre d'utilisateurs uniques ?*
- *Nombre de produits différents ?*
- *Distribution des interactions par utilisateur ?*
- *Taux d'interactions par produit ?*

1.2 — Analyse Statistique

- *Statistiques descriptives*
- *Visualisation temporelle des interactions*
- *Produits et catégories les plus populaires*

Livrables :

- *Histogrammes de distributions*
- *Courbes temporelles d'activité*
- *Répartition par catégorie*
- *Interprétations métier*

1.3 — Analyse du “Cold Start”

- *Identifier % de nouveaux utilisateurs (< 3 interactions)*
- *Identifier % de nouveaux produits (< 5 interactions)*
- *Proposer des stratégies adaptées*

Partie 2 – Feature Engineering

Mission

*Créer des **features pertinentes** pour alimenter les modèles de recommandation.*

Features Utilisateurs

- `user_activity_level`
- `user_avg_price`
- `user_favorite_category`
- `user_days_since_last_interaction`

Features Produits

- `product_popularity`
- `product_conversion_rate`
- `product_avg_rating`
- `product_price_range`

Features d’Interactions

- `user_product_affinity`

- `similar_users_bought`
- `temporal_features` *(heure, jour, saison)*

Normalisation & Encodage

- *Normaliser (StandardScaler / MinMaxScaler)*
- *Encoder (OneHot / LabelEncoder)*
- *Gérer les valeurs manquantes*

Livrables :

- *Matrice de corrélation*
- *Dataset final prêt pour la modélisation*

Partie 3 – Modélisation

Mission

*Implémenter et comparer **3 approches de recommandation**.*

Approche 1 : Filtrage Collaboratif (User-Based)

***Principe :** Recommander les produits appréciés par des utilisateurs similaires.*

```
from sklearn.metrics.pairwise import cosine_similarity

user_item_matrix = create_matrix(interactions)
user_similarity = cosine_similarity(user_item_matrix)
recommendations = predict_top_k(user_id, user_similarity, k=5)
```

Approche 2 : Filtrage par Contenu (Content-Based)

Principe : *Recommander des produits similaires à ceux déjà appréciés.*

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
product_vectors = tfidf.fit_transform(product_descriptions)
product_similarity = cosine_similarity(product_vectors)
recommendations = get_similar_products(user_history, product_similarity)
```

Approche 3 : Modèle Hybride (ML)

Principe : *Combiner contenu + comportement avec un modèle supervisé.*

```
from sklearn.ensemble import RandomForestClassifier

X = create_feature_matrix(users, products, interactions)
y = interaction_labels  # 1 si interaction, 0 sinon
model = RandomForestClassifier()
model.fit(X_train, y_train)
recommendations = model.predict_proba(X_test)
```

Gestion du Cold Start

- *Popularité globale*
- *Questionnaire utilisateur*
- *Approche hybride*

Livrables :

- *Code des 3 modèles*
- *Comparaison & documentation*

Partie 4 – Évaluation et Métriques

Métriques de Ranking

Nom	***Formule***	***Description***
Precision@K	*(Items pertinents dans le top-K) / K*	*Pertinence des recommandations*
Recall@K	*(Items pertinents dans le top-K) / (Total pertinents)*	*Couverture*
MAP	*Moyenne des AP pour tous les utilisateurs*	*Moyenne pondérée*
NDCG	*DCG / IDCG*	*Pondère par la position*

La Precision@K mesure la proportion d'éléments pertinents parmi les K premiers items recommandés à un utilisateur.

La Recall@K mesure la proportion d'items pertinents qui ont été correctement recommandés parmi l'ensemble des items pertinents disponibles.

La Mean Average Precision (MAP) mesure la qualité globale des classements produits par un système de recommandation sur l'ensemble des utilisateurs.

Le NDCG@K évalue la ***qualité du classement*** *en prenant en compte* ***l'ordre des items*** *dans les recommandations.*

Métriques Business

- *Taux de couverture*
- *Diversité*
- *Nouveauté*
- *Latence moyenne*

Validation Croisée

- *Split temporel (80/20)*
- *Justification du choix*
- *Tests multi-périodes*

Comparaison des Modèles

Modèle	***Precision@5***	***Recall@10***	***MAP***	***NDCG@10***	***Latence (ms)***
User-Based CF	*…*	*…*	*…*	*…*	*…*
Content-Based	*…*	*…*	*…*	*…*	*…*
Hybride	*…*	*…*	*…*	*…*	*…*

Livrables :

- *Graphiques comparatifs*
- *Courbes Precision-Recall*
- *Analyse critique*

Partie 5 – Pipeline & MLOps

Pipeline de Données

```
class DataPipeline:
    def load_data(self):
        """Charger les données"""
        pass

    def clean_data(self):
        """Nettoyage et validation"""
        pass

    def engineer_features(self):
        """Feature engineering"""
        pass

    def run(self):
        """Pipeline complet"""
        pass
```

Versionnement avec MLflow

```
import mlflow

with mlflow.start_run():
    mlflow.log_params({"n_neighbors": 10, "metric": "cosine"})
    mlflow.log_metrics({"precision@5": 0.82, "map": 0.76})
    mlflow.sklearn.log_model(model, "recommendation_model")
```

API FastAPI

```
from fastapi import FastAPI
app = FastAPI()

@app.get("/recommend/{user_id}")
def get_recommendations(user_id: int, top_k: int = 5):
    recommendations = model.predict(user_id, top_k)
    return {"user_id": user_id, "recommendations": recommendations}
```

Monitoring

- *Drift des données*
- *Dégradation des performances*
- *Latence*
- *Taux d'utilisation*

Livrables :

- *Pipeline complet*
- *Documentation technique*
- *Dockerfile (bonus)*
- *Dashboard de suivi (bonus)*

Partie 6 – Documentation

README.md

Inclure : description, architecture, installation, utilisation, performances, auteurs.

Model Card

- *Objectif*
- *Données d'entraînement*
- *Métriques*
- *Limites & biais*
- *Considérations RGPD*

Rapport d'Analyse

1. *Contexte et objectifs*
2. *Méthodologie*
3. *Résultats & performances*
4. *Recommandations*

Livrables :

- *README complet*
- *Model Card*
- *Rapport PDF*

Bonus

Niveau	***Description***
1 – Technique	*Deep Learning, embeddings, Optuna*
2 – Production	*Docker, Streamlit, A/B testing*
3 – Innovation	*LLM explicable, multi-agents, feedback loop*

Ressources

- *Scikit-learn*
- *Surprise (CF)*
- *MLflow*
- *FastAPI*
- *Recommender Systems Tutorial*
- *Metrics for Recommenders*

Modalités de Rendu

- *Format : `.zip` contenant code + notebooks + rapport*
- *Nom : `TP1_Nom_Prenom.zip`*
- *Date limite : à compléter*
- *Mode de dépôt : plateforme à préciser*

Bon courage !

> *Ce TP vous prépare à la mise en production réelle d'un système de recommandation IA.*