Cahier des Charges

Système de Pricing Dynamique par Reinforcement Learning pour E-commerce

Version 1.0

25 novembre 2025

Client : Entreprise E-commerce
Équipe : Data Science & AI Engineering
Statut : Validation en cours

Table des matières

1 Contexte et Objectifs

1.1 Contexte

Les entreprises e-commerce doivent ajuster dynamiquement leurs prix en fonction de multiples facteurs pour maximiser leurs profits. Les méthodes traditionnelles présentent des limitations significatives :

— Règles fixes sous-optimales ne s'adaptant pas au marché dynamique
— A/B testing long et coûteux
— Réactivité limitée face aux concurrents
— Gestion non optimisée des stocks

1.2 Objectifs

Objectif Principal : Développer un agent RL autonome capable d'ajuster les prix quotidiennement avec une amélioration cible de 15-40% des profits par rapport aux méthodes traditionnelles.

Objectifs Secondaires :
— Optimisation de la rotation des stocks
— Réduction des ruptures et surplus de stock
— Automatisation complète des décisions de pricing
— Intégration transparente aux plateformes e-commerce existantes
— Apprentissage continu et adaptation aux changements de marché

2 Spécifications Fonctionnelles

2.1 Fonctionnalités Principales

2.1.1 Décisions de Prix Automatisées

— **Fréquence :** Ajustements quotidiens des prix
— **Actions :** 5 actions discrètes : [-10%, -5%, 0%, +5%, +10%] du prix courant
— **Produits :** Gestion individuelle par produit avec possibilité d'extension multi-produits

2.1.2 Facteurs de Décision

L'agent doit prendre en compte :

— Niveau de stock actuel et coût de possession
— Prix des 2-3 principaux concurrents
— Historique de demande (7 derniers jours)
— Saisonnalité (jour de semaine, période de l'année)
— Coût unitaire du produit
— Contraintes business spécifiques

2.2 Interfaces Utilisateur

2.2.1 Environnements

— **Simulation :** Environnement de test et validation
— **Production :** Interface avec données réelles
— **Monitoring :** Dashboard de suivi des performances

2.2.2 Fonctionnalités Interface

— Configuration des produits (coût, stock initial, contraintes)
— Visualisation des performances (profits, évolution des prix)
— Override manuel des décisions automatiques
— Reporting et analytics avancés

3 Spécifications Techniques

3.1 Stack Technologique

```
# Langages et Frameworks
Python 3.8+
Stable-Baselines3 + Zoo (RL)
Gymnasium (Environnements)
Pandas, NumPy (Data Processing)
Matplotlib, Seaborn (Visualisation)
FastAPI (D ploiement API)
```

Listing 1 – Stack technique principale

3.2 Infrastructure

3.2.1 Environnement de Développement

— **Local :** Machine de développement avec accès GPU optionnel

— **Versioning :** Git avec repository privé
— **CI/CD :** Pipeline d'intégration continue

3.2.2 Environnement de Production

— **Serveurs :** Infrastructure cloud scalable
— **API : Service REST avec authentification**
— **Base de données :** PostgreSQL pour le stockage des décisions et métriques
— **Monitoring :** Logging structuré et alerting

4 Modélisation Reinforcement Learning

4.1 Espace d'État (State Space)

```
state = {
    'stock_actuel': float,           # Normalis  [0,1]
    'prix_actuel': float,            # Normalis
    'prix_concurrent_1': float,      # Normalis
    'prix_concurrent_2': float,      # Normalis
    'demande_j-1': float,            # Normalis
    'demande_j-2': float,            # Normalis
    'demande_j-7': float,            # Normalis
    'jour_semaine': int,             # One-hot encoded
    'saison': int,                   # One-hot encoded
    'cout_unitaire': float           # Normalis
}
```

Listing 2 – Définition de l'état

4.2 Espace d'Actions (Action Space)

```
actions = [
    'baisser_10_pourcent',    # -10% du prix actuel
    'baisser_5_pourcent',     # -5% du prix actuel
    'maintenir_prix',         # 0% changement
    'augmenter_5_pourcent',   # +5% du prix actuel
    'augmenter_10_pourcent'   # +10% du prix actuel
]
```

Listing 3 – Définition des actions

4.3 Fonction de Récompense

```python
def calculate_reward(self, profit, price_change, sales, stock_level):
    # R compense de base normalis e
    base_reward = profit / 100.0

    # P nalit s pour contraintes business
    stock_penalty = self._calculate_stock_penalty(stock_level)
    change_penalty = abs(price_change) * 2

    # Bonus strat giques
    strategic_bonus = self._calculate_strategic_bonus(sales, stock_level
    )

    return base_reward - stock_penalty - change_penalty +
    strategic_bonus
```

Listing 4 – Fonction de récompense

4.4 Algorithmes RL

— **PPO :** Algorithm de choix pour sa stabilité
— **DQN :** Alternative pour espaces d'actions discrets
— **A2C :** Pour comparaison de performances
— **Sélection :** Benchmark via SB3 Zoo

5 Données et Simulation

5.1 Simulateur de Marché

5.1.1 Modèle de Demande

```python
def simulate_demand(self, price, competitor_prices, season, day_type):
    #    lasticit   -prix
    price_effect = (price / avg_competitor_price) ** self.
    price_elasticity

    # Effets saisonniers
    seasonal_effect = 1.0 + 0.3 * np.sin(2 * np.pi * self.day / 30)

    # Effet jour de semaine
    day_effect = 1.2 if day_type >= 5 else 1.0

    # Bruit de march   r aliste
    noise = np.random.normal(1.0, 0.15)

    return base_demand * price_effect * seasonal_effect * day_effect *
    noise
```

Listing 5 – Modèle de demande réaliste

5.1.2 Comportement des Concurrents

— Variations aléatoires autour d'un prix de base
— Réactivité aux changements de nos prix
— Comportements agressifs ou conservateurs simulés

5.2 Stratégie Données

5.2.1 Données d'Entraînement

— **Prioritaire :** Données historiques réelles si disponibles
— **Alternative :** Données synthétiques calibrées sur cas réels
— **Validation :** Split temporel pour évaluation hors échantillon

5.2.2 Qualité des Données

— Nettoyage et préprocessing automatique
— Détection d'anomalies
— Gestion des valeurs manquantes

6 Entraînement et Évaluation

6.1 Stratégie d'Entraînement

6.1.1 Configuration

— **Volume :** 100,000 à 1,000,000 de steps
— **Optimisation :** Environnements vectorisés parallèles
— **Hyperparamètres :** Basés sur SB3 Zoo best practices
— **Tracking :** Métriques temps réel avec TensorBoard

6.1.2 Processus

1. Entraînement initial sur données simulées
2. Fine-tuning sur données réelles si disponibles
3. Validation croisée sur multiples scénarios
4. Optimisation itérative des hyperparamètres

6.2 Méthodologie d'Évaluation

6.2.1 Benchmark Baselines

```
baselines = {
    'prix_fixe': "Prix constant sur la p riode",
    'marge_constante': "Co t + pourcentage fixe",
    'suiveur_concurrent': "Alignement sur prix march ",
    'weekend_boost': "Prix variables jour/semaine",
    'clearance_pricing': "R ductions progressives"
}
```

Listing 6 – Stratégies de comparaison

6.2.2 Métriques de Performance

- **Financières :** Profit total, marge moyenne, ROI
- **Commerciales :** Rotation stock, taux écoulement, taux service
- **Techniques :** Stabilité prix, consistance décisions
- **Statistiques :** Intervalles confiance, tests A/B

6.2.3 Tests de Robustesse

- Scénario concurrent agressif
- Pic de demande soudain
- Pénurie de stock
- Changement de saisonnalité

6.3 Visualisation des Résultats

6.3.1 Dashboard Principal

- Courbes d'apprentissage et convergence
- Évolution des prix vs concurrence
- Analyse distribution des actions
- Métriques business temps réel

6.3.2 Rapports Automatisés

- Rapport de performance hebdomadaire
- Analyse comparative vs baselines
- Détection d'anomalies et alertes
- Recommendations d'amélioration

7 Déploiement et Intégration

7.1 Phase Pilote

7.1.1 Planning Pilote

— **Durée :** 2-4 semaines
— **Périmètre :** 5-10% du catalogue produits
— **Objectif :** Validation performance réelle

7.1.2 Mesures de Sécurité

— Limites de prix (min : coût + 10%, max : coût × 3)
— Validation manuelle obligatoire pour certains écarts
— Mode dégradé avec fallback sur règles traditionnelles
— Monitoring renforcé pendant la phase pilote

7.2 Architecture de Production

7.2.1 API de Service

```
POST /api/v1/pricing/decision
{
    "product_id": "string",
    "current_stock": "float",
    "current_price": "float",
    "competitor_prices": ["float"],
    "demand_history": ["float"],
    "timestamp": "datetime"
}

Response:
{
    "recommended_price": "float",
    "price_change": "float",
    "confidence": "float",
    "explanation": "string"
}
```

Listing 7 – Endpoint décision prix

7.2.2 Data Pipeline

— Collecte automatique des données marché

— Prétraitement et validation des inputs
— Logging structuré des décisions
— Archivage des métriques de performance

7.3 Supervision et Monitoring

7.3.1 Métriques de Surveillance

— Temps de réponse API (< 1 seconde)
— Exactitude des prédictions
— Dérive des performances
— Utilisation ressources système

7.3.2 Système d'Alerte

— Prix anormalement haut/bas
— Chute brutale des ventes
— Dégradation des performances
— Problèmes techniques système

8 Planning et Livrables

8.1 Planning Détaillé (10 semaines)

— **Semaines 1-2 :** Conception simulateur et modèles données
  — Design environnement Gymnasium
  — Modélisation économiques réaliste
  — Spécifications données d'entraînement
— **Semaines 3-4 :** Implémentation environnement et intégration SB3
  — Développement environnement custom
  — Intégration Stable-Baselines3 ètreConfiguration pipeline entraînement
— **Semaines 5-6 :** Entraînement et évaluation algorithmes
  — Benchmark multi-algorithmes
  — Optimisation hyperparamètres
  — Validation performances
— **Semaines 7-8 :** Développement API et interface
  — API REST de production
  — Dashboard monitoring

    - Système de logging
- **Semaines 9-10 :** Pilote et ajustements
    - Déploiement pilote
    - Collecte feedback
    - Optimisations finales

8.2 Livrables Principaux

8.2.1 Livrables Techniques

- Code source complet avec documentation
- Modèles entraînés et configurations optimales
- API de production avec documentation Swagger
- Scripts de déploiement et configuration

8.2.2 Livrables Documentation

- Documentation technique détaillée
- Guide d'installation et utilisation
- Manuel d'administration et maintenance
- Procédures de troubleshooting

8.2.3 Livrables Analyse

- Rapport d'expérimentation complet
- Analyse comparative des performances
- Recommendations déploiement à l'échelle
- Plan d'évolution et améliorations futures

9 Équipe et Budget

9.1 Composition de l'Équipe

9.1.1 Ressources Humaines

- **Ingénieur RL/Data Scientist (1 ETP)**
    - Développement algorithmes RL
    - Entraînement et optimisation modèles

  - Analyse performances
- **Data Engineer (0.5 ETP)**
  - Pipeline données et API
  - Déploiement infrastructure
  - Monitoring et logging

9.2 Budget Détaillé

9.2.1 Ressources Humaines

- **Ingénieur RL :** 10 semaines × [Taux journalier]
- **Data Engineer :** 5 semaines × [Taux journalier]
- **Total RH :** 15 semaines-homme

9.2.2 Infrastructure

- **Cloud Computing :** 500-1000€ (entraînement + déploiement)
- **Outils et Licences :** 200-500€
- **Contingence :** 15% du budget total

9.2.3 Total Budget Estimé

- **Total :** [À compléter selon taux journaliers]
- **Validité :** 30 jours

9.3 Critères de Succès

9.3.1 Techniques

- Convergence stable de l'apprentissage
- Performance 15% meilleure que baselines
- Temps de réponse < 1 seconde
- Disponibilité $> 99\%$ en production

9.3.2 Business

- Amélioration mesurable des profits
- Réduction des stocks invendus

— Satisfaction utilisateurs (équipe commerciale)
— ROI positif sous 6 mois

10 Annexes

10.1 Glossaire

— **RL :** Reinforcement Learning
— **SB3 :** Stable-Baselines3
— **PPO :** Proximal Policy Optimization
— **DQN :** Deep Q-Network
— **A2C :** Advantage Actor-Critic

10.2 Références

— Documentation Stable-Baselines3 : `https://stable-baselines3.readthedocs.io/`
— SB3 Zoo : `https://github.com/DLR-RM/rl-baselines3-zoo`
— Gymnasium : `https://gymnasium.farama.org/`