TechNova Partners : Déployer un modèle ML

API de prédiction du risque de départ des salariés (HR Attrition)

Ghislain Delabie

Projet n°5 - Formation AI engineer

21/11/2025

TECHNOVA PARTNERS

Enjeu RH : anticiper le départ des collaborateurs

Constat :

- Le turnover représente un coût important (recrutement, formation, perte de savoir-faire)
- Les RH ont besoin d'outils pour détecter les risques en amont
- Les approches manuelles ne sont ni scalables ni systématiques

Réponse proposée :

- Un modèle de Machine Learning qui estime la probabilité de départ d'un salarié
- Une API REST intégrable dans les outils RH existants
- Une capacité de traitement unitaire et en batch pour analyser des populations entières

Impact attendu :

- Mise en place de politiques de rétention plus ciblées
- Réduction des coûts liés au turnover
- Prise de décision RH davantage pilotée par la donnée

Objectifs techniques

Objectif techniques

- Entraîner un modèle XGBoost performant (ROC AUC ~0.85)
- Déployer une API REST production-ready
- Assurer la traçabilité avec une base de données
- Fournir l'interprétabilité via SHAP

Livrable final :

API publique déployée sur Hugging Face Spaces avec CI/CD et documentation

Pile technologique

Objectif du projet :

Mettre en production un modèle de Machine Learning capable de prédire le risque de départ des collaborateurs, via une API exploitable par les équipes RH.

Pile technologique :

- Modèle de classification XGBoost pour la prédiction d'attrition
- API REST développée avec FastAPI
- Docker pour la conteneurisation
- GitHub Actions pour le CI/CD
- Hugging Face Spaces pour l'hébergement et le déploiement

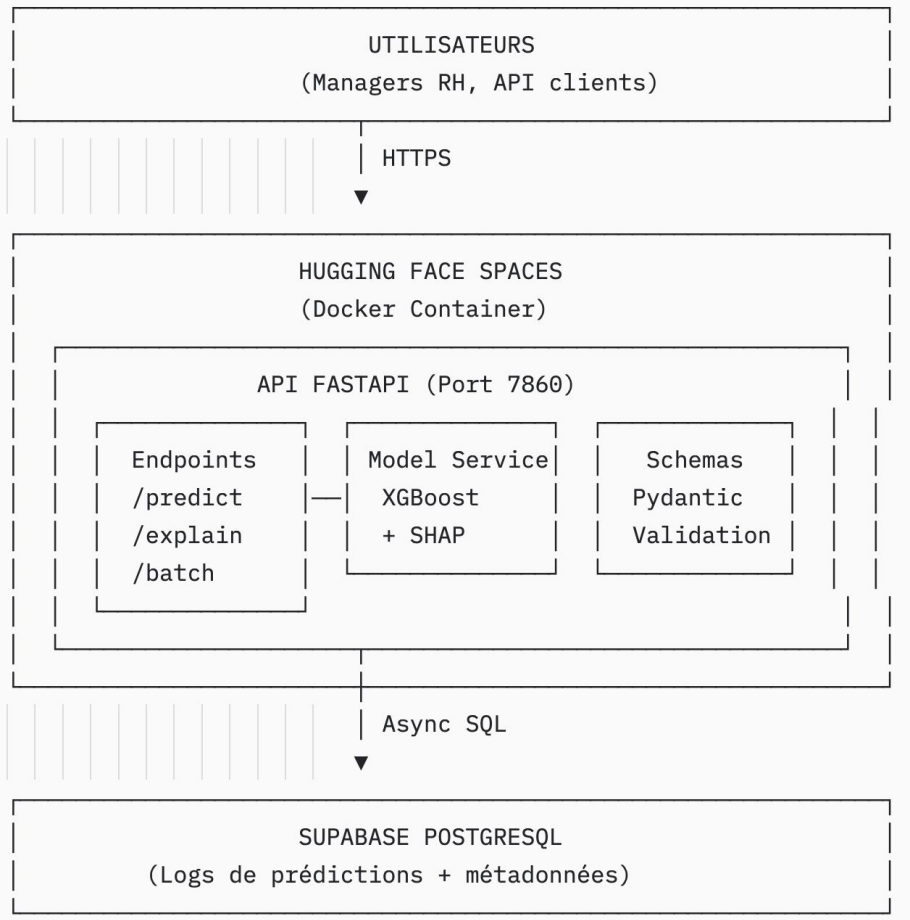
Statut du projet :

- Version v0.1.0 déployée et opérationnelle
- API en ligne : https://ghislaindelabie-oc5-ml-api.hf.space

Réalisations clés :

- Chaîne complète : de l'entraînement du modèle jusqu'au déploiement production
- 97 % de couverture de tests (65 tests automatisés)
- Pipeline CI/CD entièrement automatisé
- API REST documentée et prête à l'intégration

Architecture globale



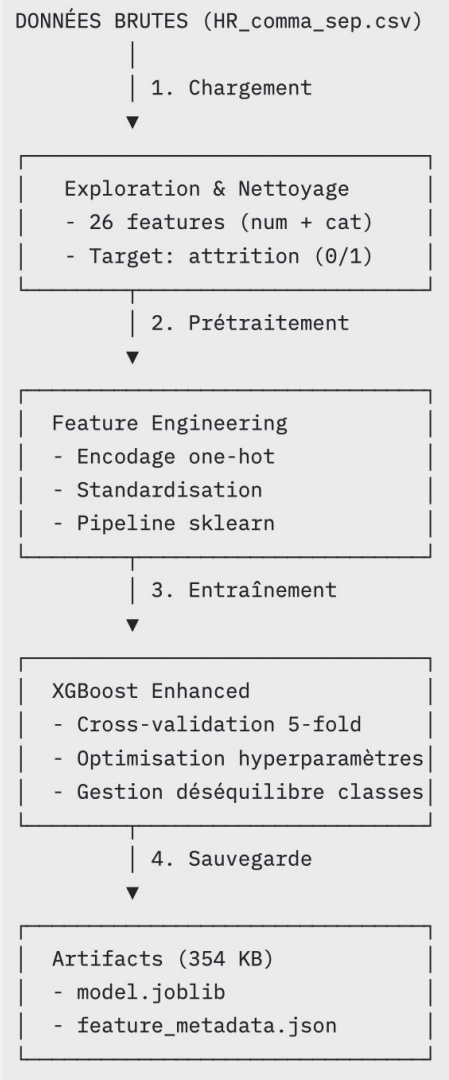
Composants principaux

- **FastAPI**: Framework web moderne et performant
- **XGBoost**: Modèle de classification pré-entraîné
- **SHAP**: Explications des prédictions
- **PostgreSQL**: Traçabilité et audit
- **Docker**: Conteneurisation pour déploiement

Pipeline d’entraînement du

Performances obtenues (cross-validation)

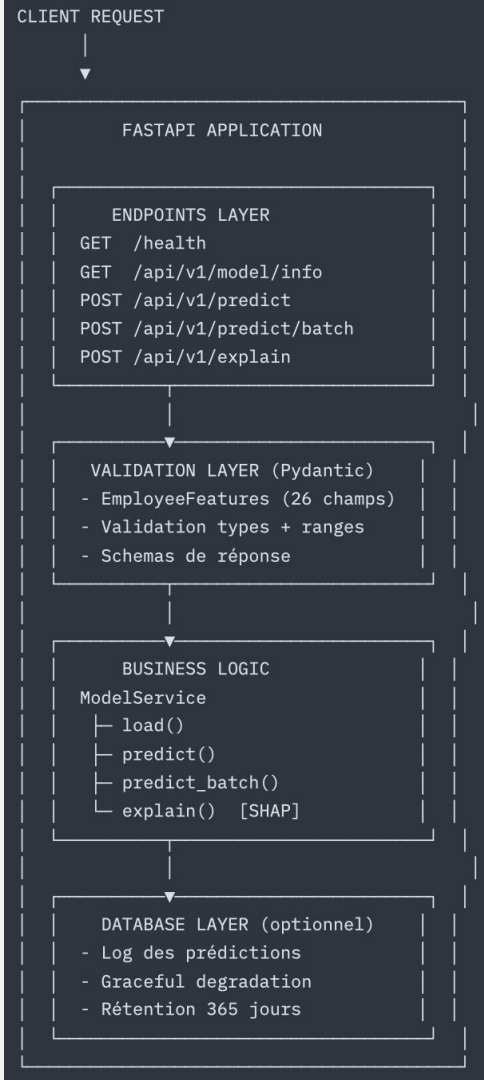
- **Accuracy**: 85.6%
- **Precision**: 62.5% @ 50% recall
- **ROC AUC**: 0.85
- **F1-Score**: 0.64



Pipeline d’entraînement du

Principes de design

- **Separation of concerns:** Couches distinctes
- **Dependency injection:** Services singleton
- **Graceful degradation:** L'API fonctionne sans BDD
- **Type safety:** Validation via Pydantic

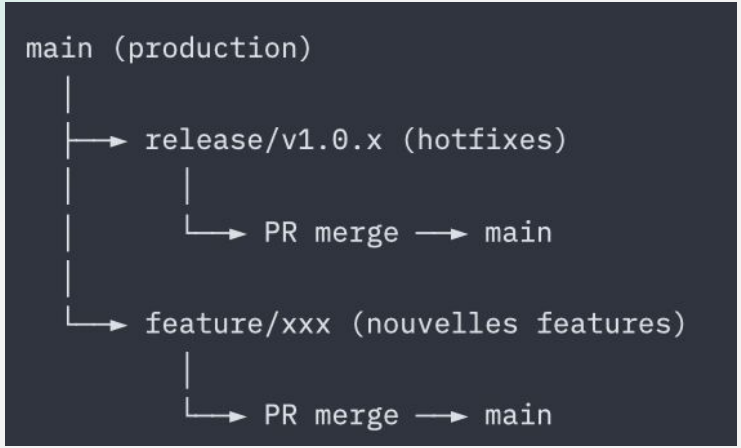


Choix d’implémentation

Aspect	Choix	Justification
Framework API	FastAPI	- Performance élevée (async)- Documentation auto (Swagger)- Validation via Pydantic- Standard moderne Python
Modèle ML	XGBoost	- Performance supérieure- Gestion déséquilibre- Compatible SHAP- Rapide en inférence
Interprétabilité	SHAP	- Standard industrie- Explications fiables- TreeExplainer optimisé- Top 5 features (~50ms)
Base de données	PostgreSQL (Supabase)	- SQL robuste et performant- Hébergement cloud gratuit- Async via asyncpg- Migrations Alembic
Déploiement	Docker + HF Spaces	- Reproductibilité totale- Gratuit et public- CI/CD intégré- URL stable
Tests	pytest + coverage	- 75 tests automatisés- 76.68% de couverture- CI obligatoire- Qualité garantie

Workflow Git et CI/CD

Git Workflow



Avantages

- Zero-downtime deployment
- Tests automatiques obligatoires
- Review code automatique
- Traçabilité complète

Pipeline CI/CD GitHub Actions)

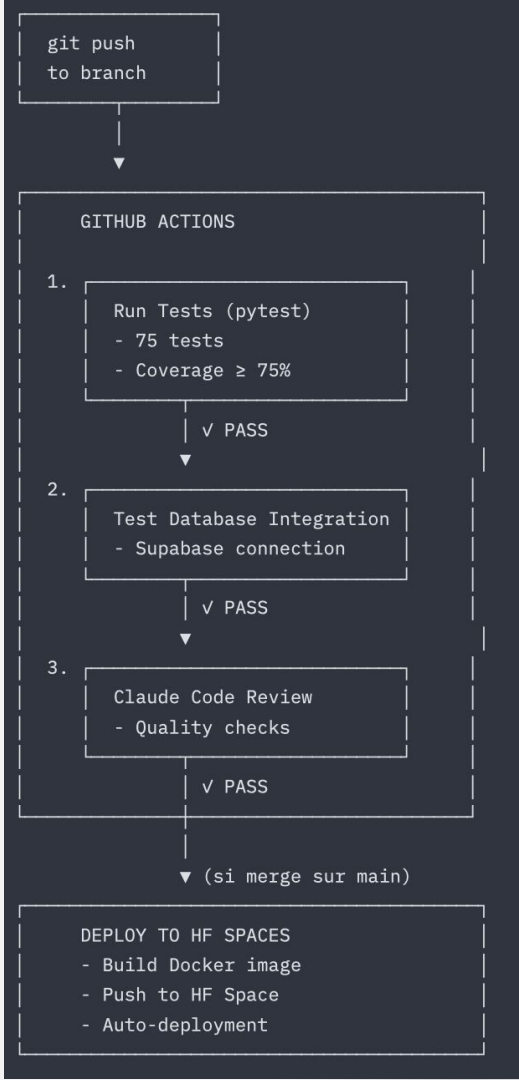


Diagramme UML des classes

Fonctionnalités implémentées

1. Prédiction simple (`POST /api/v1/predict`)

```json
Input: {
  "age": 35,
  "revenu_mensuel": 5000,
  "satisfaction_employee_environnement": 4,
  ...
}
```

```
Output: {
  "prediction": {
    "will_leave": false,
    "probability_leave": 12.5,
    "probability_stay": 87.5,
    "risk_level": "LOW"
  },
  "metadata": { "prediction_time_ms": 25 }
}
```

Fonctionnalités implémentées

2. Prédiction batch (`POST /api/v1/predict/batch`)

```json
Input: {
  "age": 35,
  "revenu_mensuel": 5000,
  "satisfaction_employee_environnement": 4,
  ...
}
```

```
Output: {
  "prediction": {
    "will_leave": false,
    "probability_leave": 12.5,
    "probability_stay": 87.5,
    "risk_level": "LOW"
  },
  "metadata": { "prediction_time_ms": 25 }
}
```

Fonctionnalités implémentées

3. Explications SHAP (`POST /api/v1/explain`)

```json
Output: {
  "top_features": [
    {
      "feature": "heure_supplementaires_Yes",
      "shap_value": 0.1523,
      "impact": "increases risk"
    },
    { "feature": "satisfaction_employee_environnement",
      "shap_value": -0.0654,
      "impact": "decreases risk"
    },
    ...
  ]
}
```

Assurance qualité et couverture des tests

Contexte

```
tests/
  ├─ test_api_integration.py      (16 tests)
  │    └─ Tests end-to-end des endpoints
  │
  ├─ test_api_contracts.py        (14 tests)
  │    └─ Validation des schemas Pydantic
  │
  ├─ test_model_service.py        (18 tests)
  │    └─ Logique métier du modèle
  │
  ├─ test_database.py             (2 tests)
  │    └─ Intégration base de données
  │
  ├─ test_api_errors.py           (6 tests)
  │    └─ Gestion des erreurs
  │
  ├─ test_model_edge_cases.py     (2 tests)
  │    └─ Cas limites
  │
  ├─ test_explain.py              (2 tests)
  │    └─ Explications SHAP
  │
  └─ ... (15 autres tests)
```

Métriques de qualité

- 75 tests automatisés (pytest)
- 76.68% de couverture de code
- 0 erreur en production

CI obligatoire avant merge
Review code automatique (Claude)

Démonstration et liens

🌐 **API publique en production:** https://ghislaindelabie-oc5-ml-api.hf.space

📚 **Documentation interactive:**
- Swagger UI: `/docs`
- ReDoc: `/redoc`
- Landing page: `/`

💻 **Code source**:** github.com/ghislaindelabie/oc5-deploy-ml

🔑 **Points forts:**
1. API REST production-ready
2. Interprétabilité via SHAP
3. Traçabilité complète (BDD)
4. CI/CD automatisé
5. Tests robustes
6. Documentation exhaustive

Merci!

Ghislain Delabie - Ingénieur IA
www.delabie.tech