Système de Prédiction des Sinistres en Assurance

Une Approche par Machine Learning

Documentation Technique Détaillée

Table des matières

1. Introduction
2. Architecture Technique
3. Modèles de Machine Learning
4. Pipeline de Données
5. API REST
6. Tests et Qualité
7. Déploiement
8. Monitoring et Maintenance
9. Sécurité
10. Perspectives
11. Conclusion

Introduction

Contexte du Projet

Ce projet vise à développer un système de prédiction des sinistres en assurance en utilisant des techniques avancées de machine learning. L'objectif principal est de fournir des estimations précises de la fréquence des sinistres et de leur coût moyen, permettant ainsi une meilleure gestion des risques et une tarification plus précise.

Objectifs

- Prédire la fréquence des sinistres avec une haute précision
- Estimer le coût moyen des sinistres
- Calculer le coût total prévisionnel
- Fournir une API REST pour l'intégration avec d'autres systèmes
- Assurer la scalabilité et la maintenabilité du système

Architecture Technique

Stack Technologique

Technologies Principales

- Python 3.9+
- FastAPI pour l'API REST
- XGBoost pour les modèles de ML
- Docker pour la conteneurisation
- GitHub Actions pour CI/CD
- pytest pour les tests automatisés

Structure du Projet

```
.
├── app.py              # Application FastAPI
├── preprocessing.py    # Prétraitement des données
```

```
├── train_xgboost.py   # Entraînement du modèle
├── predict.py         # Module de prédiction
├── test_api.py        # Tests de l'API
├── requirements.txt    # Dépendances
└── Dockerfile         # Configuration Docker
```

Modèles de Machine Learning

XGBoost

Caractéristiques du Modèle Le projet utilise XGBoost, un algorithme de gradient boosting optimisé, choisi pour : - Sa performance sur les données tabulaires - Sa gestion efficace des valeurs manquantes - Sa capacité à gérer un grand nombre de features - Sa rapidité d'entraînement et d'inférence

Features Engineering Le modèle utilise plus de 200 features, incluant : - Données géographiques (localisation, altitude, etc.) - Informations météorologiques (précipitations, températures, etc.) - Caractéristiques des bâtiments (surface, type, etc.) - Données démographiques (densité de population, etc.) - Historique des sinistres

Pipeline de Données

Prétraitement

Étapes de Prétraitement

1. Nettoyage des données
   - Suppression des doublons
   - Correction des valeurs aberrantes
   - Standardisation des formats
2. Gestion des valeurs manquantes
   - Imputation par la moyenne/médiane
   - Utilisation de stratégies avancées selon le contexte
3. Encodage des variables catégorielles
   - One-Hot Encoding
   - Label Encoding
   - Target Encoding pour certaines variables
4. Normalisation des features numériques
   - StandardScaler pour les distributions normales
   - RobustScaler pour les données avec outliers

API REST

Endpoints

Prédiction de Fréquence

```python
@app.post("/predict_freq")
async def predict_frequency(data: Dict):
    # Retourne la fréquence prédite des sinistres
    return {"freq": predicted_freq}
```

Prédiction de Coût

```python
@app.post("/predict_cm")
async def predict_cost(data: Dict):
    # Retourne le coût moyen prédit
    return {"cm": predicted_cost}
```

Prédiction Complète

```python
@app.post("/predict_sinistre")
async def predict_complete(data: Dict):
    # Retourne fréquence, coût et coût total
    return {
        "freq": freq,
        "cm": cost,
        "total_cost": freq * cost
    }
```

Tests et Qualité

Tests Automatisés

Tests Unitaires

- Tests des fonctions de prétraitement
- Tests des modèles de prédiction
- Tests de validation des données

Tests d'Intégration

- Tests des endpoints API
- Tests de bout en bout
- Tests de performance

Déploiement

Conteneurisation

Docker

```dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "app:app", "--host", "0.0.0.0"]
```

CI/CD

GitHub Actions

- Tests automatiques à chaque push
- Build et push des images Docker
- Déploiement automatique

Monitoring et Maintenance

Monitoring

Métriques Surveillées

- Temps de réponse API
- Précision des prédictions
- Utilisation des ressources
- Taux d'erreur

Maintenance

Tâches Régulières

- Mise à jour des dépendances
- Réentraînement des modèles
- Analyse des logs
- Optimisation des performances

Sécurité

Mesures de Sécurité

- Validation des entrées
- Gestion des erreurs
- Rate limiting
- Logs sécurisés

Perspectives

Améliorations Futures

- Interface utilisateur web
- Optimisation continue des modèles
- Ajout de nouvelles features
- Analyse approfondie des erreurs

Conclusion

Le système de prédiction des sinistres combine des technologies modernes et des techniques avancées de machine learning pour fournir des prédictions précises et fiables. Son architecture modulaire et sa suite complète de tests en font une solution robuste et maintenable.