DOCUMENTATION TECHNIQUE - SYSTEME DE PREDICTION DE PRETS

==========

1. VUE D'ENSEMBLE DU PROJET

Description Application web de prédiction de l'acceptation de prêts bancaires utilisant l'intelligence artificielle. Le système analyse les données personnelles et financières d'un demandeur pour prédire la probabilité d'acceptation de son prêt et fournit une analyse détaillée des facteurs influençant cette décision.

Technologies utilisées

- Backend : Python Flask
- Frontend : HTML, CSS, JavaScript
- Base de données : Supabase (PostgreSQL)
- Machine Learning : scikit-learn, SHAP
- Visualisation : matplotlib, seaborn
- Gestion de packages : npm (pour les dépendances frontend)

==========

2. ARCHITECTURE DU SYSTÈME

Structure des dossiers

Racine du projet :

- AfficheBDD.py : Interface Tkinter pour visualiser la BDD
- app.py : Application Flask principale
- loan_data.csv : Dataset d'entraînement
- README.md : Documentation du projet

Dossier ConcerneIa/ : Modules d'intelligence artificielle

- boite_a_IA.joblib : Modèle ML sauvegardé
- Model_Felix_SPOHR.py : Script d'entraînement de modèles
- scaler.joblib : Scaler de normalisation sauvegardé
- testeur.py : Tests et comparaisons de modèles
- trouver_le_meilleur_model.py : Sélection du meilleur modèle

Dossier model/ : Modèles de données et fonctions métier

- fonction_dashboard.py : Fonctions d'analyse SHAP
- Utilisateur.py : Classe Utilisateur (vide)

Dossier modelDao/ : Couche d'accès aux données

- UtilisateurDao.py : DAO pour la gestion des utilisateurs

Dossier static/ : Ressources statiques

- css/ : Feuilles de style
- image/ : Images et graphiques générés
- js/ : Scripts JavaScript
- json/ : Fichiers de données temporaires

Dossier templates/ : Templates HTML

- dashboard.html : Page de résultats
- login.html : Page de connexion
- prevision.html : Page de saisie des données

Dossier node_modules/ : Dépendances JavaScript (Supabase)

- Contient les bibliothèques Supabase pour l'authentification
- Types Node.js pour le développement
- WebSocket et autres utilitaires réseau

==========

3. COMPOSANTS PRINCIPAUX

3.1 Application Flask (app.py)

Routes principales

- GET / : Page de connexion
- GET /prevision : Formulaire de prédiction
- GET /dashboard : Affichage des résultats
- POST /predict : Traitement de la prédiction
- POST /analyse : Analyse SHAP des résultats
- POST /verif-utilisateur : Authentification

Configuration de la base de données SUPABASE_URL = 'https://xcpqafebvepowoxxppsd.supabase.co'

SUPABASE_API_KEY = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'

3.2 Modèle d'intelligence artificielle

Prétraitement des données Encodage des variables catégorielles :

- Sexe : ['male','female'] → [0,1]
- Éducation : ['Associate','Bachelor','Doctorate','High School','Master'] → [0,1,2,3,4]
- Propriété immobilière : ['MORTGAGE','OTHER','OWN','RENT'] → [0,1,2,3]
- Raison du prêt : ['DEBTCONSOLIDATION','EDUCATION',...] → [0,1,2,3,4,5]

Normalisation

- Utilisation de StandardScaler pour normaliser les données numériques
- Sauvegarde du scaler dans scaler.joblib

Modèles testés

1. MLPRegressor (Réseau de neurones)
2. LinearRegression (Régression linéaire)
3. RandomForestRegressor (Forêt aléatoire)
4. GradientBoostingRegressor (Gradient boosting)

3.3 Analyse explicative (SHAP)

Fonction principale : analyse_donnees() def analyse_donnees(modele, caracteristiques, noms_caracteristiques=None, afficher_graphique=True, sauvegarder_graphique=False, nom_fichier='analyse_shap.png')

Fonctionnalités :

- Calcul des valeurs SHAP pour l'explicabilité
- Génération de graphiques en cascade (waterfall)
- Graphiques d'importance des caractéristiques
- Sauvegarde automatique des visualisations

==========

4. INTERFACE UTILISATEUR

4.1 Page de connexion (login.html)

- Authentification simple avec login/mot de passe
- Vérification via la base de données Supabase
- Redirection vers la page de prédiction

4.2 Page de prédiction (prevision.html) Champs de saisie

- Données personnelles : Âge, sexe, diplôme, revenu, expérience
- Situation immobilière : Propriétaire, locataire, hypothèque, autre
- Données du prêt : Montant, raison, taux d'intérêt, pourcentage du revenu
- Historique financier : Historique crédit, score crédit, défauts précédents

4.3 Page de résultats (dashboard.html)

- Pourcentage de prédiction : Probabilité d'acceptation du prêt
- Statut : Accepté/Refusé basé sur un seuil de 70%
- Analyse des facteurs : Graphiques SHAP interactifs
- Raisons détaillées : Explication des facteurs positifs/négatifs

==========

5. BASE DE DONNÉES

5.1 Configuration Supabase

- Provider : Supabase (PostgreSQL managé)
- Table principale : Utilisateur

5.2 DAO (Data Access Object) class UtilisateurDao: def insert_user(self, login, mdp) # Insertion d'utilisateur def verif_user(self, user_a_verif, user_mdp_a_verif) # Vérification

==========

6. ALGORITHMES ET MÉTRIQUES

6.1 Évaluation des modèles

- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- R^2 Score
- Cross-validation (5-fold)

6.2 Seuils de décision

- Prêt accepté : Prédiction ≥ 0.70 (70%)
- Prêt refusé : Prédiction < 0.70 (70%)

6.3 Explicabilité

- SHAP TreeExplainer pour l'interprétation des résultats
- Valeurs SHAP positives/négatives pour chaque caractéristique
- Génération automatique de raisons personnalisées

==========

7. FLUX DE DONNÉES

7.1 Processus de prédiction

1. Saisie utilisateur → Formulaire web
2. Validation → Contrôles JavaScript
3. Normalisation → Application du scaler sauvegardé
4. Prédiction → Modèle ML pré-entraîné
5. Analyse SHAP → Calcul des contributions
6. Sauvegarde → Fichiers JSON temporaires
7. Affichage → Dashboard avec visualisations

7.2 Gestion des fichiers temporaires

- static/json/predictions.json : Résultats de prédiction
- static/json/analyse.json : Données d'analyse SHAP
- static/image/ : Graphiques générés automatiquement

==========

8. SÉCURITÉ ET AUTHENTIFICATION

8.1 Authentification

- Système simple login/mot de passe
- Vérification côté serveur via Supabase
- Pas de gestion de sessions avancée (limitation actuelle)

8.2 Sécurité des données

- Clés API exposées dans le code (à améliorer en production)
- Données de prédiction stockées temporairement
- Pas de chiffrement des mots de passe (limitation actuelle)

==========

9. PERFORMANCE ET OPTIMISATION

9.1 Optimisations actuelles

- Modèles pré-entraînés sauvegardés avec joblib
- Scaler pré-calculé pour éviter le re-fitting
- Génération de graphiques mise en cache

9.2 Limitations identifiées

- Pas de gestion de cache pour les prédictions
- Rechargement des modèles à chaque requête
- Stockage temporaire en fichiers JSON

==========

10. DÉPLOIEMENT ET CONFIGURATION

10.1 Prérequis pip install flask flask-cors numpy scikit-learn joblib matplotlib seaborn shap supabase npm install # Pour les dépendances frontend (Supabase JS)

10.2 Lancement de l'application python app.py

Accès via http://127.0.0.1:5000

10.3 Variables d'environnement recommandées
SUPABASE_URL=https://xcpqafebvepowoxxppsd.supabase.co
SUPABASE_KEY=your_service_role_key_here FLASK_ENV=development
FLASK_DEBUG=True

==========

11. MAINTENANCE ET ÉVOLUTION

11.1 Réentraînement du modèle

- Script trouver_le_meilleur_model.py pour sélection automatique
- Script testeur.py pour comparaison exhaustive des modèles
- Sauvegarde automatique du meilleur modèle

11.2 Améliorations futures recommandées

- Implémentation de sessions utilisateur sécurisées
- Cache Redis pour les prédictions fréquentes
- API REST complète avec documentation OpenAPI
- Tests unitaires et d'intégration
- Monitoring et logging avancés
- Chiffrement des données sensibles
- Interface d'administration pour la gestion des modèles

==========

12. LOGS ET DEBUGGING

12.1 Points de debug actuels

- Affichage console des données reçues
- Logs de vérification utilisateur
- Impression des résultats de prédiction

12.2 Fichiers de debug

- AfficheBDD.py : Interface Tkinter pour visualiser la base de données
- Graphiques SHAP sauvegardés automatiquement pour inspection

==========

13. DÉTAILS TECHNIQUES SUPPLÉMENTAIRES

13.1 Format des données d'entrée Champs obligatoires pour la prédiction :

- person_age : Âge de la personne (numérique)
- person_gender : Sexe (0=homme, 1=femme)
- person_education : Niveau d'éducation (0-4)
- person_income : Revenu annuel (numérique)
- person_emp_exp : Expérience professionnelle en années (numérique)
- person_home_ownership : Statut immobilier (0-3)
- loan_amnt : Montant du prêt demandé (numérique)
- loan_intent : Raison du prêt (0-5)
- loan_int_rate : Taux d'intérêt (numérique)
- loan_percent_income : Pourcentage du revenu (numérique)
- cb_person_cred_hist_length : Longueur historique crédit (numérique)
- credit_score : Score de crédit (numérique)
- previous_loan_defaults_on_file : Défauts précédents (0=non, 1=oui)

13.2 Format de sortie Structure JSON de réponse : { "prediction": [pourcentage_float], "analyse": [{ "nom": "nom_technique", "valeur": valeur_shap, "nom_simplifie": "nom_lisible", "raison": "explication_textuelle" }] }

13.3 Algorithmes de traitement Étapes de normalisation :

1. Conversion des valeurs catégorielles en numériques
2. Application du StandardScaler pré-entraîné
3. Transformation en array numpy 1D pour le modèle
4. Prédiction via le modèle RandomForest
5. Calcul des valeurs SHAP pour l'explicabilité

==========

Cette documentation technique couvre l'ensemble des aspects du système de prédiction de prêts, de l'architecture générale aux détails d'implémentation spécifiques.