Système Avancé de Prédiction de la Performance des Candidats

Analyse Comparative d'Algorithmes de Machine Learning pour l'Optimisation du Recrutement

Taha Bchir - Hadi chouaieb

ISET Sfax

DSI 34

Rapport de Projet de Machine Learning

Table des matières

Liste des tableaux

Résumé Étendu

Ce rapport présente une étude exhaustive sur la prédiction de la performance des candidats dans le domaine des ressources humaines, utilisant des techniques avancées de machine learning. Le système développé vise à assister les professionnels RH dans leur processus de sélection en fournissant des prédictions basées sur des données historiques.

Méthodologie

Nous avons implémenté et comparé neuf algorithmes de classification différents, chacun avec ses caractéristiques et avantages spécifiques. Le pipeline de traitement comprend :

- Un nettoyage rigoureux des données pour traiter les valeurs manquantes et aberrantes
- Une transformation appropriée des variables catégorielles et numériques
- Une optimisation systématique des hyperparamètres pour chaque algorithme
- Une évaluation robuste via validation croisée stratifiée

Résultats Principaux

- Meilleur modèle : Gradient Boosting Classifier (F1-score : 0.665)
- Variables les plus importantes : Score test technique (33.9%), Années d'expérience (32.6%)
- Performance sur test : F1-score de 0.640 avec une AUC de 0.620

Applications Pratiques

Le système a été intégré dans une application Streamlit interactive permettant :

- Des prédictions en temps réel pour des candidats individuels
- Le traitement par lots de fichiers CSV
- La visualisation des facteurs d'influence sur les décisions

1 Introduction

1.1 Contexte et Motivation

Dans le paysage concurrentiel actuel du recrutement, les organisations cherchent constamment à optimiser leurs processus de sélection. L'identification des candidats qui réussiront à long terme représente un défi complexe, souvent basé sur des critères subjectifs et des jugements intuitifs. Ce projet s'inscrit dans le mouvement croissant de la "HR Analytics", qui vise à apporter une approche data-driven aux décisions de ressources humaines.

1.2 Problématique

Le recrutement traditionnel présente plusieurs limitations :

- **Subjectivité** : Les biais cognitifs influencent les décisions
- **Inefficacité** : Processus longs et coûteux
- **Risque d'erreur** : Mauvais recrutements entraînant des coûts importants
- **Manque de prédictibilité** : Difficulté à anticiper la performance future

1.3 Objectifs Spécifiques

1. Développer un modèle prédictif robuste pour identifier les candidats performants
2. Analyser l'importance relative des différents facteurs de recrutement
3. Créer une interface utilisateur intuitive pour les praticiens RH
4. Établir un cadre méthodologique reproductible pour l'analyse RH
5. Fournir des recommandations actionnables pour l'amélioration des processus

1.4 Cadre Théorique

Le projet s'appuie sur plusieurs domaines théoriques :

- **Psychologie organisationnelle** : Théories de la performance au travail
- **Statistiques multivariées** : Techniques d'analyse des données
- **Machine Learning** : Algorithmes de classification supervisée
- **Data Mining** : Extraction de connaissances à partir de données

2 Description et Analyse des Données

2.1 Source et Généalogie des Données

Le jeu de données utilisé a été généré synthétiquement pour reproduire les caractéristiques d'un ensemble de données RH réaliste. La génération a respecté les contraintes suivantes :

— Distribution d'âge réaliste (loi normale tronquée entre 18 et 70 ans)
— Corrélations entre variables cohérentes avec la littérature
— Introduction de bruits réalistes (valeurs manquantes, aberrantes)
— Déséquilibre modéré des classes (53% vs 47%)

2.2 Analyse des Valeurs Manquantes

TABLE 1 – Analyse des valeurs manquantes par variable

2whitetableheader

tableheaderVariable	tableheaderNbre Manquants	tableheaderPourcentage	tableheaderType	tab
score_test_technique	45	4.5%	Numérique	
score_softskills	38	3.8%	Numérique	
secteur_précédent	67	6.7%	Catégorique	
langues_parlées	22	2.2%	Numérique	
mobilité	15	1.5%	Binaire	

2.3 Analyse du Déséquilibre des Classes

TABLE 2 – Répartition de la variable cible

2whitetableheader

tableheaderClasse	tableheaderEffectif	tableheaderPourcentage	tableheaderDescription
Non Performant	470	47.0%	Candidats n'ayant pas satisfait aux
Performant	530	53.0%	Candidats ayant réussi leur intég
Total	1000	100%	Ensemble complet

3 Méthodologie Détaillée

3.0.1 Techniques d'Imputation

Nous avons utilisé plusieurs stratégies d'imputation :

— **Imputation par médiane** : Pour les variables numériques avec valeurs aberrantes

— **Imputation par KNN** : Testé pour les variables corrélées
— **Imputation par mode** : Pour les variables catégorielles
— **Indicateurs de valeurs manquantes** : Création de variables binaires supplémentaires

3.0.2 Encodage des Variables Catégorielles

TABLE 3 – Comparaison des techniques d'encodage testées

2whitetableheader

tableheaderMéthode	tableheaderDescription	tableheaderPerformance
One-Hot Encoding	Création de variables binaires pour chaque catégorie	
Label Encoding	Transformation en valeurs numériques séquentielles	
Target Encoding	Encodage basé sur la moyenne de la variable cible	
Binary Encoding	Conversion en code binaire puis séparation	
Choix final	**One-Hot Encoding avec drop_first=True**	

3.0.3 Scaling et Normalisation

Trois méthodes ont été comparées :

— **StandardScaler** : Standardisation Z-score (moyenne=0, écart-type=1)
— **MinMaxScaler** : Normalisation entre 0 et 1
— **RobustScaler** : Utilisation de médiane et IQR, robuste aux outliers

3.1 Algorithmes Implémentés : Analyse Théorique

3.1.1 1. Régression Logistique

Principe : Modèle linéaire qui modélise la probabilité d'appartenance à une classe via la fonction logistique.

Avantages :

— Interprétabilité des coefficients
— Rapidité d'entraînement et de prédiction
— Faible risque d'overfitting avec régularisation

Limitations :

— Hypothèse de linéarité peu réaliste
— Sensible aux variables corrélées
— Performance limitée sur relations complexes

3.1.2 2. Forêts Aléatoires (Random Forest)

Principe : Ensemble d'arbres de décision indépendants, chacun entraîné sur un sous-échantillon bootstrap des données avec sélection aléatoire de features.

Avantages :

— Robustesse au bruit et aux outliers
— Estimation naturelle de l'importance des variables
— Pas besoin de scaling des features
— Bonne performance sur données non linéaires

Limitations :

— Modèle "black-box" moins interprétable
— Computationally intensive pour grands datasets
— Peut overfitter sur données bruitées

3.1.3 3. XGBoost (Extreme Gradient Boosting)

Principe : Algorithme de boosting gradient optimisé avec régularisation L1/L2 et techniques avancées de gestion des missing values.

Avantages :

— Excellentes performances compétitives
— Gestion native des valeurs manquantes
— Régularisation intégrée contre l'overfitting
— Optimisation parallèle efficace

Limitations :

— Sensible aux hyperparamètres
— Temps d'entraînement potentiellement long
— Requiert un tuning minutieux

3.1.4 4. LightGBM

Principe : Framework de gradient boosting utilisant l'algorithme Gradient-based One-Side Sampling (GOSS) et Exclusive Feature Bundling (EFB).

Avantages :

- Entraînement extrêmement rapide
- Faible consommation mémoire
- Bonne performance sur grands datasets
- Supporte les données catégorielles natives

Limitations :

- Peut overfitter sur petits datasets
- Sensible aux hyperparamètres
- Moins robuste aux outliers que Random Forest

3.1.5 5. Gradient Boosting Classifier

Principe : Algorithme séquentiel construisant un ensemble d'arbres faibles, où chaque nouvel arbre corrige les erreurs des précédents.

Avantages :

- Excellente performance prédictive
- Flexibilité avec différentes fonctions de perte
- Peut modéliser des relations complexes
- Interprétabilité via feature importance

Limitations :

- Sensible au bruit dans les données
- Temps d'entraînement significatif
- Requiert un tuning soigné

3.1.6 6. AdaBoost

Principe : Adaptive Boosting, ajuste les poids des instances mal classées à chaque itération.

Avantages :

- Simple et efficace
- Réduit à la fois bias et variance
- Peu d'hyperparamètres à tuner

3.1.7 7. SVM (Support Vector Machine)

Principe : Recherche de l'hyperplan optimal séparant les classes avec marge maximale.

Avantages :

— Efficace en haute dimension
— Bonne généralisation
— Contrôle explicite de la complexité

3.1.8 8. Extra Trees (Extremely Randomized Trees)

Principe : Variante de Random Forest avec sélection aléatoire des seuils de coupure.

Avantages :

— Plus rapide que Random Forest
— Réduction de la variance
— Moins sensible aux hyperparamètres

3.1.9 9. Réseau de Neurones (MLP)

Principe : Perceptron multicouches avec fonctions d'activation non linéaires.

3.2 Optimisation des Hyperparamètres

3.2.1 Stratégie de Recherche

TABLE 4 – Stratégies d'optimisation des hyperparamètres

2whitetableheader

tableheaderMéthode	tableheaderDescription	tableheaderUtilisation
Grid Search	Exploration exhaustive de toutes les combinaisons	Modèles simples
Random Search	Échantillonnage aléatoire dans l'espace des paramètres	Modèles complexes
Bayesian Optimization	Optimisation séquentielle basée sur les performances passées	XGBoost, LightGBM
Choix	**Grid Search pour comparaison équitable**	**Tous modèles**

3.2.2 Hyperparamètres Optimisés

TABLE 5 – Détail des hyperparamètres optimisés pour chaque algorithme

tableheaderAlgorithme	tableheaderHyperparamètres	tableheaderValeurs Testées
Régression Logistique	C, penalty, solver	C : [0.01, 0.1, 1, 10, 100]
tableheader Random Forest	n_estimators, max_depth, min_samples_split	n_estimators : [100, 200, 300]
XGBoost	learning_rate, max_depth, n_estimators	learning_rate : [0.01, 0.05, 0.1]
tableheader LightGBM	num_leaves, learning_rate, feature_fraction	num_leaves : [31, 63, 127]
Gradient Boosting	n_estimators, learning_rate, max_depth	n_estimators : [100, 200, 300]
tableheader AdaBoost	n_estimators, learning_rate	n_estimators : [50, 100, 200]
SVM	C, kernel, gamma	C : [0.1, 1, 10], kernel : ['linear', 'r
tableheader Extra Trees	n_estimators, max_features	n_estimators : [100, 200, 300]
MLP	hidden_layer_sizes, activation, alpha	hidden_layer_sizes : [(50,), (100,), (5

3.3 Validation et Évaluation

3.3.1 Stratégie de Validation Croisée

— **K-Fold Stratifié** : 5 folds avec préservation de la distribution des classes
— **Train-Test Split** : 70% entraînement, 30% test
— **Validation Hold-out** : Séparation temporelle simulée

3.3.2 Métriques d'Évaluation

TABLE 6 – Métriques d'évaluation utilisées

2whitetableheader

tableheaderMétrique	tableheaderFormule	tableheaderInterprétation
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Performance globale
Precision	$\frac{TP}{TP+FP}$	Qualité des prédictions positiv
Recall	$\frac{TP}{TP+FN}$	Capacité à détecter les positi
F1-Score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$	Moyenne harmonique
AUC-ROC	$\int_0^1 TPR(FPR)dFPR$	Performance globale à tous seu
Métrique Principale	**F1-Score**	**Équilibre précision/rappe**

4 Résultats Détaillés

4.1 Performances Globales

TABLE 7 – Performances comparées des algorithmes (Validation Croisée 5-Fold)

2whitetableheader

tableheaderAlgorithme	tableheaderF1-Score	tableheaderAccuracy	tableheaderPrecision
accentcolor!20Gradient Boosting	0.665 ± 0.015	0.652 ± 0.012	0.647 ± 0.018
AdaBoost	0.662 ± 0.014	0.648 ± 0.013	0.643 ± 0.017
XGBoost	0.660 ± 0.016	0.645 ± 0.014	0.640 ± 0.019
Random Forest	0.657 ± 0.013	0.642 ± 0.011	0.638 ± 0.016
LightGBM	0.655 ± 0.017	0.640 ± 0.015	0.635 ± 0.020
Extra Trees	0.648 ± 0.014	0.633 ± 0.012	0.629 ± 0.017
SVM	0.642 ± 0.012	0.627 ± 0.010	0.623 ± 0.015
MLP	0.635 ± 0.018	0.620 ± 0.016	0.616 ± 0.021
Régression Logistique	0.630 ± 0.011	0.615 ± 0.009	0.611 ± 0.014

4.2 Importance des Caractéristiques

TABLE 8 – Importance des caractéristiques - Top 15

2whitetableheader

tableheaderRang	tableheaderCaractéristique	tableheaderImportance (%)	tableheaderInterprét
1	Score test technique	33.9	Compétence techni dominant
2	Années d'expérience	32.6	Expérience profess cond facteur
3	Âge	7.8	Maturité, étape de c
4	Score softskills	7.0	Compétences relat émotionnelles
5	Langues parlées	5.2	Compétences linguis national
6	Mobilité	4.7	Flexibilité géographi
7	Disponibilité immédiate	4.5	Réactivité organisat
8	Niveau études (Bac+5)	1.2	Éducation supérieur
9	Spécialité (Informatique)	1.1	Domaine technique s
10	Secteur précédent (Tech)	1.0	Expérience sectoriell
11	Niveau études (Doctorat)	0.8	Éducation avancée
12	Spécialité (Data Science)	0.7	Domaine émergent
13	Secteur précédent (Consulting)	0.5	Expérience conseil
14	Spécialité (Finance)	0.4	Domaine financier
15	Mobilité (Non)	0.2	Limitation géograph

4.3 Analyse des Erreurs

2whitetableheader

tableheaderType Erreur	tableheaderEffectif	tableheaderPourcentage	tableheaderCoût Relatif	
Faux Positifs	85	28.3%	Élevé	Re
Faux Négatifs	72	24.0%	Modéré	
Vrais Positifs	183	61.0%	-	
Vrais Négatifs	160	53.3%	-	
Total Erreurs	157	52.3%	-	

5 Implémentation Technique Avancée

5.1 Pipeline de Production

5.1.1 Code Principal d'Entraînement

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
import pickle
import json

class HRPredictionPipeline:
    def __init__(self, model_path='models/'):
        self.model_path = model_path
        self.scaler = StandardScaler()
        self.model = None
        self.feature_names = None

    def load_and_preprocess(self, filepath):
        """Chargement et pr traitement des donn es"""
        df = pd.read_csv(filepath)

        # Nettoyage des donn es
        df_clean = self._clean_data(df)

        # Encodage des variables cat gorielles
        df_encoded = self._encode_categorical(df_clean)

        # S paration features/target
        X, y = self._prepare_features_target(df_encoded)

        return X, y, df_encoded

    def _clean_data(self, df):
        """  tapes  de nettoyage"""
        # Filtrage  ge
        df = df[(df[' ge '] >= 18) & (df[' ge '] <= 70)]

        # Imputation valeurs manquantes
        numeric_cols = df.select_dtypes(include=np.number).columns
        for col in numeric_cols:
            if df[col].isnull().any():
                df[col].fillna(df[col].median(), inplace=True)

        return df

    def train_model(self, X_train, y_train):
        """Entra nement avec optimisation hyperparam tres"""
        # D finition de la grille de recherche
        param_grid = {
```

```
            'n_estimators': [100, 200, 300],
            'max_depth': [3, 5, 7],
            'learning_rate': [0.01, 0.1, 0.2],
            'subsample': [0.8, 0.9, 1.0]
        }

        # Initialisation mod le
        gb_model = GradientBoostingClassifier(random_state=42)

        # Recherche grille
        grid_search = GridSearchCV(
            gb_model,
            param_grid,
            cv=5,
            scoring='f1',
            n_jobs=-1,
            verbose=1
        )

        # Entra nement
        grid_search.fit(X_train, y_train)

        # R cup ration meilleur mod le
        self.model = grid_search.best_estimator_
        self.best_params = grid_search.best_params_

        return self.model

    def save_model(self, filename='hr_gradient_boosting.pkl'):
        """Sauvegarde du mod le et artefacts"""
        model_data = {
            'model': self.model,
            'scaler': self.scaler,
            'feature_names': self.feature_names,
            'best_params': self.best_params,
            'version': '2.0',
            'date': pd.Timestamp.now().isoformat()
        }

        with open(f'{self.model_path}{filename}', 'wb') as f:
            pickle.dump(model_data, f)

        # Sauvegarde m tadonn es
        metadata = {
            'model_type': 'GradientBoostingClassifier',
            'performance': self.evaluate_model(),
            'features_importance': self.get_feature_importance(),
            'training_date': pd.Timestamp.now().isoformat()
        }

        with open(f'{self.model_path}metadata.json', 'w') as f:
            json.dump(metadata, f, indent=4)
```

Listing 1 – Pipeline complet d'entraînement

5.2 Application Streamlit

```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import plotly.graph_objects as go
from plotly.subplots import make_subplots

class HRPredictionApp:
    def __init__(self):
        self.load_model()
        self.setup_page()

    def setup_page(self):
        st.set_page_config(
            page_title="Syst me  Pr diction  Performance  RH",
            page_icon="    ",
            layout="wide"
        )

        st.title("     Syst me  Intelligent  de  Pr diction  RH")
        st.markdown("""
        Cette  application  utilise  un  mod le  de  Machine  Learning  pour
            pr dire
        la  performance  des  candidats  apr s  6  mois  d'int gration.
        """)

    def load_model(self):
        """Chargement  du  mod le  entra n  """
        try:
            with open('models/hr_gradient_boosting.pkl', 'rb') as f:
                model_data = pickle.load(f)

            self.model = model_data['model']
            self.scaler = model_data['scaler']
            self.feature_names = model_data['feature_names']

        except FileNotFoundError:
            st.error("Mod le  non  trouv .  Veuillez  entra ner  le
                mod le  d'abord.")
            st.stop()

    def display_prediction_dashboard(self):
        """Interface  principale"""
```

```
        tab1, tab2, tab3 = st.tabs([
            "      Pr diction Individuelle",
            "      Pr diction par Lot",
            "      Analyse du Mod le"
        ])

        with tab1:
            self.single_prediction_interface()

        with tab2:
            self.batch_prediction_interface()

        with tab3:
            self.model_analysis_interface()

    def single_prediction_interface(self):
        """Interface pour pr diction unique"""
        col1, col2 = st.columns([2, 1])

        with col1:
            st.header("Saisie des Informations du Candidat")

            # Formulaire interactif
            with st.form("candidate_form"):
                age = st.slider(" ge ", 18, 70, 30)
                experience = st.slider("Ann es d'exp rience", 0, 50,
                    5)
                tech_score = st.slider("Score test technique", 0, 100,
                    70)
                soft_score = st.slider("Score soft skills", 0, 100, 65)

                submitted = st.form_submit_button("Pr dire la
                    Performance")

                if submitted:
                    # Pr paration des donn es
                    candidate_data = self.prepare_candidate_data(
                        age, experience, tech_score, soft_score
                    )

                    # Pr diction
                    prediction, probability = self.predict(
                        candidate_data)

                    # Affichage r sultats
                    with col2:
                        self.display_prediction_results(
                            prediction, probability, candidate_data
                        )
```

Listing 2 – Application Streamlit interactive

5.3 Intégration Continue et Monitoring

TABLE 9 – Système de monitoring et validation

2whitetableheader

tableheaderComposant	tableheaderDescription	tableheaderFréquence
Validation des données	Vérification format et plages des inputs	En temps réel
Dérive des données	Monitoring distribution features vs entraînement	Quotidien
Performance modèle	Tracking métriques sur nouvelles données	Hebdomadaire
Business metrics	Impact sur taux de rétention et performance	Mensuel
Ré-entraînement	Mise à jour modèle avec nouvelles données	Trimestriel

6 Discussion et Analyse Critique

6.1 Interprétation des Résultats

6.1.1 Supériorité du Gradient Boosting

La supériorité du Gradient Boosting Classifier peut s'expliquer par plusieurs facteurs :

- **Capture de relations non linéaires** : Capacité à modéliser des interactions complexes entre variables
- **Robustesse au bruit** : Approche séquentielle réduisant l'impact des outliers
- **Équilibre biais-variance** : Boosting réduit le biais tout en contrôlant la variance
- **Flexibilité** : Support de différentes fonctions de perte et paramètres

6.1.2 Importance des Caractéristiques

L'analyse d'importance révèle plusieurs insights RH importants :

1. **Dominance des compétences techniques** : Le score technique représente 34% de l'importance
2. **Valeur de l'expérience** : Les années d'expérience sont presque aussi importantes
3. **Facteurs soft sous-estimés** : Les softskills ont une importance modérée (7%)
4. **Caractéristiques démographiques** : L'âge a une influence significative

6.2 Limitations et Biais Potentiels

6.2.1 Limitations Techniques

TABLE 10 – Analyse des limitations techniques

2whitetableheader

tableheaderLimitation	tableheaderImpact	tableheaderGravité
Taille échantillon limitée	Risque d'overfitting, généralisation limitée	Élevée
Variables incomplètes	Facteurs importants potentiellement omis	Moyenne
Bruit dans les données	Performance sous-optimale	Faible
Variables corrélées	Instabilité des coefficients	Moyenne
Déséquilibre classes	Biais en faveur de la classe majoritaire	Faible

6.2.2 Biais Éthiques Potentiels

— **Biais d'âge** : Le modèle pourrait discriminer les candidats plus âgés
— **Biais éducatif** : Privilège des diplômes supérieurs
— **Biais sectoriel** : Avantage pour certains secteurs d'activité
— **Biais de mesure** : Les scores tests pourraient être biaisés

6.3 Validation Business

6.3.1 Impact sur les Métriques RH

TABLE 11 – Impact business potentiel du système

2whitetableheader

tableheaderMétrique	tableheaderAvant	tableheaderAprès (Est.)	tableheaderImpact
Taux succès recrutement	65%	75%	+10 points
Coût par recrutement	15k€	12k€	-20%
Délai intégration	6 mois	5 mois	-17%
Rétention 1 an	80%	85%	+5 points
Satisfaction manager	70%	80%	+10 points

7 Conclusion et Perspectives

7.1 Synthèse des Contributions

Ce projet apporte plusieurs contributions significatives :

7.1.1 Contributions Techniques

— Implémentation et comparaison systématique de 9 algorithmes ML
— Développement d'un pipeline de production robuste
— Création d'une application interactive pour utilisateurs finaux
— Méthodologie de validation rigoureuse

7.1.2 Contributions RH

— Identification des facteurs clés de performance
— Outil d'aide à la décision objectif
— Réduction des biais de recrutement
— Optimisation des processus de sélection

7.2 Recommandations Stratégiques

7.2.1 À Court Terme (0-6 mois)

1. Déployer le système en phase pilote sur un département
2. Former les recruteurs à l'interprétation des résultats
3. Établir un processus de feedback pour amélioration continue
4. Monitorer les impacts sur la diversité et l'inclusion

7.2.2 À Moyen Terme (6-18 mois)

1. Intégrer des sources de données supplémentaires
2. Développer des modèles spécifiques par métier
3. Implémenter un système de recommendation de formations
4. Étendre à la prédiction de la rétention

7.2.3 À Long Terme (18+ mois)

1. Créer un écosystème complet de talent analytics
2. Intégrer l'IA explicative pour la transparence
3. Développer des modèles prédictifs pour le développement de carrière
4. Contribuer à la recherche académique en HR analytics

7.3 Directions Futures de Recherche

— **Apprentissage par transfert** : Adaptation de modèles entre organisations
— **Modèles temporels** : Prédiction de l'évolution de la performance
— **Analyse de réseaux** : Influence des relations interpersonnelles
— **NLP avancé** : Analyse sémantique des CV et entretiens
— **Apprentissage par renforcement** : Optimisation dynamique des processus

7.4 Déclaration Éthique

Ce système doit être utilisé selon les principes suivants :

— **Transparence** : Les candidats doivent être informés de l'utilisation d'IA
— **Contrôle humain** : Les prédictions doivent rester des recommandations
— **Équité** : Surveillance continue des biais potentiels
— **Vie privée** : Protection des données personnelles
— **Responsabilité** : Désignation claire de la responsabilité décisionnelle

Remerciements

Je tiens à exprimer ma gratitude aux experts RH qui ont contribué à la validation des hypothèses de ce projet, ainsi qu'à la communauté open-source pour les outils et bibliothèques qui ont rendu ce travail possible.

Déclaration d'Intérêts

L'auteur déclare n'avoir aucun conflit d'intérêts financier ou personnel qui pourrait influencer les résultats présentés dans ce rapport.

8 Annexes Techniques

8.1 Code Complet de Prétraitement

```
"""
Script de pr traitement complet pour les donn es RH
Auteur: Taha Bchir
Date: \today
"""

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import KNNImputer
import warnings
warnings.filterwarnings('ignore')

class HRDataPreprocessor:
    """Classe pour le pr traitement des donn es RH"""

    def __init__(self, config=None):
        self.config = config or self.default_config()
        self.scaler = StandardScaler()
        self.imputer = KNNImputer(n_neighbors=5)
        self.encoder = OneHotEncoder(drop='first', sparse=False)
        self.feature_names = None

    @staticmethod
    def default_config():
        return {
            'age_range': (18, 70),
            'score_range': (0, 100),
            'imputation_strategy': 'median',
            'encoding_method': 'onehot',
            'scaling_method': 'standard'
        }

    def full_pipeline(self, df):
        """Ex cute le pipeline complet"""

        # 1. Nettoyage initial
        df_clean = self.clean_data(df)

        # 2. Traitement valeurs manquantes
        df_imputed = self.handle_missing_values(df_clean)

        # 3. Encodage variables cat gorielles
        df_encoded = self.encode_categorical(df_imputed)

        # 4. Scaling des variables num riques
```

```
        df_scaled = self.scale_features(df_encoded)

        # 5. Feature engineering
        df_engineered = self.feature_engineering(df_scaled)

        # 6. S lection features
        df_final = self.select_features(df_engineered)

        return df_final

    def clean_data(self, df):
        """ tapes  de nettoyage"""

        # Suppression des doublons
        df = df.drop_duplicates()

        # Filtrage des  ges  invalides
        min_age, max_age = self.config['age_range']
        df = df[(df[' ge '] >= min_age) & (df[' ge '] <= max_age)]

        # Correction des scores hors limites
        min_score, max_score = self.config['score_range']
        df['score_test_technique'] = df['score_test_technique'].clip(
            min_score, max_score)
        df['score_softskills'] = df['score_softskills'].clip(min_score,
            max_score)

        # Standardisation du texte
        text_cols = ['niveau_ tudes', 'sp cialit ', '
            secteur_pr c dent']
        for col in text_cols:
            if col in df.columns:
                df[col] = df[col].str.lower().str.strip()

        return df

    def handle_missing_values(self, df):
        """Gestion des valeurs manquantes"""

        # Strat gies par type de variable
        strategies = {
            'numeric': self.config['imputation_strategy'],
            'categorical': 'mode',
            'binary': 'mode'
        }

        for col in df.columns:
            if df[col].isnull().any():
                col_type = self.get_column_type(df[col])
                strategy = strategies[col_type]

                if strategy == 'median':
```

```
                    df[col].fillna(df[col].median(), inplace=True)
                elif strategy == 'mean':
                    df[col].fillna(df[col].mean(), inplace=True)
                elif strategy == 'mode':
                    df[col].fillna(df[col].mode()[0], inplace=True)
                elif strategy == 'knn':
                    # Imputation KNN pour variables corr l es
                    pass

        return df

    def get_column_type(self, series):
        """D termine le type de colonne"""
        if series.dtype in ['int64', 'float64']:
            if series.nunique() == 2:
                return 'binary'
            else:
                return 'numeric'
        else:
            return 'categorical'
```

Listing 3 – Script complet de prétraitement des données

8.2 Configuration d'Environnement

```
# Core Data Science
numpy==1.21.0
pandas==1.3.0
scikit-learn==0.24.2
scipy==1.7.0

# Machine Learning Algorithms
xgboost==1.4.2
lightgbm==3.2.1
catboost==0.26.1

# Visualization
matplotlib==3.4.2
seaborn==0.11.1
plotly==5.3.1

# Web Application
streamlit==1.8.1
flask==2.0.1
dash==2.0.0

# Utilities
joblib==1.0.1
imbalanced-learn==0.8.0
```

```
optuna==2.10.0
shap==0.39.0

# Documentation
jupyter==1.0.0
notebook==6.4.0
pytest==6.2.5
```

Listing 4 – Fichier requirements.txt

8.3 Documentation de l'API

```
"""
API Documentation - HR Prediction System
Base URL: https://api.hr-predict.com/v1
"""

# Endpoints disponibles
ENDPOINTS = {
    'single_prediction': '/predict',
    'batch_prediction': '/predict/batch',
    'model_info': '/model/info',
    'model_retrain': '/model/retrain',
    'monitoring': '/monitoring/metrics'
}

# Exemple de requ te
import requests
import json

def predict_candidate(candidate_data):
    """Pr diction pour un candidat unique"""

    url = "https://api.hr-predict.com/v1/predict"
    headers = {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer YOUR_API_KEY'
    }

    response = requests.post(
        url,
        headers=headers,
        data=json.dumps(candidate_data)
    )

    if response.status_code == 200:
        return response.json()
    else:
        raise Exception(f"API Error: {response.status_code}")
```

```
38
39 # R ponse type
40 SAMPLE_RESPONSE = {
41     "success": True,
42     "prediction": 1,
43     "probability": 0.78,
44     "confidence_interval": [0.72, 0.84],
45     "key_factors": [
46         {"feature": "score_test_technique", "importance": 0.34},
47         {"feature": "annees_experience", "importance": 0.27}
48     ],
49     "model_version": "2.0",
50     "timestamp": "2024-01-15T10:30:00Z"
51 }
```

Listing 5 – Documentation API REST

8.4 Plan de Déploiement

TABLE 12 – Plan de déploiement en production

2whitetableheader

tableheaderPhase	tableheaderDurée	tableheaderResponsable	tableheaderActivités
Pré-déploiement	2 semaines	Équipe Data	Tests de charge, validation données
Déploiement pilote	1 mois	Équipe RH	Formation utilisateurs, collecte feedback
Déploiement complet	3 mois	Équipe IT	Intégration systèmes existants
Monitoring	Continue	Équipe Ops	Surveillance performance, maintenance
Optimisation	Continue	Équipe Data	Amélioration continue modèle