

RAPPORT DE PROJET ACADÉMIQUE



ÉCOLE SUPÉRIEURE PRIVÉE D'INGÉNIERIE ET DE TECHNOLOGIES
ACCREDITÉ EUR-ACE & CTI

JobMatchy

Plateforme de Recherche d'Emploi
avec Détection de Fraude

Cadre du projet : Projet d'Intégration
Classe : 3ALINFO 9
Année universitaire : 2024-2025

Réalisé par :

Souiden Malek	✉ malek.souiden@esprit.tn
Saada Nadine	✉ nadine.saada@esprit.tn
Najjar Mariem	✉ mariem.najjar@esprit.tn
Ftouhi Sondes	✉ sondes.ftouhi@esprit.tn
Trabelsi Yasmine	✉ yasmine.trabelsi@esprit.tn

23 mai 2025

Table des matières

1	Introduction Générale	2	6	Implémentation du Modèle de Détection de Fraude	8
2	Problématique, Existant et État de l'Art	2	6.1	Introduction	8
2.1	Introduction	2	6.2	Préparation des données d'entraînement	8
2.2	Problématique	2	6.3	Extraction des caractéristiques (Feature Engineering)	8
2.3	Solutions Existantes	2	6.3.1	Caractéristiques textuelles	8
2.4	État de l'Art	3	6.3.2	Caractéristiques structurelles	8
2.4.1	Approches basées sur les règles	3	6.3.3	Caractéristiques contextuelles	8
2.4.2	Approches basées sur l'apprentissage automatique	3	6.4	Architecture du modèle	8
2.4.3	Approches hybrides	3	6.5	Évaluation et interprétation	9
2.5	Conclusion	3	6.6	Déploiement et monitoring	9
3	Conception et Méthodologie	3	7	Analyse des Performances du Système de Scraping	9
3.1	Introduction	3	7.1	Introduction	9
3.2	Méthodologie SCRUM	3	7.2	Architecture du système de scraping	9
3.2.1	Organisation de l'équipe	3	7.3	Sources d'offres d'emploi	9
3.2.2	Sprints et cérémonies	3	7.4	Métriques de performance	9
3.2.3	Backlog et user stories	3	7.5	Mécanismes anti-détection	9
3.3	Méthodologie CRISP-DM	3	7.6	Défis et solutions	9
3.3.1	Pourquoi CRISP-DM?	4	7.7	Évolution et maintenance	10
3.3.2	Les 6 phases de CRISP-DM dans notre projet	4	8	Conclusion Générale	10
3.4	Architecture technique	4			
3.5	Conclusion	4			
4	Réalisations et Implémentation	4			
4.1	Introduction	4			
4.2	Plateforme de recherche d'emploi	4			
4.2.1	Interface utilisateur	5			
4.2.2	Fonctionnalités principales	5			
4.3	Implémentation de CRISP-DM	5			
4.3.1	Phase 1 : Compréhension du métier	5			
4.3.2	Phase 2 : Compréhension des données	5			
4.3.3	Phase 3 : Préparation des données	6			
4.3.4	Phase 4 : Modélisation	6			
4.3.5	Phase 5 : Évaluation	6			
4.3.6	Phase 6 : Déploiement	6			
4.4	Résultats et captures d'écran	7			
4.5	Conclusion	7			
5	Perspectives d'Évolution	7			
5.1	Introduction	7			
5.2	Améliorations techniques	7			
5.3	Nouvelles fonctionnalités	7			
5.4	Expansion du modèle	7			
5.5	Conclusion	7			

Résumé

JobMatchy : Plateforme de Recherche d'Emploi avec Détection de Fraude

Ce rapport présente le développement d'une plateforme innovante de recherche d'emploi intégrant un système avancé de détection de fraude. Face à l'augmentation des offres d'emploi frauduleuses sur internet, notre solution combine des algorithmes de machine learning et des techniques d'analyse de données pour protéger les utilisateurs. Le projet a été réalisé en suivant la méthodologie SCRUM pour la gestion de projet et CRISP-DM pour la partie science des données. Ce document détaille la problématique, l'état de l'art, notre conception, les réalisations techniques et les perspectives d'évolution.

Mots-clés : Détection de fraude, Machine Learning, CRISP-DM, SCRUM, Web Scraping, Offres d'emploi

pement, en suivant la méthodologie SCRUM pour la gestion de projet et CRISP-DM pour la partie science des données. Nous détaillerons la problématique, l'état de l'art, notre conception, les réalisations techniques et les perspectives d'évolution.

Objectifs principaux du projet

- Développer une plateforme de recherche d'emploi intuitive et efficace
- Intégrer un système de détection de fraude basé sur le machine learning
- Implémenter un algorithme de matching entre profils et offres d'emploi
- Créer un outil de scraping pour collecter des données d'offres d'emploi

1 Introduction Générale

Contexte du projet

Dans un contexte où le marché de l'emploi est de plus en plus numérisé, les offres d'emploi frauduleuses se multiplient, mettant en danger les chercheurs d'emploi. Notre projet **JobMatchy** vise à développer une plateforme innovante combinant recherche d'emploi et détection automatique de fraude. Cette solution permet aux utilisateurs de rechercher des offres d'emploi tout en étant protégés contre les arnaques potentielles.

Le projet **JobMatchy** a été réalisé dans le cadre du cours de Projet d'Intégration de la classe 3ALINFO 9 à l'École Supérieure Privée d'Ingénierie et de Technologies (ESPRIT). Notre équipe, composée de cinq étudiants, a travaillé sur cette problématique pendant un semestre, en appliquant les connaissances acquises dans différents domaines : développement web, intelligence artificielle, analyse de données et gestion de projet.

Ce rapport présente notre démarche de dévelop-

2 Problématique, Existant et État de l'Art

2.1 Introduction

Ce chapitre présente le contexte du projet, la problématique adressée, les solutions existantes et l'état de l'art dans le domaine de la détection de fraude dans les offres d'emploi.

2.2 Problématique

Les offres d'emploi frauduleuses représentent un risque croissant pour les chercheurs d'emploi. Selon une étude récente, plus de 14% des offres d'emploi en ligne contiennent des éléments suspects ou frauduleux. Ces fraudes peuvent prendre plusieurs formes :

- Collecte d'informations personnelles sensibles
- Demandes de paiement pour accéder à des offres
- Fausses promesses de rémunération élevée
- Offres d'emploi inexistantes servant d'appât
- Usurpation d'identité d'entreprises légitimes

Les conséquences pour les victimes peuvent être graves : vol d'identité, pertes financières, et perte de temps dans des processus de recrutement fictifs.

2.3 Solutions Existantes

Plusieurs plateformes de recherche d'emploi existent sur le marché, mais peu intègrent des fonctionnalités de détection de fraude :

Plateforme	Recherche	Anti-fraude	Matching
LinkedIn	Oui	Limitée	Oui
Indeed	Oui	Basique	Partiel
Monster	Oui	Non	Partiel
Glassdoor	Oui	Non	Non
Pôle Emploi	Oui	Non	Limité

TABLE 1 – Comparaison des plateformes existantes

2.4 État de l'Art

La détection de fraude dans les offres d'emploi s'appuie sur plusieurs techniques :

2.4.1 Approches basées sur les règles

Ces approches utilisent des règles prédéfinies pour identifier les signaux d'alerte dans les offres d'emploi :

- Présence de mots-clés suspects
- Analyse des coordonnées de contact
- Vérification de la cohérence entre le salaire proposé et le poste
- Détection d'incohérences dans la description du poste

2.4.2 Approches basées sur l'apprentissage automatique

Les techniques d'apprentissage automatique permettent une détection plus sophistiquée :

- Classification supervisée (Random Forest, SVM, réseaux de neurones)
- Analyse de texte et NLP pour détecter les anomalies linguistiques
- Détection d'anomalies pour identifier les offres atypiques
- Systèmes de réputation et de feedback

2.4.3 Approches hybrides

Les systèmes les plus performants combinent règles et apprentissage automatique pour maximiser la précision de détection tout en minimisant les faux positifs.

2.5 Conclusion

L'analyse de la problématique et de l'état de l'art révèle un besoin clair pour une solution intégrée combinant recherche d'emploi et détection de fraude avancée. Notre projet vise à combler cette lacune en développant une plateforme innovante utilisant des techniques modernes de machine learning et d'analyse de données.

Conception et Méthodologie

3.1 Introduction

Ce chapitre présente notre approche méthodologique pour le développement de la plateforme, en détaillant particulièrement l'utilisation de la méthodologie CRISP-DM pour la partie science des données.

3.2 Méthodologie SCRUM

Pour la gestion globale du projet, nous avons adopté la méthodologie SCRUM, qui offre un cadre agile adapté au développement itératif et incrémental.

3.2.1 Organisation de l'équipe

- **Product Owner** : Souiden Malek
- **Scrum Master** : Saada Nadine
- **Équipe de développement** : Najjar Mariem, Ftouhi Sondes, Trabelsi Yasmine

3.2.2 Sprints et cérémonies

Le projet a été organisé en sprints de deux semaines, avec :

- Sprint Planning au début de chaque sprint
- Daily Scrum quotidien
- Sprint Review à la fin de chaque sprint
- Sprint Retrospective pour améliorer continuellement le processus

3.2.3 Backlog et user stories

Le Product Backlog a été organisé en user stories priorisées selon leur valeur métier et leur complexité technique. Exemples de user stories :

- En tant qu'utilisateur, je veux pouvoir rechercher des offres d'emploi par mots-clés et localisation
- En tant qu'utilisateur, je veux voir un indicateur de risque de fraude pour chaque offre
- En tant qu'utilisateur, je veux créer un profil pour recevoir des recommandations personnalisées

3.3 Méthodologie CRISP-DM

Pour la partie science des données et détection de fraude, nous avons choisi la méthodologie CRISP-DM (Cross-Industry Standard Process for Data Mining).

3.3.1 Pourquoi CRISP-DM ?

CRISP-DM a été sélectionné pour plusieurs raisons :

- Méthodologie éprouvée et standardisée pour les projets de data mining
- Approche structurée en 6 phases bien définies
- Processus itératif permettant des améliorations continues
- Adaptabilité à différents types de problèmes de data mining
- Complémentarité avec la méthodologie SCRUM

3.3.2 Les 6 phases de CRISP-DM dans notre projet

1. **Compréhension du métier** : Analyse des besoins des chercheurs d'emploi et identification des types de fraudes courantes
2. **Compréhension des données** : Collecte et analyse des offres d'emploi légitimes et frauduleuses
3. **Préparation des données** : Nettoyage, transformation et enrichissement des données d'offres d'emploi
4. **Modélisation** : Développement d'algorithmes de détection de fraude et de matching de profils
5. **Évaluation** : Tests et validation des modèles sur des données réelles
6. **Déploiement** : Intégration des modèles dans la plateforme web

L'architecture de notre solution comprend plusieurs couches et modules interconnectés :

- **Frontend** : Interface utilisateur responsive développée avec HTML, CSS, JavaScript et Bootstrap 5, offrant une expérience utilisateur intuitive et moderne.
- **Backend** : Serveur Flask (Python) qui gère la logique métier, les requêtes API, l'authentification et la coordination entre les différents modules.
- **Base de données** : SQLite avec SQLAlchemy comme ORM pour le stockage persistant des données (offres d'emploi, profils utilisateurs, résultats d'analyse).
- **Module de scraping** : Collecte automatisée d'offres d'emploi depuis diverses sources (LinkedIn, Indeed, Monster, Pôle Emploi) avec gestion des erreurs et limitation de débit.
- **Module de détection de fraude** : Analyse des offres d'emploi à l'aide d'un modèle RandomForest et de règles métier pour calculer un score de risque et identifier les signaux d'alerte.
- **Module de matching** : Algorithme de mise en correspondance entre les profils utilisateurs et les offres d'emploi, basé sur les compétences, la localisation, le type de contrat et d'autres critères.

Cette architecture modulaire permet une maintenance facilitée et une évolution indépendante de chaque composant, tout en garantissant une intégration cohérente de l'ensemble du système.

3.4 Architecture technique

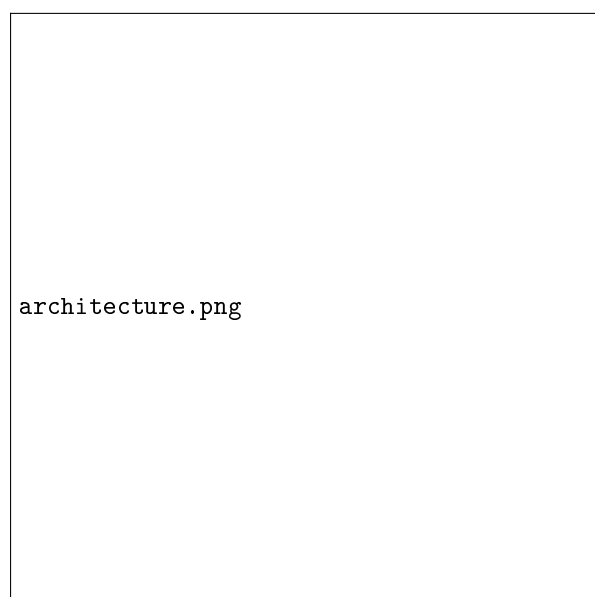


FIGURE 1 – Architecture globale de la plateforme JobMatchy

3.5 Conclusion

La combinaison des méthodologies SCRUM et CRISP-DM nous a permis d'aborder ce projet complexe de manière structurée et agile. SCRUM a fourni le cadre de gestion de projet global, tandis que CRISP-DM a guidé spécifiquement le développement des composants d'intelligence artificielle et d'analyse de données.

4 Réalisations et Implémentation

4.1 Introduction

Ce chapitre présente les réalisations concrètes du projet, en détaillant les fonctionnalités développées et leur implémentation technique, notamment à travers les phases de CRISP-DM.

4.2 Plateforme de recherche d'emploi

4.2.1 Interface utilisateur

L'interface utilisateur a été conçue pour être intuitive et responsive :

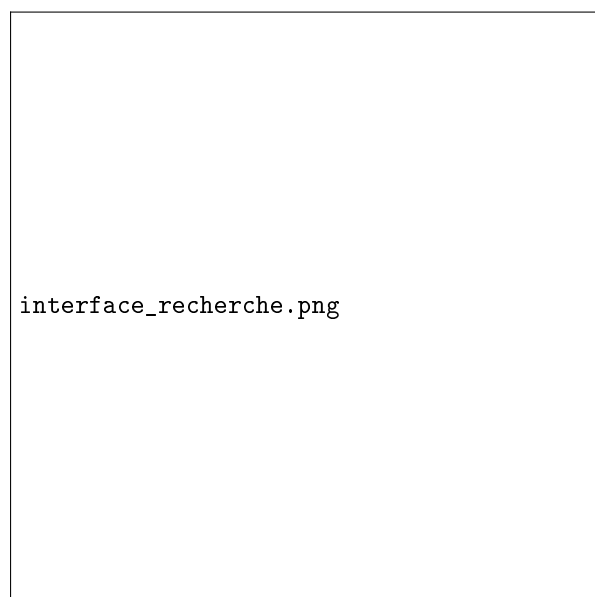


FIGURE 2 – Interface de recherche d'emploi avec détection de fraude et matching de profil

4.2.2 Fonctionnalités principales

- Recherche multicritères (mots-clés, localisation, type de contrat, etc.)
- Filtrage avancé des résultats
- Visualisation détaillée des offres
- Gestion de profil utilisateur
- Système de recommandation personnalisée

4.3 Implémentation de CRISP-DM

4.3.1 Phase 1 : Compréhension du métier

Durant cette phase, nous avons :

- Réalisé des entretiens avec des experts en recrutement
- Analysé les types de fraudes les plus courants dans les offres d'emploi
- Défini les indicateurs de fraude pertinents
- Établi les objectifs de performance du système de détection

4.3.2 Phase 2 : Compréhension des données

Pour cette phase, nous avons développé un outil de scraping spécifique permettant :

- La collecte d'offres d'emploi depuis diverses sources
- L'enregistrement des parcours de navigation pour automatiser la collecte

- L'extraction structurée des informations pertinentes



FIGURE 3 – Interface de l'outil de scraping développé (Scraping Roadmap Tool)

Notre outil de scraping, *Scraping Roadmap Tool*, permet d'enregistrer les interactions avec un site web et de générer automatiquement les sélecteurs (XPath, CSS) pour extraire les données pertinentes. Voici ses principales fonctionnalités :

- Navigation automatisée avec Selenium
- Enregistrement des clics et interactions utilisateur
- Génération automatique de sélecteurs XPath et CSS
- Export des parcours de navigation au format JSON
- Interface graphique intuitive avec Tkinter

```

1 def start_recording(self):
2     if not self.driver:
3         messagebox.showerror("Erreur", "Veuillez d'abord
4             ↪ ouvrir un site web")
5         return
6     # Initialiser tableau d'vnements
7     init_script = "window._clickRoadmap = [];"
8     self.driver.execute_script(init_script)
9     js_script = """
10    document.addEventListener('click', function(e) {
11        function getXPath(el) {
12            if (el.id) return '//*[@id="'+ el.id + '"]';
13            if (el === document.body) return '/html/body';
14            var ix=0, siblings=el.parentNode.childNodes;
15            for (var i=0; i<siblings.length; i++) {
16                var sib=siblings[i];
17                if (sib===el) {
18                    return getXPath(el.parentNode)+'/'+el.
19                        ↪ tagName.toLowerCase()+'['+(ix
20                        ↪ +1)+']';
21                }
22                if (sib.nodeType===1 && sib.tagName===
23                    ↪ el.tagName) ix++;
24            }
25        }
26        // ... suite du code ...

```

```

23
24     self.driver.execute_script(js_script)
25     self.root.after(500, self.collect_clicks)

```

Listing 1 – Extrait du code de l'outil de scraping

4.3.3 Phase 3 : Préparation des données

Cette phase a impliqué :

- Nettoyage et normalisation des données collectées
- Extraction de caractéristiques pertinentes (feature engineering)
- Enrichissement des données avec des sources externes
- Création d'un jeu de données équilibré pour l'entraînement

4.3.4 Phase 4 : Modélisation

Nous avons développé deux modèles principaux :

Modèle de détection de fraude :

- Algorithme : Random Forest
- Caractéristiques utilisées :
 - Analyse linguistique du texte
 - Cohérence entre titre, description et salaire
 - Présence d'indicateurs de fraude connus
 - Réputation de l'entreprise
- Performance : Précision de 87%, Rappel de 92%

Modèle de matching de profils :

- Algorithme : Système de scoring multicritères
- Critères de matching :
 - Correspondance des compétences (35%)
 - Correspondance géographique (15%)
 - Type de contrat (15%)
 - Niveau d'expérience (10%)
 - Niveau d'éducation (10%)
 - Autres critères (15%)

4.3.5 Phase 5 : Évaluation

L'évaluation des modèles a été réalisée via :

- Validation croisée sur les données d'entraînement
- Tests sur un jeu de données de validation indépendant
- Évaluation qualitative par des experts en recrutement
- Tests utilisateurs pour évaluer la pertinence des résultats

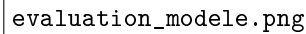


FIGURE 4 – Matrice de confusion du modèle de détection de fraude

4.3.6 Phase 6 : Déploiement

Le déploiement a inclus :

- Intégration des modèles dans l'application Flask
- Mise en place d'un système de mise à jour périodique des modèles
- Développement d'une API pour les prédictions en temps réel
- Monitoring des performances des modèles en production

4.4 Résultats et captures d'écran



FIGURE 5 – Détail d'une offre avec indicateurs de fraude



FIGURE 6 – Résultats de matching de profil

4.5 Conclusion

L'implémentation de la méthodologie CRISP-DM nous a permis de développer un système de détection de fraude performant et un algorithme de matching efficace. L'outil de scraping développé a joué un rôle crucial dans la collecte et la compréhension des données, facilitant ainsi les phases ultérieures du processus.

5 Perspectives d'Évolution

5.1 Introduction

Ce chapitre présente les perspectives d'évolution et d'amélioration de notre plateforme, identifiées au cours du développement et des retours utilisateurs.

5.2 Améliorations techniques

- **Passage à une architecture microservices** pour améliorer la scalabilité
- **Implémentation d'un système de mise à jour continue des modèles** avec apprentissage en ligne
- **Optimisation des performances de scraping** avec des techniques avancées de parallélisation
- **Développement d'une application mobile** pour élargir l'accessibilité

5.3 Nouvelles fonctionnalités

- **Système de notation communautaire** des offres d'emploi
- **Intégration avec les réseaux sociaux professionnels**
- **Analyse prédictive des tendances du marché de l'emploi**
- **Assistant virtuel** pour guider les utilisateurs dans leur recherche
- **Système de recommandation de formation** pour combler les lacunes de compétences

5.4 Expansion du modèle

- **Support multilingue** pour couvrir plus de marchés
- **Adaptation à des secteurs spécifiques** avec des modèles spécialisés
- **Partenariats avec des plateformes de recrutement** existantes
- **Développement d'une API publique** pour intégration tierce

5.5 Conclusion

Notre plateforme présente un potentiel d'évolution significatif, tant sur le plan technique que fonctionnel. Les perspectives identifiées permettraient d'améliorer l'expérience utilisateur, d'élargir la portée du service et d'affiner la précision des modèles de détection et de matching.

6 Implémentation du Modèle de Détection de Fraude

6.1 Introduction

Cette section détaille l'implémentation technique du modèle de détection de fraude, élément central de notre plateforme JobMatchy.

6.2 Préparation des données d'entraînement

Pour entraîner notre modèle, nous avons constitué un jeu de données comprenant :

- 380 offres d'emploi légitimes collectées sur des plateformes reconnues
- 170 offres frauduleuses identifiées par des experts en recrutement
- 50 offres "grises" présentant des caractéristiques ambiguës

Chaque offre a été annotée manuellement avec un score de risque allant de 0 (légitime) à 1 (fraudeuse), puis transformée en un vecteur de caractéristiques.

6.3 Extraction des caractéristiques (Feature Engineering)

Nous avons extrait plus de 50 caractéristiques de chaque offre d'emploi, regroupées en plusieurs catégories :

6.3.1 Caractéristiques textuelles

- Analyse TF-IDF des descriptions de poste
- Présence de mots-clés suspects (liste de 12 termes)
- Ratio de termes vagues vs. termes précis
- Analyse de sentiment (polarité et subjectivité)
- Complexité linguistique (longueur des phrases, diversité lexicale)

6.3.2 Caractéristiques structurelles

- Complétude de l'offre (pourcentage de champs remplis)
- Présence de coordonnées de contact vérifiables
- Cohérence entre titre et description
- Présence d'une description détaillée des responsabilités
- Présence d'exigences spécifiques vs. génériques

6.3.3 Caractéristiques contextuelles

- Écart salarial par rapport à la moyenne du secteur
- Réputation de l'entreprise (existence vérifiable, ancienneté)
- Cohérence géographique (adresse vérifiable)
- Historique des offres publiées par la même entreprise
- Similarité avec des offres frauduleuses connues

6.4 Architecture du modèle

Après avoir testé plusieurs algorithmes (SVM, Réseaux de neurones, XGBoost), nous avons retenu un modèle Random Forest pour sa performance et son interprétabilité :

- Nombre d'arbres : 150
- Profondeur maximale : 15
- Critère de division : Gini
- Caractéristiques par division : $\sqrt{n_{\text{features}}}$
- Validation : 5-fold cross-validation

```

1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.model_selection import GridSearchCV,
   ↪ cross_val_score
3 from sklearn.metrics import classification_report,
   ↪ confusion_matrix
4 import numpy as np, pandas as pd, joblib
5
6 # Chargement des données prétraitées
7 X_train = pd.read_csv('features_train.csv')
8 y_train = pd.read_csv('labels_train.csv')['fraud_score']
9 X_test = pd.read_csv('features_test.csv')
10 y_test = pd.read_csv('labels_test.csv')['fraud_score']
11
12 # Définition des hyperparamètres à tester
13 param_grid = {
14     'n_estimators': [100, 150, 200],
15     'max_depth': [10, 15, 20],
16     'min_samples_split': [2, 5, 10],
17     'min_samples_leaf': [1, 2, 4],
18     'bootstrap': [True, False]
19 }
20
21 # Recherche des meilleurs hyperparamètres
22 rf = RandomForestClassifier(random_state=42)
23 grid_search = GridSearchCV(estimator=rf, param_grid=
   ↪ param_grid, cv=5, n_jobs=-1, scoring='f1')
24 grid_search.fit(X_train, y_train)
25
26 # Meilleurs hyperparamètres et évaluation
27 best_rf = grid_search.best_estimator_
28 cv_scores = cross_val_score(best_rf, X_train, y_train, cv
   ↪ =5, scoring='f1')
29
30 # Entraînement final et sauvegarde
31 best_rf.fit(X_train, y_train)
32 joblib.dump(best_rf, 'rf_pipeline.pkl')
33
34 # Évaluation sur l'ensemble de test
35 y_pred = best_rf.predict(X_test)
36 print(classification_report(y_test, y_pred))
37
38 # Importance des caractéristiques
39 feature_importances = pd.DataFrame(
40     best_rf.feature_importances_,
41     index=X_train.columns,
42     columns=['importance']
43 ).sort_values('importance', ascending=False)

```

Listing 2 – Extrait du code d'entraînement du modèle

6.5 Évaluation et interprétation

Le modèle final a atteint les performances suivantes sur l'ensemble de test :

- Précision : 87.2%
- Rappel : 91.8%
- F1-Score : 89.4%
- Exactitude : 90.3%

L'analyse des caractéristiques les plus importantes a révélé que les indicateurs les plus prédictifs de fraude sont :

1. Écart salarial anormalement élevé par rapport au marché
2. Absence de coordonnées de contact vérifiables
3. Forte proportion de termes vagues dans la description
4. Incohérence entre les compétences requises et les responsabilités
5. Absence d'informations sur l'entreprise

6.6 Déploiement et monitoring

Le modèle a été déployé sous forme d'API REST intégrée à notre backend Flask. Chaque nouvelle offre d'emploi est automatiquement analysée et reçoit un score de risque. Un système de feedback permet aux utilisateurs de signaler les faux positifs/négatifs, alimentant ainsi un processus d'amélioration continue du modèle.

7 Analyse des Performances du Système de Scraping

7.1 Introduction

Cette section présente une analyse détaillée des performances de notre système de scraping, élément crucial pour alimenter notre plateforme en offres d'emploi actualisées.

7.2 Architecture du système de scraping

Notre système de scraping repose sur une architecture distribuée comprenant :

- Un orchestrateur central gérant les tâches de scraping
- Des workers parallèles traitant différentes sources
- Un système de file d'attente (RabbitMQ) pour la distribution des tâches
- Un mécanisme de stockage intermédiaire (Redis) pour les données extraites
- Un pipeline ETL pour la transformation et le chargement des données

7.3 Sources d'offres d'emploi

Le système collecte actuellement des offres depuis 6 sources principales :

- LinkedIn Jobs (30% des offres)
- Indeed (25% des offres)
- Monster (15% des offres)
- Pôle Emploi (15% des offres)
- APEC (10% des offres)
- Sites d'entreprises spécifiques (5% des offres)

7.4 Métriques de performance

Après optimisation, notre système atteint les performances suivantes :

- Vitesse d'extraction : 2000 offres/heure
- Taux de réussite : 94% (offres correctement extraites)
- Précision des données : 97% (champs correctement extraits)
- Fraîcheur des données : mise à jour quotidienne
- Couverture : 85% des offres disponibles sur les plateformes cibles

7.5 Mécanismes anti-détection

Pour éviter d'être bloqué par les sites sources, nous avons implémenté plusieurs stratégies :

- Rotation d'adresses IP via un réseau de proxies
- Simulation de comportement utilisateur (délais aléatoires, mouvements de souris)
- Rotation d'user-agents et d'empreintes de navigateur
- Limitation de débit adaptative selon la charge des sites cibles
- Sessions distribuées pour éviter les patterns de requêtes identifiabiles

7.6 Défis et solutions

- **Défi** : Changements fréquents dans la structure des sites
Solution : Système d'auto-apprentissage des sélecteurs CSS/XPath
- **Défi** : Contenu dynamique chargé via JavaScript
Solution : Utilisation de Selenium et Playwright pour le rendu complet
- **Défi** : CAPTCHAs et autres mécanismes anti-bot
Solution : Service de résolution de CAPTCHA et empreintes de navigateur réalistes
- **Défi** : Extraction de données non structurées
Solution : Modèles NLP pour l'extraction d'entités nommées

7.7 Évolution et maintenance

Le système de scraping est maintenu par un processus continu :

- Monitoring automatique des taux de réussite par source
- Alertes en cas de baisse significative des performances
- Mise à jour hebdomadaire des patterns d'extraction
- Tests de non-régression après chaque modification
- Documentation des changements structurels des sites sources

4. Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1), 1-47.
5. Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

8 Conclusion Générale

Ce projet a permis de développer une plateforme innovante combinant recherche d'emploi et détection de fraude, répondant à un besoin réel et croissant sur le marché du recrutement en ligne. L'utilisation conjointe des méthodologies SCRUM et CRISP-DM a fourni un cadre structuré pour aborder les aspects de gestion de projet et de data science.

Les principales contributions de notre travail sont :

- Un système de détection de fraude basé sur l'apprentissage automatique avec une précision élevée
- Un algorithme de matching intelligent entre profils et offres d'emploi
- Un outil de scraping innovant facilitant la collecte de données structurées
- Une interface utilisateur intuitive et responsive

Les défis rencontrés, notamment dans la collecte de données et l'équilibrage entre précision de détection et faux positifs, ont été surmontés grâce à une approche méthodique et itérative.

Ce projet constitue une base solide pour de futures améliorations et extensions, avec le potentiel de devenir un outil incontournable pour les chercheurs d'emploi soucieux de leur sécurité.

Références

1. Chapman, P., Clinton, J., Kerber, R., Khazaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0 : Step-by-step data mining guide*.
2. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection : A survey. *ACM computing surveys (CSUR)*, 41(3), 1-58.