ActuarIA SOFTWARE
Vehicle Insurance Fraud Detection

BARKATI Soufian
JBIHA Ghita
KHALOUI Ghita

10 décembre 2023

Table des matières

1 Introduction

La fraude à l'assurance, en particulier dans le domaine de l'assurance automobile, reste un défi majeur pour le secteur de l'assurance. Cette activité frauduleuse englobe un spectre de pratiques trompeuses, allant des demandes d'indemnisation falsifiées ou exagérées liées à des dommages matériels aux accidents manigancés et aux déclarations de blessures fictives. L'exploitation des assurances via des accidents truqués, des passagers imaginaires et des demandes d'indemnisation gonflées a représenté une charge financière considérable pour les compagnies d'assurance, affectant leur crédibilité et, inévitablement, les primes d'assurance payées par les assurés honnêtes.

Ce projet vise à exploiter les techniques basées sur les données et les algorithmes d'apprentissage automatique pour lutter contre ce problème récurrent. L'ensemble des données fournies comprend une collection complète d'attributs liés aux véhicules, y compris les modèles, les spécificités des accidents et les détails de la police tels que l'ancienneté et le type de police. Les objectifs englobent une approche à multiples facettes, commençant par une exploration et une analyse approfondies de l'ensemble de données. Cette analyse sera suivie d'étapes de prétraitement, si nécessaire, afin d'affiner et de préparer les données pour la modélisation.

Les points principaux de ce projet comprennent l'exploration et la mise en œuvre d'au moins deux modèles d'apprentissage automatique. Ces modèles seront entraînés et évalués en fonction de leur efficacité à distinguer les demandes d'assurance légitimes des demandes frauduleuses. En outre, l'objectif final est de fournir un outil Python convivial capable d'évaluer les données de réclamation entrantes pour détecter les soumissions frauduleuses potentielles, permettant ainsi aux utilisateurs de prendre des décisions plus éclairées concernant la légitimité des réclamations d'assurance. Grâce à cet outil, l'objectif est d'améliorer l'efficacité et la précision de l'identification des demandes d'indemnisation frauduleuses dans le domaine de l'assurance.

Le jeu de données **{insurance_fraud_dataset.csv}** comprend des informations sur des réclamations d'assurance automobile. Le jeu de données contient 1000 entrées et 40 colonnes.

2 Pré-traitement et analyse des données

2.1 pré-traitement des données

Le dataset est composé de 1000 lignes et 40 colonnes. L'analyse initiale révèle peu de valeurs manquantes, à l'exception de la colonne **_c39** qui semble entièrement constituée de valeurs nulles. Les données se répartissent principalement en entiers (**int64**), flottants (**float64**), et objets (**object**), suggérant la présence de variables catégorielles. La colonne **_c39** a été supprimée car elle ne contient pas d'informations pertinentes, tout comme les lignes contenant des valeurs manquantes dans la colonne **authorities_contacted**, afin de maintenir la cohérence des données. La liste finale des colonnes après le prétraitement inclut des attributs tels que **months_as_customer**, **age**, **policy_number**, **policy_bind_date**, **policy_state**, **incident_type**, **collision_type**, **incident_severity**, et **fraud_reported**, entre autres.

2.2 Exploration des Données

En examinant plus en détail les caractéristiques numériques de notre jeu de données, nous pouvons mieux comprendre la distribution de nos variables.

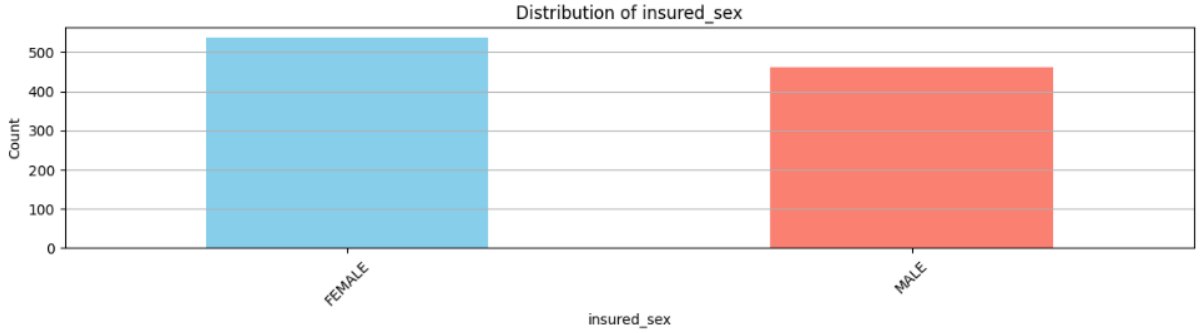
- La durée moyenne en mois en tant que client (**months_as_customer**) est d'environ 203.95 mois, avec une variabilité significative indiquée par un écart-type de 115.11 mois.
- L'âge moyen des assurés (**age**) est d'environ 38.95 ans, avec une dispersion de 9.14 ans. L'âge minimum est de 19 ans, tandis que l'âge maximum atteint 64 ans.
- La franchise moyenne de la police (**policy_deductable**) est de 1136, avec des valeurs allant de 500 à 2000.
- La prime annuelle moyenne (**policy_annual_premium**) s'élève à 1256.41, avec une plage allant de 433.33 à 2047.59.
- La limite moyenne de la garantie umbrella (**umbrella_limit**) est d'environ 1.10e+06, avec une variation considérable, comme en témoigne l'écart-type élevé.
- Le code postal moyen des assurés (insured_zip) est d'environ 501214.49, avec une variation notable.
- Les gains en capital moyens (**capital-gains**) sont d'environ 25126.10, tandis que les pertes en capital moyennes (**capital-loss**) sont d'environ -26793.70.
- L'heure moyenne de l'incident (**incident_hour_of_the_day**) est d'environ 11.64 heures, avec une plage de 0 à 23 heures.
- Le nombre moyen de véhicules impliqués (**number_of_vehicles_involved**) est d'environ 1.84, avec une majorité des incidents impliquant 1 à 2 véhicules.
- Le nombre moyen de blessures corporelles (**bodily_injuries**) est d'environ 0.99, avec une distribution relativement uniforme entre 0 et 3 blessures.
- Le nombre moyen de témoins (**witnesses**) est d'environ 1.49, indiquant généralement la présence d'un ou deux témoins lors des incidents.
- La somme totale des réclamations (**total_claim_amount**) a une moyenne de 52761.94 avec une variation allant de 100 à 114920. Les réclamations pour blessures (**injury_claim**) ont une moyenne de 7433.42, tandis que les réclamations de propriété (**property_claim**) et de véhicule (**vehicle_claim**) ont respectivement une moyenne de 7399.57 et 37928.95.
- L'année moyenne des véhicules impliqués (**auto_year**) est d'environ 2005.10, avec une fourchette allant de 1995 à 2015.

2.3 Analyse des données

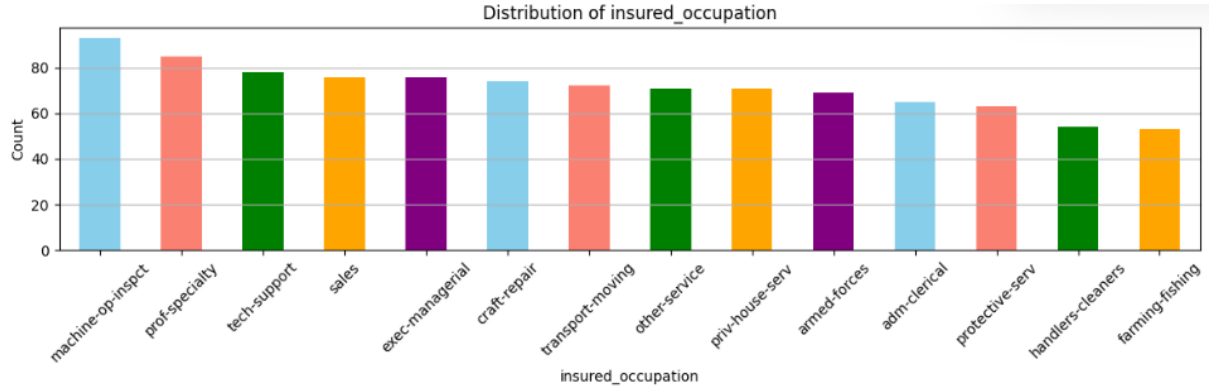
Nous poursuivons notre exploration en analysant visuellement la distribution des fraudes et en examinant la répartition de certaines catégories.

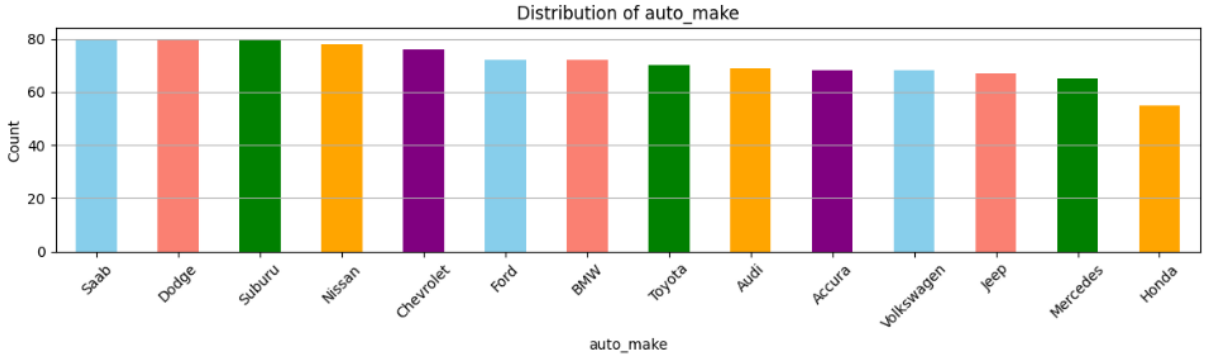
— **Distribution des Fraudes** : Nous avons utilisé un graphique pour illustrer la distribution des fraudes. On constate que le nombre de réclamations non frauduleuses (0) est d'environ 750, tandis que le nombre de réclamations frauduleuses (1) est d'environ 250.

— **Distribution par Catégories** : On a créé des graphiques à barres pour visualiser la distribution de différentes catégories en fonction de la fraude. Par exemple, dans la distribution du sexe assuré (**insured_sex**), on observe que les femmes représentent environ 525 occurrences, tandis que les hommes représentent environ 475 occurrences.
Ces graphiques nous offrent un aperçu visuel des tendances et des relations potentielles entre différentes catégories et la variable cible **fraud_reported**.

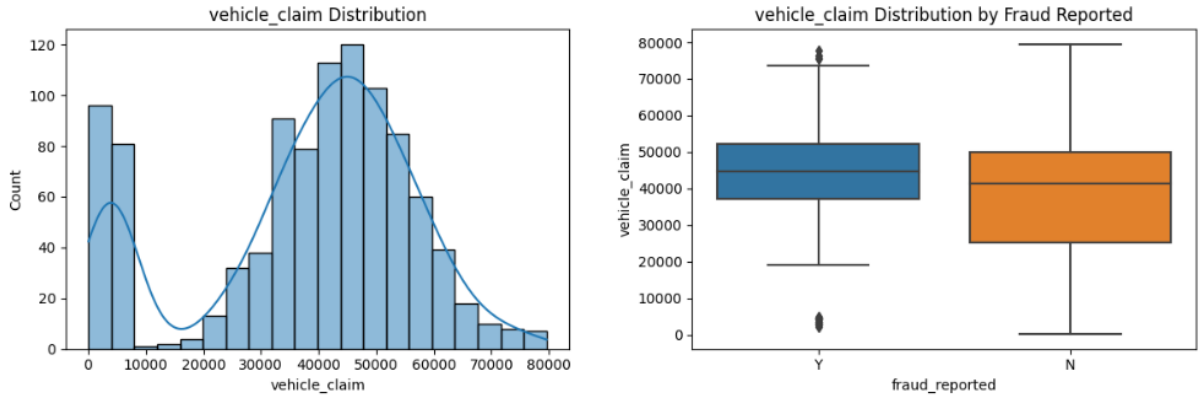


Voici deux autres exemples de distribution **insured_occupation** et **auto_make** :



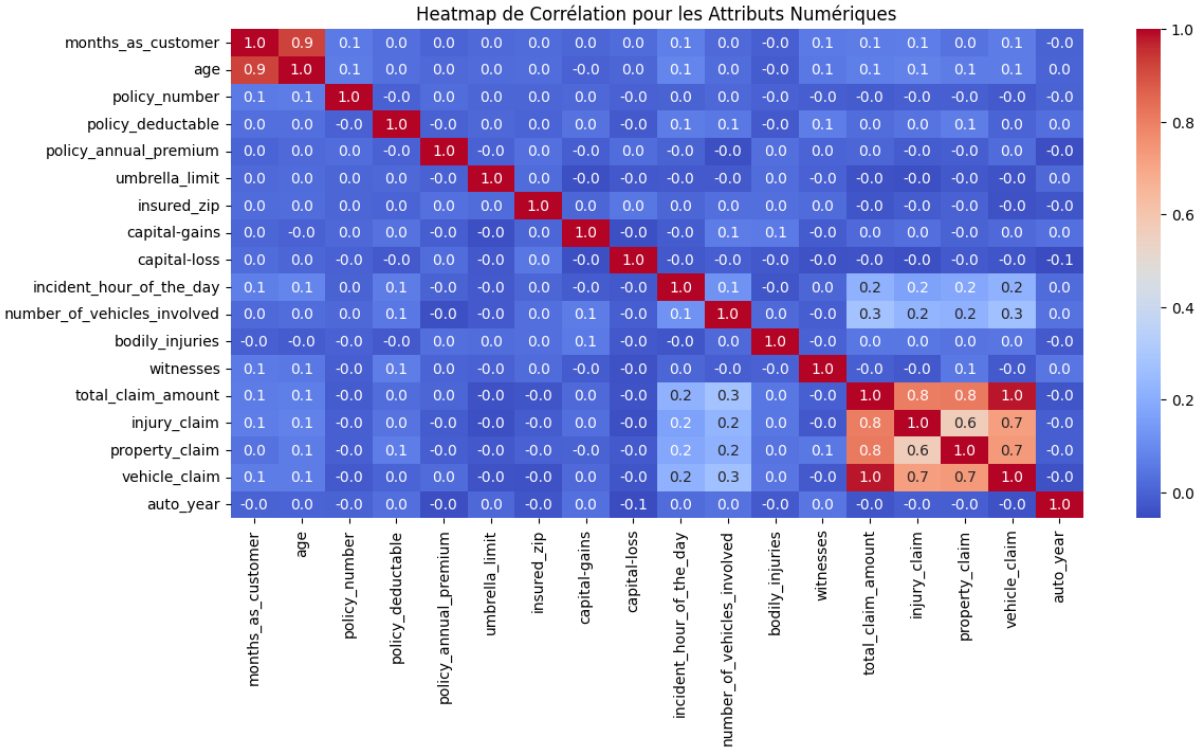


— **Histogrammes et Boxplots** : On a utilisé des histogrammes avec densité (KDE) pour visualiser la distribution des montants totaux de réclamation, des réclamations pour véhicule et des réclamations pour propriété. On a utilisé les boxplots pour comparer ces distributions en fonction de la fraude signalée.



Ces graphiques fournissent un aperçu des tendances de distribution pour les montants de réclamation totaux, les réclamations pour véhicule et les réclamations pour propriété, en distinguant également les différences potentielles entre les réclamations frauduleuses et non frauduleuses.

— **Matrice de Corrélation** : Pour mieux comprendre les relations linéaires entre les attributs numériques, On a calculé et visualisé une matrice de corrélation à l'aide d'un heatmap.



Les colonnes numériques, comprenant des variables telles que **months_as_customer**, **age**, **policy_deductable**, **total_claim_amount**, etc., ont été incluses dans le calcul de la matrice de corrélation.

Le heatmap de corrélation nous a permis de visualiser les forces et les directions des relations linéaires entre les différentes variables numériques. Les valeurs de corrélation varient de -1 à 1, où -1 indique une corrélation négative parfaite, 1 une corrélation positive parfaite, et 0 l'absence de corrélation.
La corrélation la plus élevée de valeur (1) est entre **total_claim_amount** et **vehicle_claim**. Or, la corrélation la plus basse (-0.1) est entre **capital-loss** et **auto_year**.

Une attention particulière peut être accordée aux paires d'attributs avec des coefficients de corrélation élevés, car cela peut indiquer une relation potentiellement importante entre ces variables.

2.4 Pré-traitement des Données

La phase de pré-traitement du jeu de données a impliqué des étapes précises pour optimiser le jeu de données en vue d'applications d'analyse et d'apprentissage automatique. Dans un premier temps, les colonnes catégorielles ont subi un encodage manuel, un processus qui consiste à mapper les valeurs catégorielles en représentations numériques. Cet encodage a permis de convertir les données non numériques en un format adapté aux calculs mathématiques et à la modélisation. De plus, une attention particulière a été accordée aux colonnes contenant des données binaires, où les valeurs 'OUI' et 'NON' ont été traduites en '1' et '0', respectivement, tandis que les valeurs inconnues ont été désignées '2' pour indiquer leur indisponibilité ou leur caractère non définis. Ensuite, les colonnes de type chaîne de caractères ont été soumises à un encodage automatique des étiquettes, utilisant l'instance LabelEncoder pour convertir les données textuelles en équivalents numériques. Cette stratégie de transformation a standardisé le jeu de données en convertissant les colonnes catégorielles basées sur des chaînes de caractères en un format numérique, assurant une uniformité dans la représentation des données à travers les différentes caractéristiques. Par conséquent, la plupart des colonnes du jeu de données présentent des types de données numériques, améliorant la compatibilité des données avec les algorithmes d'apprentissage automatique. Ces mesures de pré-traitement ont été mises en œuvre pour simplifier le jeu de données, le rendant utilisables pour les procédures suivantes d'analyse et d'entraînement.

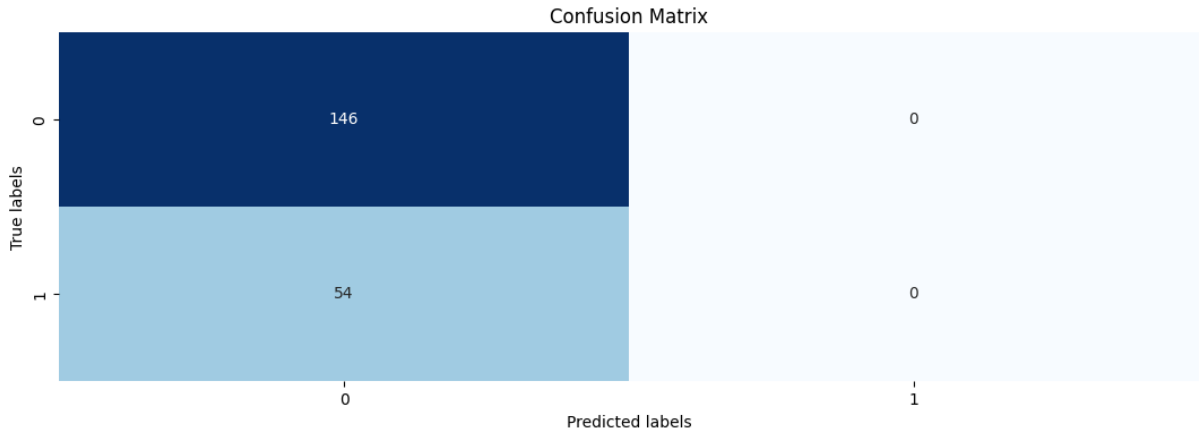
3 Machine Learning

Les données ont été divisées en ensembles d'entraînement (80%) et de test (20%) en utilisant la fonction **train_test_split** de scikit-learn.
Cette division nous a permis de créer un ensemble d'entraînement sur lequel les modèles peuvent être formés, et un ensemble de test pour évaluer leurs performances sur des données non vues.
On commence par définir une liste appelée "features" qui contient toutes les caractéristiques potentielles extraites du jeu de données. En parallèle, on identifie la variable cible, **fraud_reported**, que le modèle cherchera à prédire.
Ensuite, le jeu de données est divisé en deux ensembles distincts : un ensemble d'entraînement (**X_train, y_train**) et un ensemble de test (**X_test, y_test**). Cette division est primordiale pour évaluer la performance du modèle après son entraînement.

3.1 LogisticRegression

La régression logistique est un algorithme d'apprentissage supervisé utilisé pour la classification binaire. Dans ce cas, on détectera la fraude. Lors de l'entraînement, le modèle cherche à ajuster un hyperplan pour séparer au mieux les deux classes. Pour ce faire, il utilise une fonction logistique qui transforme la sortie en une probabilité, et un seuil est appliqué pour classer les instances. Dans ce cas, une régression logistique est entraînée sur les données d'entraînement en utilisant le solver 'saga' avec 1000 itérations. Le solver 'saga' est adapté aux grands ensembles de données et prend en charge les pénalités l1 et l2 pour gérer la régularisation et améliorer la convergence. La classification report (**classification_report(y_test, y_pred)**) et la matrice

de confusion permettent d'évaluer les performances du modèle. Dans la matrice de confusion, les valeurs en diagonale représentent les prédictions correctes pour chaque classe, tandis que les valeurs hors diagonale représentent les erreurs de prédiction. Ici, le modèle montre une précision



de 73% pour la classe 0, ce qui signifie que sur toutes les prédictions faites pour la classe 0, 73% étaient correctes. Dans la matrice de confusion, la classe 0 est associée à la valeur "Négatif" ou "Non Fraude". Cependant, la précision pour la classe 1 est de 0%, indiquant que le modèle ne prédit jamais cette classe. Dans la matrice de confusion, la classe 1 est associée à la valeur "Positif" ou "Fraude".
Le recall (taux de vrais positifs prédits parmi tous les vrais positifs) et le F1-score (moyenne pondérée de la précision et du recall) pour la classe 1 sont également nuls, ce qui montre l'incapacité du modèle à identifier la classe minoritaire (frauduleuse) dans ce contexte. Cela peut résulter d'un déséquilibre de classe dans les données ou d'une complexité insuffisante du modèle pour capturer la nature complexe de la fraude.

3.2 Polynomial Regression

C'est une boucle d'entraînement de modèles de régression polynomiale pour différents degrés de polynômes (1, 2 et 3). Chaque modèle est entraîné sur un sous-ensemble des données d'entraînement et par la suite utilisé pour faire des prédictions sur les données de test. Ensuite, on évalue la précision de chaque modèle.
On a utilisé un modèle de régression linéaire avec différents degrés de polynômes, créé à l'aide de la méthode **make_pipeline** de **scikit-learn**.

Les résultats obtenus indiquent que le modèle avec un degré de polynôme de 1 a la plus haute précision, avec une précision de 65.50%. Cependant, si le degré de polynôme augmente, la précision diminue de manière significative. On obtient notamment 40.00% pour le degré 2 et 27.00% pour le degré 3.

3.3 Random Forest

On a créé un processus de réglage des hyperparamètres pour un modèle de régression Random Forest. Cela implique de faire varier les paramètres **test_size**, **n_estimators**, et **max_depth** pour améliorer les performances prédictives du modèle. Cette manipulation des hyperparamètres nous permet d'identifier la configuration qui maximise l'exactitude du modèle, assurant des prédictions plus fiables sur les données de test.

Pour chaque division, un modèle de régression Random Forest est initialisé et entraîné sur l'ensemble d'entraînement (X_train, y_train).

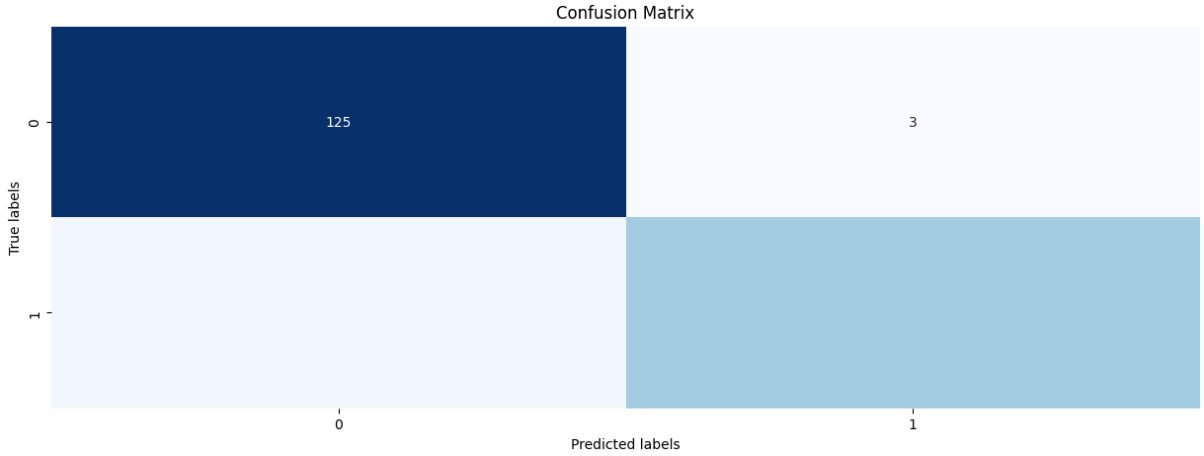
L'évaluation à travers diverses tailles d'ensemble de test a révélé une tendance : une taille de test de 0,2 a généralement conduit à des performances plus élevées comparée à des tailles de 0,15 ou 0,25. Cela suggère une meilleure adaptation du modèle à cette proportion de division.
Les variations de n_estimators et max_depth ont également eu un effet important sur les performances : l'exactitude variait en fonction de ces hyperparamètres. La configuration optimale, qui a donné une exactitude de 87,50%, était atteinte avec 100 arbres (n_estimators) et une profondeur maximale de 5 (max_depth).

Cette analyse ne peut que souligner l'importance du réglage détaillé des hyperparamètres afin d'obtenir des bonnes performances et de bien prédire la variable cible **fraud_reported**.

3.4 Random Forest Classifier

Les résultats de la classification présentent une précision très remarquable, indiquant la capacité du modèle à prédire avec précision les deux classes, frauduleuse (1) et non frauduleuse (0). La précision globale du modèle atteint 95%, ce qui signifie que sur l'ensemble des prédictions effectuées, 95% sont correctes. En effet, on obtient une précision de 95% pour la classe 0 (non frauduleuse) et de 94% pour la classe 1 (frauduleuse).

La matrice de confusion confirme ces performances élevées. Sur les 182 échantillons de test,



le modèle a correctement classé 128 échantillons de la classe 0 et 47 échantillons de la classe 1. Seulement trois échantillons de la classe 0 ont été classés à tort comme appartenant à la classe 1, indiquant une faible erreur de type faux positifs. Cela signifie que le modèle peut manquer certaines fraudes réelles.

L'analyse du rapport de classification confirme ces résultats en montrant des scores de rappel de 98% pour la classe 0 et de 87% pour la classe 1. Cela indique que le modèle est plus sensible à la détection des cas non frauduleux que des cas frauduleux, ce qui peut être crucial dans le contexte de la détection de fraudes.

Après avoir testé ces différents modèles, Random Forest Classifier présente des performances globalement élevées avec une précision satisfaisante, mais pourrait bénéficier d'une amélioration pour détecter plus efficacement les fraudes réelles tout en minimisant les faux positifs. C'est pourquoi, c'est le modèle qu'on va choisir.

4 Interface

On vous a fourni un outil python vous permettant de détecter si une demande d'indemnisation est frauduleuse ou non lorsque vous soumettez les données requises.

Voici l'interface d'utilisation.

Détection de fraude

Months as Customer:

350

Age:

20

Policy State:

OH

Insured Sex:

Male

Incident Type:

Single Vehicle Collision

Incident Severity:

Major Damage

Collision Type:

Front Collision

Insured Education Level:

PhD

Authorities Contacted:

Witnesses:

2

Total Claim Amount:

100

Injury Claim:

280

Property Claim:

20

Vehicle Claim:

1440

Auto Make:

BMW

Auto Model:

X5

Auto Year:

2011

Détecter la fraude

5 Conclusion

Ce projet de détection de fraudes d'assurance a été une exploration des modèles de classification. À travers l'utilisation de divers algorithmes tels que la régression logistique, les Random Forest et les modèles de régression polynomiale, nous avons étudié comment ces techniques peuvent être appliquées pour identifier les cas de fraudes potentielles dans les demandes d'assurance.

Bien que les modèles aient montré des performances significatives, il reste, comme toujours, des opportunités d'amélioration pour renforcer la capacité à détecter les fraudes réelles tout en minimisant les fausses alertes.