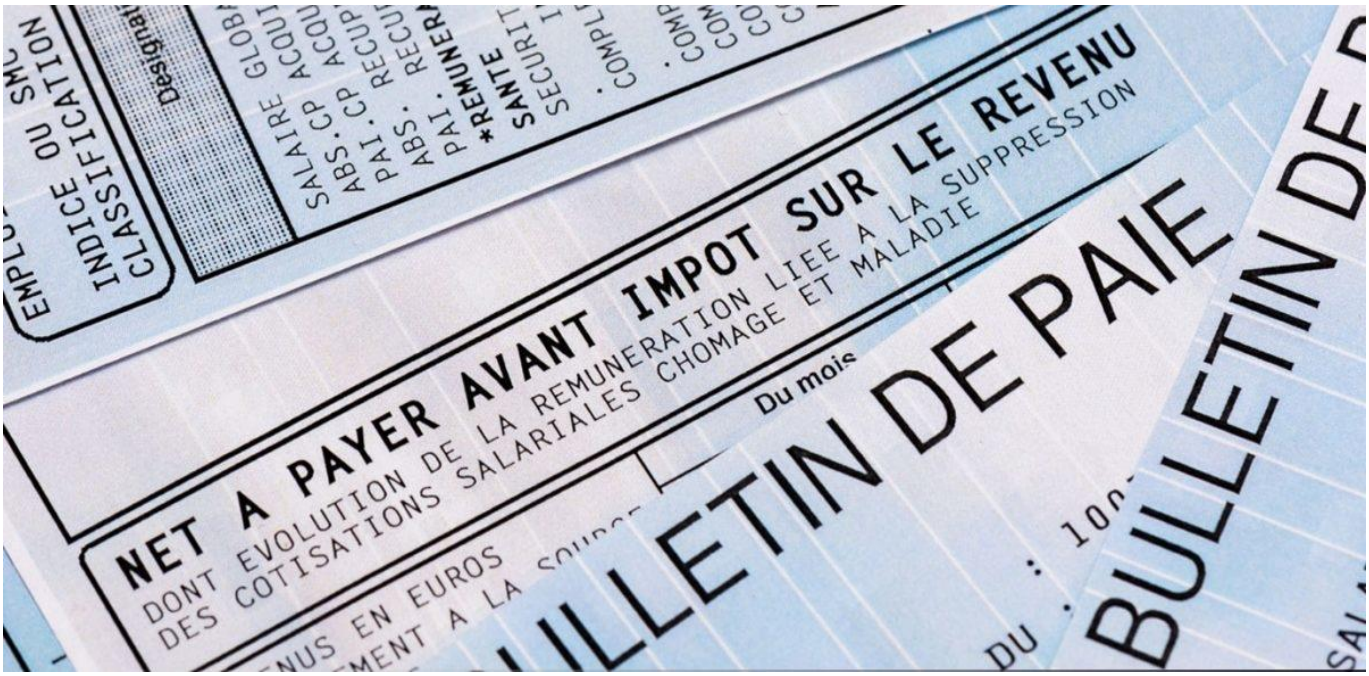


Rapport de projet : Détection de fraude de montant de salaire

~ 08/12/2021 ~

Alexis Guillotin
Clément Marie Brisson
Yolène Moysan

SOMMAIRE

Introduction

De nombreuses fraudes se font dans de nombreux domaines, notamment la fraude sur les bulletins de salaire. Ce projet a pour objectif de détecter des fraudes sur les montants de salaire, en utilisant de l'Intelligence Artificielle (IA) et Python.

Dans un premier temps, nous nous concentrons sur la préparation de notre dataset, ensuite, on s'intéressera aux différentes méthodes utilisées pour détecter la fraude sur les montants de salaire et pour finir, quels résultats nous avons obtenus.

Préparation de dataset

Notre dataset est composé de bulletins de salaires et des relevés de comptes correspondant. Pour préparer notre dataset. Pour enrichir notre dataset, nous avons créé un oversampling avec les exemples qu'on avait reçus, en ajoutant des exemples modifiés (avec des cas de fraude).

Méthodes utilisées

L'objectif de notre projet, est de pouvoir trouver une méthode qui nous permettra de détecter la fraude sur les montants de salaire, on récupère le montant net sur le bulletin de salaire et on le compare à celui du relevé de compte. Si le montant est différent, cela s'agit donc d'une fraude.

Dans premier temps, nous avons utilisé une méthode pour pouvoir détecter une fraude sur un unique cas. Cette méthode était de regarder où on pouvait récupérer le salaire net dans le bulletin de salaire. Après avoir remarqué qu'il se situait à la fin du fichier, nous avons choisi de le récupérer, en transformant nos fichiers en TXT, en reprenant la fin du texte et en supprimant « EUR » qui se situait derrière. Ensuite, nous avons fait une boucle qui permet de regarder dans tout le fichier relevé de compte, si un montant correspondait à celui du salaire net qui se situe dans le bulletin de salaire.

```python
#Récupérer le salaire net de la fiche de paie
words = str(list_traite_salaire[len(list_traite_salaire)-1])
lines = words.split("\n")
for line in lines:
    print(line)
    if "NET PAYÉ" in line:
        print ("TROUVE ! " +line)
salaire_net = words[len(words)-14:len(words)-6]
#print("Salaire net : "+salaire_net)

#Relevé de compte
fraude = True
for page in list_traite_compte:
    if salaire_net in page:
        print("Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.")
        fraude = False
    elif salaire_net.replace('.',',') in page:
        print("Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.")
        fraude = False
if fraude == True:
    print("ATTENTION ! Il n'y a pas de virement qui correspond au montant inscrit sur la fiche de paie.")
```

Voici le résultat de ce code sur 7 bulletins de salaire :

```
python3 Payslip.py
2021-12-08 09:34:45.824710: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dyn
brary 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object file: No such file or directo
2021-12-08 09:34:45.824775: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror i
o not have a GPU set up on your machine.
Vérification n° 1
        Lecture de releve_banque_1_page1...
        Lecture de bulletin_salaire1...
                SALAIRE NET:  3059,51
        CORRESPONDANCE : Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.
Vérification n° 2
        Lecture de releve_banque_2_page1...
        Lecture de bulletin_salaire2...
                SALAIRE NET:  2647,85
        CORRESPONDANCE : Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.
Vérification n° 3
        Lecture de releve_banque_3_page1...
        Lecture de bulletin_salaire3...
                SALAIRE NET:  4582,56
        CORRESPONDANCE : Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.
Vérification n° 4
        Lecture de releve_banque_4_page1...
        Lecture de bulletin_salaire4...
                SALAIRE NET: :8569,47
        SUSPICION DE FRAUDE : pas de virement qui correspond au montant inscrit sur la fiche de paie.
Vérification n° 5
        Lecture de releve_banque_5_page1...
        Lecture de bulletin_salaire5...
                SALAIRE NET: 5836,56
        SUSPICION DE FRAUDE : pas de virement qui correspond au montant inscrit sur la fiche de paie.
Vérification n° 6
        Lecture de releve_banque_6_page1...
        Lecture de bulletin_salaire6...
                SALAIRE NET:  1201,51
        SUSPICION DE FRAUDE : pas de virement qui correspond au montant inscrit sur la fiche de paie.
Vérification n° 7
        Lecture de releve_banque_7_page1...
        Lecture de bulletin_salaire7...
                SALAIRE NET:  2506,51
        CORRESPONDANCE : Il y a bien un virement qui correspond au montant affiché sur le buletin de salaire.
```

Cette méthode était efficace, étant donné qu'elle sait détecter la fraude (vérification 4 et 6, cas il ne retrouve pas le même montant de le relevé de compte, et la vérification 5, car le montant a été déplacé) mais elle a des limites. Elle ne fonctionne uniquement sur un seul type de bulletin de salaire.

Nous avons donc testé une deuxième méthode passant directement par l'IA. Cette méthode consiste à créer des fichiers CSV, où on renseignait le contenu du texte qu'on avait récupéré, tout en précisant si c'est une fraude ou non et pour finir, des mots clés permettant pouvant correspondre dans le relevé de compte.

```
enc = LabelEncoder()
# Load data.
data = pd.read_csv("data_final.csv", index_col=0, usecols=[0,1,2,4])
data = data.dropna()
samples = data.transcription
text_labels  = [label_name.lower() for label_name in data.medical_specialty]
labels = enc.fit_transform(np.array(text_labels))
# Transform data.
max_df = 0.5
min_df = 0.001
max_features = 1000
ngram_range = (1,1)
final_stopwords_list = stopwords.words('french')
tfidf_vectorizer = TfidfVectorizer(max_df=max_df, min_df=min_df, max_features=max_features, stop_words=final_stopwords_list, ngram_range=ngram_range)
tfidf = tfidf_vectorizer.fit_transform(samples)
feature_names = tfidf_vectorizer.get_feature_names()
# Title.
st.sidebar.header("Constructing dictonary of words.")
# Upper bound for tf-idf value.
max_df = 0.3
# Lower bound for tf-idf value.
min_df = 0.01
# Size of dictionary.
max_features = 500
# Dimensionality reduction.
dim_red = TruncatedSVD(n_components=2)
data_red = dim_red.fit_transform(tfidf)

# Number of trees.
n_estimators = 1000
# Define classifier.
forest_clf = RandomForestClassifier(n_estimators=n_estimators, max_depth=None, max_leaf_nodes=None, class_weight='balanced', oob_score=True, n_jobs=-1, random_state=0)
# Define grid.
parameters = {'max_leaf_nodes':np.linspace(20,35,14,dtype='int')}
# Balanced accuracy as performance measure.
clf = RandomizedSearchCV(forest_clf, parameters, n_iter=10, cv=3, scoring='accuracy',n_jobs=-1)
# Train/optimize classifier.
classifier = clf.fit(tfidf, labels)
# Retrieve optimum.
forest = classifier.best_estimator_
feature_importances = forest.feature_importances_
indices = np.argsort(feature_importances)[::-1]
st.sidebar.header("Customizing the model.")
n_estimators = 1000
max_leaf_nodes = 25
max_depth = 5
class_weight = 'balanced_subsample'
# Retrieve values.
y_true = labels
y_pred = classifier.predict(tfidf)
# Compute scores.
f1_score_ = f1_score(y_true,y_pred,average="weighted")
cm = confusion_matrix(y_true,y_pred)
print(f1_score_)
print(cm)
```

Cette IA s'est faite en différente étapes :
Le principe fondamental de notre projet est le **Natural Language Processing** (**NLP**), qui est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle, qui vise à créer des outils de traitement de la langue naturelle pour diverses applications. Il ne doit pas être confondu avec la linguistique informatique, qui vise à comprendre les langues au moyen d'outils informatiques.

Ensuite, nous avons préparé nos données en utilisant des **Preprocessing**, tel que :

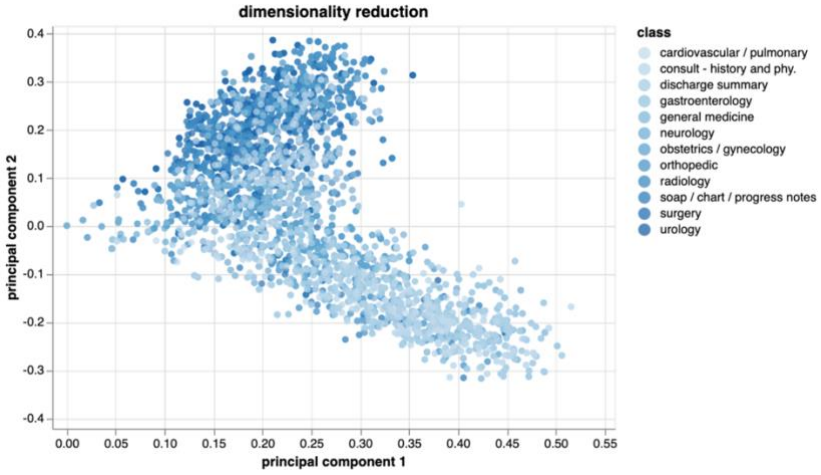
- Stopwords (ex: a, et, ou, puis etc…)
- Lemmatisation (ex: marche ; 'marcher', 'marché', 'marchons')
- Convertir le texte dans une forme standard
- Réduire la taille du texte
- Bag-of-Words, qui consiste à faire une représentation simplifiée du texte
- term frequency–inverse document frequency

Nous avons aussi utilisé la méthode **Term frequency–inverse document frequency (tf-idf)**, qui est une méthode de pondération. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \qquad \mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D) \qquad \mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Le résultat du calcul tf-idf est une matrice de dimensions N x D, c'est-à-dire nombre de documents multiplié par la taille du dictionnaire, remplie de valeurs tf-idf qui servent de caractéristiques.

La **réduction de la dimensionnalité** permet de ne garder que les attributs avec le poids le plus important, c'est à dire les attributs qui vont pouvoir déterminer le mieux possible si un bulletin est frauduleux ou non. En ne gardant que deux attributs on peut alors l'afficher sous la forme d'un tableau à deux dimensions. Voici un exemple d'un graphique représentant la réduction de la dimensionnalité



Pour finir, nous sommes à l'étape du **model building**. Cette étape permet de choisir le meilleur modèle, dans notre cas random forest, on entraine l'IA pour ainsi la tester.

Analyses et résultats de l'IA

Dans un premier temps, nous avions uniquement 3 jeux de bulletins de paie et de relevés de comptes. Alors, nous avons choisi de faire de l'oversampling. Et nous avons donc obtenu un F1 score de 1. Cependant, nous trouvions que l'échantillon n'était pas représentatif. C'est pourquoi nous avons décidé de modifier des bulletins de paie et des relevés de compte afin d'enrichir notre échantillon et ainsi améliorer notre oversampling. Voici le résultat obtenu (*In [1])* :

```
In [1]: runfile('/Users/dlyosa/Desktop/Projet-python/NLP.py', wdir='/Users/dlyosa/Desktop/
Projet-python')
1.0
[[166   0]
 [  0 333]]

In [2]: runfile('/Users/dlyosa/Desktop/Projet-python/NLP.py', wdir='/Users/dlyosa/Desktop/
Projet-python')
/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_selection/_split.py:670:
UserWarning: The least populated class in y has only 1 members, which is less than
n_splits=3.
  warnings.warn(("The least populated class in y has only %d"
0.9523367808062914
[[  1   0   0   0   0   0   0   0]
 [  0  10   0   0   0   0   0   0]
 [  0   0   6   0   0   0   0   0]
 [  0   0   0   1   0   0   0   0]
 [  0   0   0   0   1   0   0   0]
 [  0   0   0   0   0   1   0   0]
 [  0   0   1   0   0   0 111   5]
 [  0   0   0   0   0   0   8 148]]
```

Le deuxième résultat (*In [2]*), correspond à un test qu'on a effectué pour vérifier notre IA. En effet, nous avons récupéré une base de données avec de nombreuses données, et on voit que la prédiction de fraude est à 0.95. On peut donc dire que notre algorithme fonctionne bien.

Notre algorithme a rencontré des difficultés sur certains bulletins de salaires et certains relevés de compte à la suite de la mauvaise qualité de l'image. Nous avions alors pensé à améliorer la qualité des images en implémentant une IA. Cependant, cette méthode s'est avérée coûteuse en temps de développement. Nous avons donc préféré nous concentrer sur la partie de dataprep. Cependant, cette option nous aurait permit d'encore améliorer notre algorithme. En effet, avec une meilleure qualité d'image, l'algorithme aurait détecté le texte avec plus de facilité et aurait certainement eu une meilleure précision.

Conclusion

Pour conclure, on a utilisé deux méthodes qui ont fait leurs preuves tout ayant des limites. La principale limite étant le manque de données. Des ouvertures d'approfondissement sont possibles, notamment en automatisant pour une plus grande base de données. Comme on a pu le voir, notre Intelligence Artificielle fonctionne correctement, et permet donc de détecter la fraude sur le bulletin de salaire.