Rapport

Auteur
Ibrahim Traore, Haroun Mezrioui

Université Paris Dauphine

Mai 2023

1 Introduction

L'intelligence artificielle et l'apprentissage automatique ont connu une croissance exponentielle ces dernières années, transformant de nombreux domaines d'activité. Dans ce rapport, nous présentons notre projet d'apprentissage automatique axé sur la prédiction de la présence du COVID-19 chez une personne.

Le COVID-19, une maladie infectieuse causée par le coronavirus SARS-CoV-2, a eu un impact majeur sur la santé publique mondiale. La détection précoce des personnes atteintes du COVID-19 est cruciale pour contenir la propagation du virus et fournir des soins médicaux appropriés. Dans cette optique, nous avons utilisé un ensemble de données téléchargé à partir de Kaggle, qui regroupe des informations cliniques et des caractéristiques médicales de patients présumés atteints du COVID-19.

Notre objectif principal dans ce projet est de développer un modèle de prédiction basé sur l'apprentissage automatique capable de déterminer si une personne est atteinte ou non du COVID-19 en se basant sur ces caractéristiques médicales. Cette tâche revêt une importance capitale dans la lutte contre la propagation du virus, car elle peut aider à orienter les efforts de dépistage et de prise en charge des patients.

Pour atteindre cet objectif, nous avons suivi une méthodologie rigoureuse comprenant plusieurs étapes clés. Tout d'abord, nous avons procédé à l'exploration et à l'analyse approfondie du jeu de données, en examinant les caractéristiques disponibles et en identifiant les éventuelles valeurs manquantes ou aberrantes. Ensuite, nous avons effectué un prétraitement des données, comprenant la normalisation, la gestion des valeurs manquantes et la conversion des variables catégorielles en variables numériques.

Par la suite, nous avons sélectionné et entraîné plusieurs modèles d'apprentissage automatique, tels que les algorithmes de classification tels que le SVM, les forêts aléatoires afin de prédire la présence du COVID-19 chez les individus. Nous avons optimisé les hyperparamètres de ces modèles pour améliorer leurs performances et nous les avons évalués en utilisant des métriques appropriées telles que la précision, le rappel et la F-mesure.

Ce rapport est organisé de la manière suivante : dans la section suivante, nous décrirons en détail la méthodologie que nous avons suivie pour mener à bien ce projet, y compris la collecte des données et les étapes de prétraitement. Ensuite, nous présenterons les différents modèles d'apprentissage automatique que nous avons utilisés, leurs entraînements et les résultats obtenus. Nous conclurons en récapitulant les résultats, en identifiant les limitations et en suggérant des pistes d'amélioration pour de futures recherches dans le domaine de la prédiction du COVID-19.

Ce projet a été une opportunité passionnante de mettre en pratique nos connaissances en apprentissage automatique.

2 Objectifs

Dans ce projet, nous avons identifié à la fois des objectifs commerciaux et des objectifs spécifiques en apprentissage automatique pour guider notre démarche.

1. Améliorer les décisions de dépistage et de prise en charge des patients : Notre objectif commercial principal était d'améliorer la capacité des professionnels de la santé à prendre des décisions éclairées en matière de dépistage et de prise en charge des patients atteints du COVID-19. En fournissant un modèle prédictif précis, nous espérions réduire les erreurs de diagnostic et orienter les actions médicales de manière appropriée.

2. Développer un modèle prédictif performant : Notre objectif en apprentissage automatique était de construire un modèle de prédiction robuste et précis pour déterminer

si une personne est atteinte du COVID-19 ou non. Nous cherchions à maximiser les métriques d'évaluation telles que la précision, le rappel et la F-mesure, afin d'obtenir un modèle fiable pour la prédiction.

3. Identifier les caractéristiques médicales pertinentes : Nous cherchions à explorer les données et à identifier les caractéristiques médicales les plus pertinentes pour la prédiction du COVID-19. L'objectif était de comprendre les facteurs cliniques importants liés à la maladie et d'orienter les futures recherches et les décisions médicales en fonction de ces résultats.

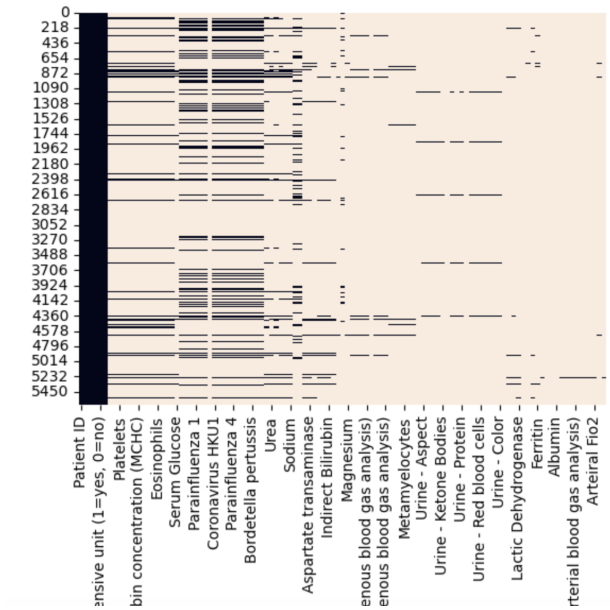


Figure 2: Visualisation de valeurs manquantes.

3 Exploration de données

3.1 Description de données

Nous avons 5644 observations et 111 variables explicatives dans notre dataset

	Patient ID	Patient age quantile	SARS-Cov-2 exam result	Patient addmited to regular ward (1=yes, 0=no)	Patient addmited to semi-intensive unit (1=yes, 0=no)	Patient addmited to intensive care unit (1=yes, 0=no)	Hematocrit	Hemoglobin	Platelets	Mean platelet volume	Red blood Cells	Lymphocytes	Mean corpusc hemogl concentration (MC
0	44477f75e8169d2	13	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	
1	126e9dd13932f68	17	negative	0	0	0	0.236515	-0.02234	-0.517413	0.010677	0.102004	0.318366	-0.9
2	a46b4402a0e5696	8	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	
3	f7d619a94f97c45	5	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	
4	d9e41465789c2b5	15	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	

Figure 1: Apercu de notre dataset.

3.2 Analyse des valeurs manquantes

Dans notre dataset nous avons 88% de valeurs manquantes

Après avoir visualiser le pourcentage de valeurs manquantes nous avons decider (puisqu'on a beaucoup de variables manquantes) d'eliminer toutes les variables dont le pourcentage de données manquantes est superieur à 90% ce sont des colonnes inutiles.

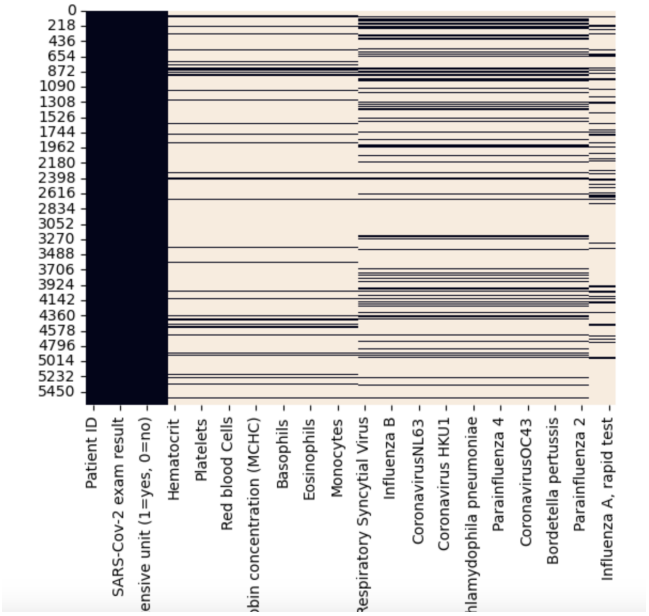


Figure 3: Visualisation après suppression.

Après cette operation de suppression on est

passé de 111 colonnes à 39 colonnes.

Nous avons decidé aussi d'eliminer la colonne "Patient ID" car cette colonne est inutile elle ne contient que l'ID des patients ce qui nous importe peu voire pas du tout. A la fin on aura 5644 lignes et 38 colonnes.

	Patient age quantile	SARS-Cov-2 exam result	Patient addmited to regular ward (1=yes, 0=no)	Patient addmited to semi-intensive unit (1=yes, 0=no)	Patient addmited to intensive care unit (1=yes, 0=no)	Hematocrit	Hemoglobin	Platelets	Mean platelet volume	Red blood Cells	Lymphocytes	Mean corpuscular hemoglobin concentration (MCHC)	Leukocyt
0	13	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	17	negative	0	0	0	0.236515	-0.022340	-0.517413	0.010677	0.102004	0.318366	-0.950790	-0.0946
2	8	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	5	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	15	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	
5639	3	positive	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5640	17	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5641	4	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5642	10	negative	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5643	19	positive	0	0	0	0.694287	0.541564	-0.906829	-0.325903	0.578024	-0.295726	-0.353319	-1.2884

5644 rows × 38 columns

Figure 4: Visualisation .

3.3 Variable Target

Nous observons que notre dataset contient 5086 cas negatifs et 558 cas positifs. Nous pouvons affirmer que nos classes ne sont pas equilibrées.

```
df['SARS-Cov-2 exam result'].value_counts()
```

```
negative    5086
positive     558
Name: SARS-Cov-2 exam result, dtype: int64
```

3.4 Histogrammes des variables continue

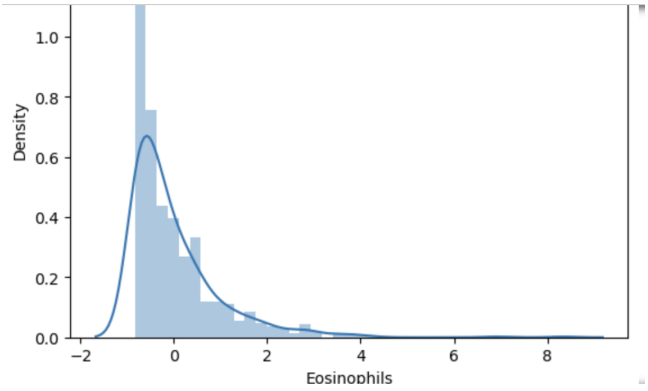


Figure 5: Visualisation de la variable Eosinophils .

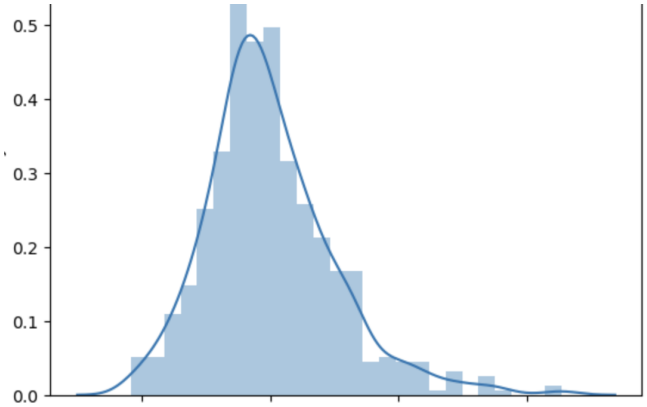
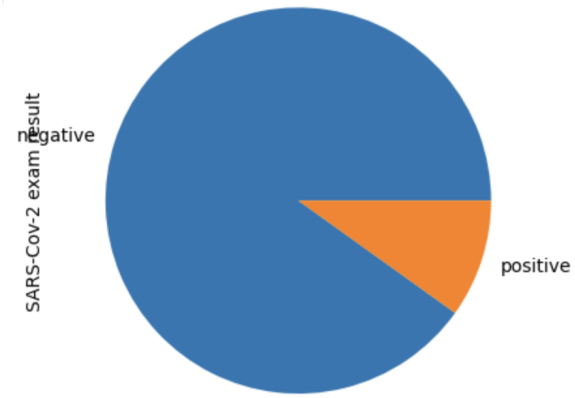
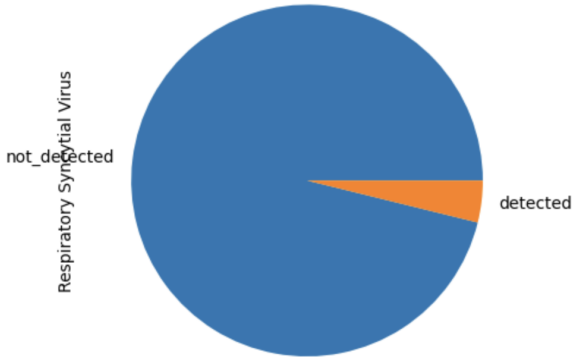


Figure 6: Visualisation de la variable Monocytes .

La première chose que nous voyons c'est que toutes nos courbes sont centrées en zeros et un ecart-type egale à 1 ce qui nous laisse comprendre que ces données ont été standardisées.Nous pouvons aussi voir que pas mal de ces variables suivent une distribution normale.

3.5 Pie charts des variables qualitatives





3.6 Relation Target/Variables-Blood

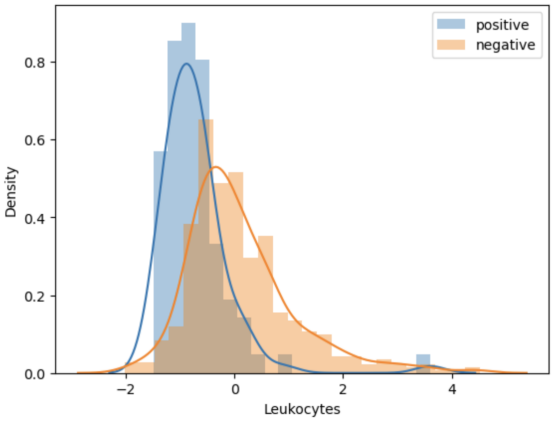


Figure 7: Variable Leukocytes.

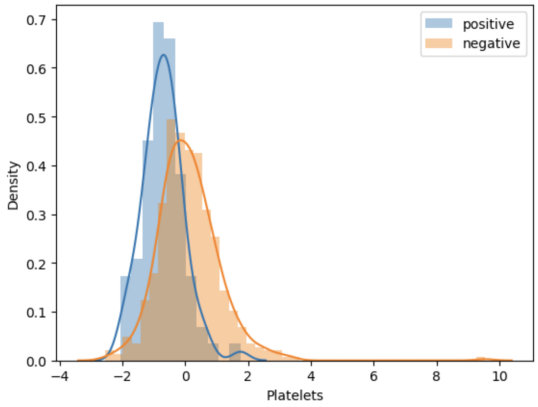


Figure 8: Variables Platelets.

Nous constatons que les personnes atteintes du covid-19 ont des taux de platelets et de leukocytes differents de celles qui sont negatives alors la conclusion que nous pouvons tirer c'est que les platelets et les leukocytes sont des variables significatives. C'est ainsi que nous avons procédé pour toutes nos variables de types sang (nous avons divisé notre dataset en deux types de variables Sang et Virus).

3.6.1 Test de Student

Le test de Student, également connu sous le nom de test t de Student, est un test statistique utilisé pour comparer les moyennes de deux échantillons indépendants et déterminer s'il existe une différence significative entre elles.

En visualisant la relation Target/ Variables nous etions arriver à la conclusion que les taux moyens de "Leukocytes,Platelets,Monocytes,Eosinophils,Basophils" sont differents chez les individus positifs et negatifs. Pour etre beaucoup plus rigoureux nous allons faire un test de student pour confirmer nos dits.

Notre hypothèse H0 est le suivant: H0 = Les taux moyens sont ÉGAUX chez les individus positifs et négatifs.

```
for col in blood_columns:
    print(f'{col :-<50} {t_test(col)}')
```

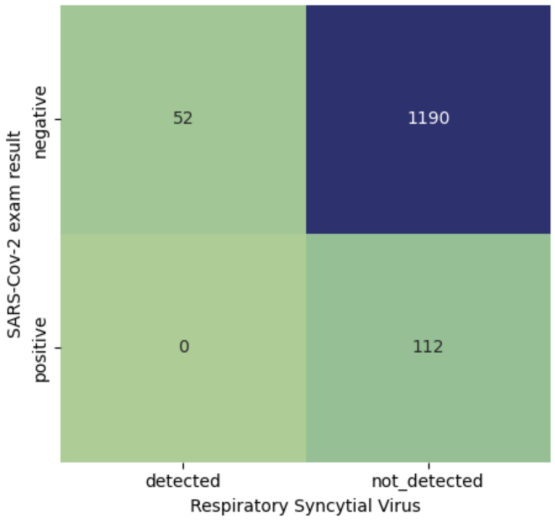
```
Hematocrit---------------------------------------- 0
Hemoglobin---------------------------------------- 0
Platelets----------------------------------------- H0 Rejetée
Mean platelet volume ------------------------------ 0
Red blood Cells----------------------------------- 0
Lymphocytes--------------------------------------- 0
Mean corpuscular hemoglobin concentration (MCHC)-- 0
Leukocytes---------------------------------------- H0 Rejetée
Basophils----------------------------------------- 0
Mean corpuscular hemoglobin (MCH)----------------- 0
Eosinophils--------------------------------------- H0 Rejetée
Mean corpuscular volume (MCV)--------------------- 0
Monocytes----------------------------------------- H0 Rejetée
Red blood cell distribution width (RDW)----------- 0
```

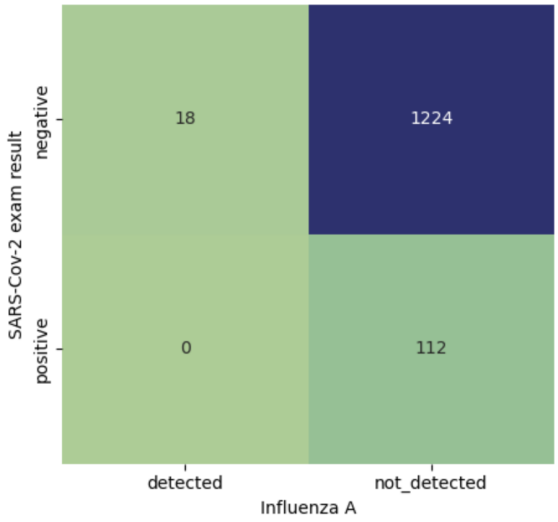
Figure 9: Test de Student.

Nous avons des preuves statistiquement significatives pour conclure qu'il y a une différence significative entre les taux moyens des variables "Leukocytes","Basophils","Platelets","Monocytes" et donc ces 4 variables sont importantes pour predire si une personne est atteinte ou non par le covid-19.

3.7 Relation Target/Test viral

En comparant notre target avec les variables de type "test viral" nous observons que nos variables n'ont aucun lien avec le covid-19 donc ils ne sont pas significatives mais attention nous avons jugé necessaire de verifier cette hypothèse dans la partie pre-processing. On va appliquer un modèle en gardant ces variables et appliquer un modèle en les enlevant et par la suite comparer les scores pour voir si ce qu'on dit est vrai ou faux.





3.8 Analyse de la variable Patient age quantile

Il est difficile pour nous d'interpreter pourquoi l'age dans notre cas est entre 0 et 19 alors qu'en general l'age est entre 1 et 100 ans et malheureusement la personne qui a posté ce dataset sur kaggle n'a laissé aucune information sur la variable "Patient age quantile" donc tout ce que nous pouvons faire c'est emettre des hypothèses. En jettant un coup d'oeil dans la discussion sur kaggle on s'est aperçu que beaucoup de personnes ont été confronté à ce problème aussi donc l'hypothèse qu'ils ont emis c'est de voir ces chiffres comme des tranches d'age: 1 [0 5] 2 [6 10] 3 [11 15] 4 [16 20] 5 [21 25] 6 [26 30] .....

Grace à notre fonction countplot nous avons pu dementir une information qui circulait en disant que les enfants ne peuvent pas etre atteint du covid-19 nous avons belle et bien vu qu'enfant comme adulte sont exposés à cette maladie.

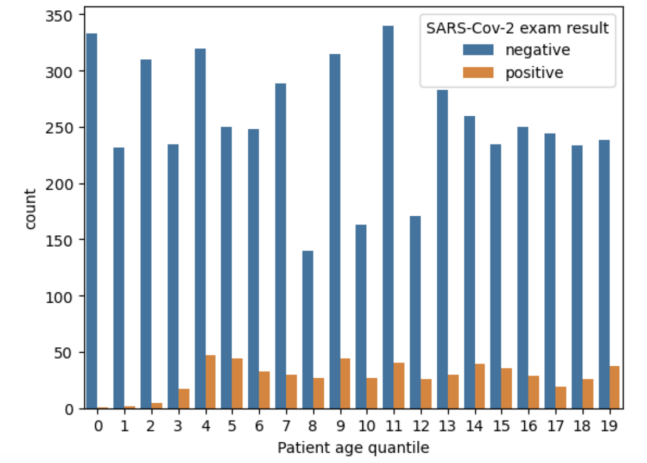


Figure 10: Countplot de patient age quantile.

4 Pre-traitement

Le prétraitement des données est une étape essentielle dans tout projet de machine learning. Il vise à préparer les données brutes avant de les utiliser pour entraîner un modèle. Dans cette section, nous décrirons les différentes étapes de prétraitement que nous avons appliquées à notre dataset afin de garantir la qualité et la pertinence des données utilisées dans notre projet de prédiction du COVID-19.

4.1 Sélection des variables pertinentes

Avant de procéder à la gestion des valeurs manquantes, nous avons effectué une sélection des variables pertinentes. Cette étape nous permet de réduire le nombre de variables à traiter et de focaliser notre analyse sur les caractéristiques les plus informatives. Nous

avons decidé de supprimer les variables de type viral.

```
blood_columns = list(df.columns[(missing_rate < 0.9) & (missing_rate >0.88)])
viral_columns = list(df.columns[(missing_rate < 0.80) & (missing_rate > 0.75)])
```

```
key_columns = ['Patient age quantile', 'SARS-Cov-2 exam result']
ds = df[key_columns + blood_columns]
```

```
df_imp = ds.copy()
```

4.2 Gestion des valeurs manquantes

Lors de l'exploration initiale de notre dataset, nous avons identifié la présence de valeurs manquantes dans certaines variables. Afin de préserver l'intégrité des données et de minimiser l'impact sur les performances de notre modèle, nous avons mis en place une stratégie de gestion des valeurs manquantes.

Nous avons divisé notre dataset en deux groupes : les observations avec un résultat négatif pour l'examen COVID-19 (neg) et celles avec un résultat positif (pos). Pour chaque groupe, nous avons utilisé la méthode de la médiane pour remplacer les valeurs manquantes dans les variables numériques.

```
# imputing median values on the missing values
neg = df_imp[df_imp['SARS-Cov-2 exam result']=='negative']
neg.fillna(neg.median(), inplace=True)
pos = df_imp[df_imp['SARS-Cov-2 exam result']=='positive']
pos.fillna(pos.median(), inplace= True)
```

4.3 Encodage des variables catégorielles

Une étape essentielle du prétraitement des données est l'encodage des variables catégorielles pour les rendre compatibles avec les algorithmes de machine learning. Dans notre projet, nous avons utilisé une fonction d'encodage personnalisée appelée encode() pour transformer les valeurs catégorielles en valeurs numériques. Voici comment la fonction d'encodage a été implémentée :

```
def encode(df):
    code = {
              'positive' : 1,
              'negative' : 0

            }
    for col in df.select_dtypes('object'):
        df.loc[:,col] = df[col].map(code)

    return df
```

5 Construction et évaluation du modèle

Une fois que nous avons effectué le prétraitement des données, nous entrons dans la phase de construction et d'évaluation du modèle. Cette étape cruciale consiste à choisir un modèle approprié pour notre projet de prédiction du COVID-19, à l'entraîner sur les données prétraitées et à évaluer ses performances.

5.1 Gestion du déséquilibre des classes avec SMOTE

Dans notre dataset, nous avons constaté un déséquilibre entre les classes positives et négatives de notre variable cible 'SARS-Cov-2 exam result'. Cela signifie que le nombre d'échantillons pour la classe positive est significativement inférieur au nombre d'échantillons pour la classe négative. Un tel déséquilibre peut affecter les performances de notre modèle, car il peut entraîner un biais vers la classe majoritaire.

Pour remédier à ce déséquilibre, nous avons appliqué une technique appelée SMOTE (Synthetic Minority Over-sampling Technique). SMOTE est une méthode de sur-échantillonnage qui génère de nouveaux échantillons synthétiques de la classe minoritaire pour équilibrer les classes. Voici comment nous l'avons intégré dans notre code :

```
X = dp.drop('SARS-Cov-2 exam result', axis=1)
y = df_median['SARS-Cov-2 exam result']
```

```
X, y = SMOTE().fit_resample(X, y)
```

5.2 Division des données en ensembles d'entraînement et de test

Après avoir effectué la gestion du déséquilibre des classes avec SMOTE, nous avons divisé notre dataset en ensembles d'entraînement et de test. Cette division nous permet de mesurer les performances de notre modèle sur des données indépendantes qu'il n'a pas encore vues.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

5.3 Mise à l'échelle des valeurs

La mise à l'échelle des valeurs est une étape essentielle, notamment pour les modèles qui sont sensibles à l'échelle des variables. Cette étape vise à transformer les valeurs des caractéristiques afin qu'elles se situent dans une plage spécifique, ce qui facilite la convergence du modèle et garantit une comparaison équitable entre les variables.

Dans notre projet, nous avons appliqué trois méthodes de mise à l'échelle appelée "Scaler","MinMaxScaler","RobustScaler" sur nos variables explicatives (X) avant de les utiliser pour entraîner notre modèle de prédiction du COVID-19.

5.4 Modèle

Dans cette section, nous allons présenter les modèles que nous avons utilisés pour prédire le COVID-19 à partir des données prétraitées. Notre objectif était de développer des modèles performants qui puissent classifier de manière précise si une personne est positive ou négative au COVID-19 en se basant sur différentes caractéristiques.

- AdaBoost : L'algorithme AdaBoost est une méthode d'ensemble qui combine plusieurs modèles simples pour créer un modèle fort. Il fonctionne en ajustant itérativement les poids des échantillons mal classés, permettant ainsi de se concentrer sur les échantillons difficiles à classifier.

- Random Forest : La forêt aléatoire est un algorithme d'ensemble basé sur les arbres de décision. Il crée un ensemble d'arbres de décision et combine leurs prédictions pour obtenir une prédiction finale. Les arbres de décision individuels sont construits de manière aléatoire et combinés pour réduire le surajustement et améliorer les performances.

- SVM : Support Vector Machine (SVM) est un algorithme d'apprentissage automatique supervisé utilisé à la fois pour la classification et la régression. Bien que nous disions également des problèmes de régression, cela convient mieux à la classification. L'objectif de l'algorithme SVM est de trouver un hyperplan dans un espace à N dimensions qui classe distinctement les points de données. La dimension de l'hyperplan dépend du nombre d'entités. Si le nombre d'entités en entrée est de deux, l'hyperplan n'est qu'une ligne. Si le nombre d'entités en entrée est de trois, l'hyperplan devient un plan 2D. Il devient difficile d'imaginer quand le nombre de fonctionnalités dépasse trois.

5.5 Entrainement et evaluation des performances des modèles

- AdaBoost

```
model_1 = make_pipeline(StandardScaler(),AdaBoostClassifier())
model_1.fit(X_train,y_train)
pred = model.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1551
           1       0.98      0.98      0.98      1501

    accuracy                           0.98      3052
   macro avg       0.98      0.98      0.98      3052
weighted avg       0.98      0.98      0.98      3052
```

```
model_1 = make_pipeline(RobustScaler(),AdaBoostClassifier())
model_1.fit(X_train,y_train)
pred = model.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1551
           1       0.98      0.98      0.98      1501

    accuracy                           0.98      3052
   macro avg       0.98      0.98      0.98      3052
weighted avg       0.98      0.98      0.98      3052
```

- RandomForest

```
model_2 = make_pipeline(StandardScaler(),RandomForestClassifier(random_state=0))
model_2.fit(X_train,y_train)
pred = model_2.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1551
           1       0.99      1.00      0.99      1501

    accuracy                           0.99      3052
   macro avg       0.99      0.99      0.99      3052
weighted avg       0.99      0.99      0.99      3052
```

```
model_2 = make_pipeline(MinMaxScaler(),RandomForestClassifier(random_state=0))
model_2.fit(X_train,y_train)
pred = model_2.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1551
           1       0.99      1.00      0.99      1501

    accuracy                           0.99      3052
   macro avg       0.99      0.99      0.99      3052
weighted avg       0.99      0.99      0.99      3052
```

- XGBoost

```
model_3 = make_pipeline(RobustScaler(),xgboost.XGBClassifier(n_estimators=200, max_depth=4))
model_3.fit(X_train,y_train)
pred = model_3.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1551
           1       0.99      1.00      0.99      1501

    accuracy                           0.99      3052
   macro avg       0.99      0.99      0.99      3052
weighted avg       0.99      0.99      0.99      3052
```

```
model_3 = make_pipeline(MinMaxScaler(),xgboost.XGBClassifier(n_estimators=200, max_depth=4))
model_3.fit(X_train,y_train)
pred = model_3.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1551
           1       0.99      1.00      0.99      1501

    accuracy                           0.99      3052
   macro avg       0.99      0.99      0.99      3052
weighted avg       0.99      0.99      0.99      3052
```

- SVM

6 Optimisation des paramètres

Dans cette section, nous avons utilisé la méthode GridSearchCV pour optimiser les hyperparamètres de nos modèles. GridSearchCV est une technique de recherche exhaustive qui permet d'explorer différentes combinaisons d'hyperparamètres afin de trouver celle qui maximise les performances du modèle.

Nous avons appliqué GridSearchCV aux modèles suivants : AdaBoost, Random Forest, XGBoost et SVM. Chaque modèle a été évalué en utilisant une grille de paramètres spécifique, permettant de rechercher les meilleures valeurs pour les hyperparamètres du modèle.

6.1 Résultats de l'optimisation des paramètres

- AdaBoost

Table 1: Paramètres pour Adaboost

Paramètre	Valeurs
Nombre d'estimateur	10,50,250,1000
Learning-rate	0.01,0.1

Ces résultats mettent en évidence la performance du modèle AdaBoost, avec un score de 0.9868. Les paramètres optimaux identifiés sont un nombre

d'estimateurs de 1000 et un taux d'apprentissage de 0.1, ce qui suggère que l'AdaBoost atteint de bonnes performances en utilisant un grand nombre d'estimateurs et un taux d'apprentissage moyen

Table 2: Résultats de GridSearch CV pour AdaBoost

Meilleur score	0.9868
Meilleurs paramètres	
Learning-rate	0.1
N-estimators	1000

- RandomForest

Table 3: Paramètres pour RandomForest

Paramètre	**Valeurs**
Nombre d'estimateur	200, 500
Profondeur maximale	4,5,6,7,8
critère	['gini', 'entropy']

Ces résultats soulignent les performances solides du modèle Random Forest, avec un score élevé de 0.9930. Les paramètres optimaux, tels que le critère de séparation ('gini'), la profondeur maximale (7) et le nombre d'estimateurs (200), ont été identifiés comme étant les plus performants pour notre tâche d'apprentissage automatique.

Table 4: Résultats de GridSearch CV pour RandomForest

Meilleur score	0.9930
Meilleurs paramètres	
criterion	gini
Max-depth	7
N-estimators	200

- XGBoost

Les performances exceptionnelles du modèle XGBoost ont été mises en évidence grâce à une optimisation minutieuse des hyperparamètres. En utilisant un taux d'apprentissage de 0.1, une profondeur maximale de 5 et 200 estimateurs, nous avons obtenu un score remarquable de **0.9933** lors de l'évaluation par GridSearch CV. Ces résultats démontrent la puissance et la précision de XGBoost dans la résolution de notre problème.

Table 5: Paramètres pour XGBoost

Paramètre	**Valeurs**
Nombre de profondeur	3, 5, 7
Nombre d'estimateur	100, 200, 300
Learning Rate	0.1, 0.5, 1.0

Table 6: Résultats de GridSearch CV pour XGBoost

Meilleur score	0.9933
Meilleurs paramètres	
Learning Rate	0.1
Nombre de profondeur	5
Nombre d'estimateur	200

6.2 Interpretation des resultats

L'objectif de notre étude était d'optimiser les performances de plusieurs modèles d'apprentissage automatique (XGBoost, Random Forest, AdaBoost, SVM) en utilisant la technique de recherche de grille (GridSearchCV) pour trouver les meilleurs paramètres.

Pour le modèle XGBoost, nous avons testé différentes combinaisons de paramètres, notamment le taux d'apprentissage, le nombre d'estimateurs et la profondeur maximale de l'arbre. Après avoir effectué la recherche par grille, nous avons obtenu un score optimal de 0.9933. Les meilleurs paramètres identifiés étaient un taux d'apprentissage de 0.1, une profondeur maximale de 5 et 200 estimateurs.

En ce qui concerne le modèle Random Forest, nous avons exploré les paramètres tels

que le nombre d'estimateurs, la profondeur maximale de l'arbre et le critère de division. Le score optimal obtenu était de 0.9930 avec les paramètres suivants : critère 'gini', profondeur maximale de 7 et 200 estimateurs.

Pour le modèle AdaBoost, nous avons varié le nombre d'estimateurs et le taux d'apprentissage. Le score optimal obtenu était de 0.9868 avec 1000 estimateurs et un taux d'apprentissage de 0.1.

En analysant ces résultats, nous pouvons conclure que les trois modèles (XGBoost, Random Forest et AdaBoost) ont atteint des performances élevées avec des scores supérieurs à 0.98. Cela démontre l'efficacité de ces modèles dans la tâche de prédiction sur notre ensemble de données.

Comparativement, le modèle XGBoost a obtenu le meilleur score avec une précision de 0.9933, suivi par le modèle Random Forest avec une précision de 0.9930 et le modèle AdaBoost avec une précision de 0.9868.

Il est important de souligner que ces performances élevées peuvent être attribuées à la recherche par grille qui nous a permis d'identifier les combinaisons optimales de paramètres pour chaque modèle. Cette approche systématique a joué un rôle clé dans l'amélioration des performances des modèles.

En conclusion, notre analyse des résultats confirme l'efficacité des modèles XGBoost, Random Forest et AdaBoost dans la tâche de prédiction. Les résultats obtenus mettent en évidence l'importance de l'optimisation des paramètres pour améliorer les performances des modèles d'apprentissage automatique. Ces informations sont précieuses pour guider les futures études et applications qui nécessitent des prédictions précises.

References

[1] C. Huyen, "Designing machine learning systems," 2022.

[1]