AI28 – Machine Learning UTC
M. Alaya et S. Bouzebda FISA – GI04

TP N° 3
Analyse descriptive exploratoire (AED)

Après la récupération des données, l'étape d'analyse descriptive exploratoire (AED) vient pour faire une sorte de synthèse des données. En effet, dans cette étape, on ne cherche pas à prendre des décisions mais plutôt à comprendre les données pour découvrir des tendances, caractéristiques et corrélations à examiner plus en profondeur par la suite. On cherche aussi à comprendre la signification des variables et trouver des relations entre elles. La démarche d'une AED contient, généralement, 5 étapes définies comme suit:

1. **Analyse de forme: description des variables**
   - Identifier la variable cible (target).
   - Description des variables (features): cliniques, économiques, politiques, sociales, ...
   - Dimensions du dataset (shape).
   - Types de variables: quantitatives / qualitatives / nombre de variables.
   - Visualisation des valeurs manquantes et si besoin faire un nettoyage préliminaire de ces valeurs.
2. **Analyse univariée: signification des variables**
   - Visualisation de la variables cible.
   - Histogrammes des variables quantitatives (répartition de leurs valeurs).
   - Visualisation des variables qualitatives (type object): nombre de catégorie dans chaque variable.
3. **Analyse bivariée: relation entre la variable cible et les autres variables**
   - Création des sous ensembles: par exemple selon la variables cible.
   - Relation: variable cible / variables quantitatives.
   - Relation: variables cible / variables qualitatives.
   - Construction des hypothèses.
4. **Analyse approfondie**
   - Relation: variables quantitatives / variables quantitatives.
   - Relation: variables qualitatives / variables qualitatives.
   - Construction des hypothèses.
5. **Validation des tests d'hypothèse**

1 Jeux de données

Le jeux de données contient des données anonymes provenant de patients consultés à l'Hospital Israelita Albert Einstein, à São Paulo, au Brésil, et dont les échantillons ont été recueillis pour effectuer la RT-PCR du SARS-CoV-2 et d'autres tests cliniques lors d'une visite à l'hôpital. Toutes

les données cliniques ont été rendues anonymes conformément aux meilleures pratiques et recommandations internationales. Elles ont été aussi standardisées pour avoir une moyenne de zéro et un écart-type unitaire.

On peut envisager deux tâches à faire avec ce dataset:
1. Prédire les cas confirmés de COVID-19 parmi les cas présumés. Sur la base des résultats des tests cliniques, couramment recueillis pour un cas présumé de COVID-19 lors d'une visite aux urgences, serait-il possible de prédire le résultat du test pour le SARS-COV-2 (positif/négatif)?

2. Prédire le risque d'hospitalisation en service général, en unité semi-intensive ou en unité de soins intensifs chez les cas confirmés de COVID-19. Sur la base des résultats des tests cliniques, couramment collectés parmi les cas confirmés de COVID-19 lors d'une visite aux urgences, serait-il possible de prédire quels patients devront être admis au service de surveillance, soins semi-intensifs, et soins intensifs ?

2 Analyse de forme: description des variables

1. Télécharger le fichier `covid-19.xlsx` à partir de l'espace moodle chapitre 3 et charger les données dans un dataframe `data` de Pandas.
2. Faire une vue d'ensemble des données, par exemple afficher les 5 premières observations.
3. Afficher les noms de toutes les variables et les classer en variables explicatives et variable cible (target). On remarque que la variable, numéro de patient, `Patient ID` ne sert à rien. On peut donc la supprimer avec la méthode `drop` de Pandas.
4. Créer un nouveau dataframe, `df_covid`, une copie de `data`.
5. Identifier les types de variables, utiliser `df_covid.dtypes`. Interpréter, (existence de variables qualitatives, quantitatives, ...).
6. Calculer le nombre de chaque type de variables, utiliser `df_covid.dtypes.values_counts()`. Visualiser dans un graphique en camembert les pourcentages des types de variables (utiliser la méthode `df_covid.dtypes.values_counts().plot.pie()`. On remarque que dans ce graphique qu'on affiche les types des variables: type `float` pour les variables quantitatives (continues), `object` (qualitatives).
7. **Visualisation des valeurs manquantes**
   (a) On rappelle que la méthode `isna()` du Pandas sert à vérifier si un dataframe contient une valeur `NaN`. Exécuter: `df_covid.isna()[:5]`. On remarque que çà retourne un dataframe booléen `(True, False)`.
   (b) Utiliser la méthode `heatmap` de Seaborn pour afficher l'ensemble des valeurs manquantes de `df_covid`. Interpréter.
   ```
   plt.figure(figsize=(26, 12))
   sns.heatmap(df_covidid.isna(), cbar=False)
   plt.show()
   ```
   (c) Mesurer le pourcentage des valeurs manquantes de chacune des variables. Exécuter: `(df_covid.isna().sum() / df_covid.shape[0]) * 100`. Interpréter.
   (d) Trier les pourcentages des valeurs manquantes dans l'ordre croissant: `miss_rates = miss_rates.sort_values(ascending=True)`.
8. Dans la suite de notre étude exploratoire, nous allons supprimer les variables ayant un pourcentage de valeurs manquantes qui dépasse 90%. Utiliser une indexation booléenne pour

supprimer ces variables. Exécuter:

```
df_covid.columns[miss_rates < 90]
df_covid = df_covid.columns[miss_rates < 90] # Réécriture de df_covid
```

Afficher de nouveau la carte heatmap des valeurs manquantes et les pourcentages des valeurs manquantes triés dans un ordre croissant. Interpréter.

3 Analyse univariée: signification des variables

3.1 Variable cible

1. Afficher variable cible, son type, et ses modalités s'il s'agit d'un type qualitatif.
2. On rappelle qu'une variable du dataframe `df_covid` a une structure de Series et donc on peut utiliser les mêmes méthodes d'un dataframe. Avec la méthode `value_counts()`, calculer le nombre de patients testés positifs et négatifs au covid.
3. Afficher sous forme de pourcentage les cas positifs et négatifs. Utiliser la méthode `value_counts(normalize=True)`. Interpréter.

3.2 Histogrammes des variables quantitatives

4. Afficher les noms de toutes les variables continues, utiliser la méthode `select_dtypes('float')` de Pandas.
5. Écrire une fonction qui utilise la méthode `distplot` de Seaborn et retourne les graphiques des distributions des variables continues.
6. Calculer le coefficient d'assymètrie de deux variables `Hemtocrit` et `Red blood cell distirbution with (RDW)`.
7. Interpréter le graphique pour la variable `Patient age quantile`.

3.3 Visualisation des variables qualitatives

8. Afficher les différentes modalités (catégories) de chaque variable qualitative. Exécuter le code suivant:

```
for col in df_covid.select_dtypes('object'):
    print(f'{col :-<70} {df_covid[col].unique()}') # créer un système de marge
```

9. Visualiser dans des graphiques en camembert les catégories de chacune des variables, (utiliser la méthode `df_covid.dtypes.values_counts().plot.pie()`. Interpréter.

4 Analyse bivariée: relation entre la variable cible / autres variables)

4.1 Création des sous-ensembles

1. Créer deux dataframes `df_covid_pos` et `df_covid_neg` correspondent aux patients testés positifs et négatifs, respectivement. Afficher les dimensions (shapes) de chaque nouveau dataframe.
2. Créer deux groupes de variables: on remarque que les tests cliniques correspondent aux test viraux et taux sanguins selon le pourcentage des valeurs manquantes. Exécuter:

```
miss_rates = df_covid.isna().sum() / len(df_covid)
 bloodtest_features = df_covid.columns[(miss_rates < 0.9) & (miss_rates >0.88)]
 viraltest_features = df_covid.columns[(miss_rates < 0.88) & (miss_rates >0.75)]
```

3. Afficher ces deux groupes.

4.2 Relation: variable cible / variables quantitatives (taux sanguins)

4. Tracer les distributions conditionnelles des variables de groupe taux sanguins sachant la variable cible patients soient testés positifs et négatifs au covid. Utiliser la méthode `distplt` de seaborn.
   les Nous remarquons que les taux des Platelets, Lymphocytes et Leukocytes chez les individus testés positifs au Corona virus sont différents de ceux des personnes testées négatives. Pour bien valider ces constations, il va falloir les transformer en tests d'hypothèse et étudier leurs robustesses, c'est à dire vérifier la signification de chaque test (voir la section 5 sur la validation des tests).
5. Tracer la distribution conditionnelle de la variable age sachant la cible. Interpréter.

4.3 Relation: variable cible / variables quantitatives (tests viraux)

6. Nous constatons que la variable cible et les variables des tests viraux sont de type catégoriel. Pour tracer la distribution des tests viraux sachant la variable cible on utilise *tableaux de contingence (cross tabulation, cross-tab)*, qui sont des méthodes de représentation de données issues d'un comptage permettant d'estimer la dépendance entre deux caractères. Ces tableaux consistent à croiser deux caractères en dénombrant l'effectif correspondant à leur conjonction. Exécuter `pd.crosstab(df_covid['SARS-Cov-2 exam result'], df_covid['Influenza A'])`.
   Utiliser la fonction `crosstab` de Pandas pour tracer les tables de contingence:

```
for col in  viraltest_features:
    plt.figure()
    sns.heatmap(pd.crosstab(df_covid['SARS-Cov-2 exam result'],
                            df_covid[col]), annot=True, fmt='d')
    plt.show()
```

   Interpréter.

5 Analyse approfondie

5.1 Relation: variables quantitatives / variables quantitatives

1. Utiliser la fonction `pairplot` de Seaborn pour tracer les graphiques des distributions bivariées par paires de variables taux sanguins. Interpréter.
2. Calculer la matrice de corrélations entre les variables taux sanguins. Rappelons que plus la corrélation est proche de 1 (respectivement -1), plus les deux variables évoluent positivement (respectivement négativement) les unes avec les autres, c'est à dire quand l'une augmente l'autre augemnte aussi (respectivement quand l'une augmente l'autre diminue).
3. Utiliser la méthode `clustermap` de Seaborn qui permet de rassembler directement les variables et les organise sous forme de cluster ayant des fortes corrélations linéaires.
4. Étudier s'il existe une relation linéaire entre la variable âge et les variables taux sanguins, utiliser la corrélation âge /taux sanguins.

`df_covid.corr()['Patient age quantile'].sort_values()`. Interpréter.

5.2 Relation: variables qualitatives / variables qualitatives

5. Supprimer les variables `Influenza A, rapid test` et `Influenza B, rapid test` car elles ont une très mauvaise sensibilité donc ces tests sont très peu fiables.
6. Tracer un graphique qui montrent de combien chaque patient est testé positif à une maladie. Exécuter
```
np.sum(df_covid[viraltest_features] == 'detected', axis=1).plot()
plt.show()
```

5.3 Relation: variables qualitatives / variables quantitatives

8. Nous allons créer une nouvelle variable `etre_malade` qui permet de vérifier si un patient est atteint d'au minimum 2 maladies: `df_covid['etre_malade'] = np.sum(df_covid[bloodtest_features] == 'detected', axis=1) > 1.`
9. Créer deux dataframes `df_malade` et `df_non_malade` pour deux groupes (patients malades et non malades)
10. Tracer les distributions des variables taux sanguins sachant que les patients soient malades ou non malades. Remarquer que:
les taux des Lymphocytes, Leukocytes, MCH, MCV sont différents entre les personnes malades et non malades. Pour rendre ces constations robustes il va falloir formuler des tests d'hypothèses pour comparer les moyennes des ces taux.

5.4 Relation hospitalisation / être malade

A présent, nous nous intéressons aux trois variables `Patient addmited to regular ward (1=yes, 0=no)`, `Patient addmited to semi-intensive unit (1=yes, 0=no)` et `Patient addmited to intensive care unit (1=yes, 0=no)` qui indiquent l'état d'hospitalisation des individus soient: *patient admis au service de surveillance, soins semi-intensive,* et *soins intensifs*.

11. Utiliser la fonction `hospitalisaiton` ci-dessous pour créer une nouvelle variable `etre_hospitalise` (pensez à utiliser la fonction `map` ou `apply`).

```
def hospitalisation(dataframe=df_covid):
    if dataframe['Patient addmited to regular ward (1=yes, 0=no)'] == 1:
        return 'surveillance'
    elif dataframe['Patient addmited to semi-intensive unit (1=yes, 0=no)'] == 1:
        return 'semi-intensive'
    elif dataframe['Patient addmited to intensive care unit (1=yes, 0=no)'] == 1:
        return 'soins intensifs'
    else:
        return 'autre'
```

12. Tracer la distribution conditionnelles des variables taux sanguins des patients sachant leurs états d'hospitalisation. Interpréter.

6 Validation des hypothèses

Hypothèse 1. *Les patients testés positifs au covid-19 ont des taux de Leukocytes, Monocytes, Patletes significativement différents par rapport aux patients testés négatifs.*

Hypothèse 2. *Les patients testés positifs d'une maladie quelconque ont des taux Lymphocytes, Leukocytes, MCH, MCV significativement différents para rapport aux patients non malades.*

1. Pour l'**Hypothèse 1**, on construit:

- (H_0) : Les taux moyens sont égaux chez les patients testés positifs et négatifs.
- (H_1) : Les taux moyens sont différents chez les patients testés positifs négatifs.

Pour tester cette hypothèse, nous allons utiliser un *test de Student*. Ce test permet de vérifier si la moyenne entre deux distributions est significativement différente. Une condition pour pouvoir appliquer ce test est que les deux échantillons doivent avoir la même taille, en particulier dans notre cas: le nombre de patients testés positifs doit égal au nombre des patients testés négatifs. Cependant, cette condition n'est pas vérifié. Par conséquent, nous allons faire un sous-échantillonnage, c'est à dire on choisit aléatoirement un sous-échantillon de `df_covid_pos` de même taille que celui de l'échantillon `df_covid_neg`.

2. Avec la le module `ttest_ind` de Scipy tester la comparaison des moyennes de taux sanguins chez les patients testés positifs et négatifs. Utiliser le code suivant:

```
from scipy.stats import ttest_ind # Importer le ttest_ind de scipy
n_samples_pos, _ = df_covid.shape
df_covid_neg.sample(n_samplesPos)
df_covid_neg_samples = df_covid_neg.sample(n_samples_pos)
```

   On choisit un seuil $\alpha = 5\%$. On vérifie si la p-valeur issue du test est inférieur à α et si tel est le cas on rejette (H_0) sinon on retourne 0.

```
def test_comparison(col):
    alpha = 0.05# 5%
    stat, p = ttest_ind(df_covid_neg.sample(n_samples_pos)[col].dropna(),
                        df_covid_pos[col].dropna())
    if p < alpha:
        return 'H0 Rejetée'
    else:
        return 0
```

   On peut afficher les résultats du test:

```
for col in cols_taux_sanguins:
    print(f'{col:-<70} {t_comparison(col)}')
```

3.