Prédiction du Covid 19

1. Contexte :

La prédiction du covid19 est importante pour atténuer sa propagation. Actuellement, la plupart des méthodes de diagnostic impliquent l'échantillonnage des fluides nasaux, de la salive ou du sang suivi de tests à base d'acide nucléique ou le dépistage sérologique sanguin des infections passées. Les diagnostics basés sur les acides nucléiques peuvent nécessiter des échantillons prélevés plusieurs jours après l'exposition pour une détection positive sans ambiguïté. De plus, ils ne peuvent pas être mis en œuvre systématiquement à faible coût et sont limités par les pénuries émergentes de réactifs clés.

L'une des solutions pour palier à ce problème est l'utilisation de **l'Analyse Prédictive**.
L'objectif de ce devoir étant de développer **un modèle de prédiction du Covid 19 à partir de 4 paramètres : L'âge, le sexe, La saturation en oxygène SPO2 et la Température**.

2. Méthodologie :

Le but ici est la classification, c'est-à-dire: étant donné un ensemble de données d'entrée avec des labels de classe (Covid ou non Covid), Vous devez développer un modèle pour prédire avec précision la classe d'une nouvelle donnée d'entrée inconnue.

2.1 Dataset :

Le dataset à utiliser est accessible via la plateforme GitHub (https://github.com/ieee8023/covid-chestxray-dataset/blob/master/metadata.csv).

2.2 Analyse de données :

A- Importer le dataset dans votre environnement Jupiter Notebook.

In [1]:

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from IPython.core.pylabtools import figsize
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

In [2]:

```
%matplotlib inline
figsize(14, 7)
sns.set(style='ticks')
pd.options.mode.chained_assignment = None
```

In [3]:

```
df = pd.read_csv("dataset.csv")
df.head()
```

Out[3]:

	patientid	offset	sex	age	finding	RT_PCR_positive	survival	intubated	intu
0	2	0.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	
1	2	3.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	
2	2	5.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	
3	2	6.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	
4	4	0.0	F	52.0	Pneumonia/Viral/COVID-19	Y	NaN	N	

5 rows × 30 columns

B- Générer un nouveau dataset en ne retenant que les variables d'intérêts (âge, sexe, saturation en oxygène SPO2 et Température. Afficher un descriptif de ce nouveau dataset.

```
In [4]: df.columns
```

```
Out[4]: Index(['patientid', 'offset', 'sex', 'age', 'finding', 'RT_PCR_positive',
               'survival', 'intubated', 'intubation_present', 'went_icu', 'in_icu',
               'needed_supplemental_O2', 'extubated', 'temperature', 'pO2_saturatio
        n',
               'leukocyte_count', 'neutrophil_count', 'lymphocyte_count', 'view',
               'modality', 'date', 'location', 'folder', 'filename', 'doi', 'url',
               'license', 'clinical_notes', 'other_notes', 'Unnamed: 29'],
              dtype='object')
```

```
In [5]: new_df = df[['age', 'sex', 'pO2_saturation', 'temperature', 'finding']]
```

C- Visualiser un extrait des enregistrements.

```
In [6]: new_df.head()
```

Out[6]:

	age	sex	pO2_saturation	temperature	finding
0	65.0	M	NaN	NaN	Pneumonia/Viral/COVID-19
1	65.0	M	NaN	NaN	Pneumonia/Viral/COVID-19
2	65.0	M	NaN	NaN	Pneumonia/Viral/COVID-19
3	65.0	M	NaN	NaN	Pneumonia/Viral/COVID-19
4	52.0	F	NaN	NaN	Pneumonia/Viral/COVID-19

D- Procéder au nettoyage des données. Enlever les data manquantes.

dans cette étape je supprimer tous les lignes qui a plus de 3 valeurs null.

```
In [7]: len(new_df)
```

```
Out[7]: 950
```

```
In [8]: threshold = 0.75
        data_1 = new_df.loc[new_df.isnull().mean(axis=1) < threshold]
```

```
In [9]: len(data_1)
```

```
Out[9]: 893
```

Pour la colonne âge

```
In [10]: data_1["age"].isnull().sum()
```

```
Out[10]: 180
```

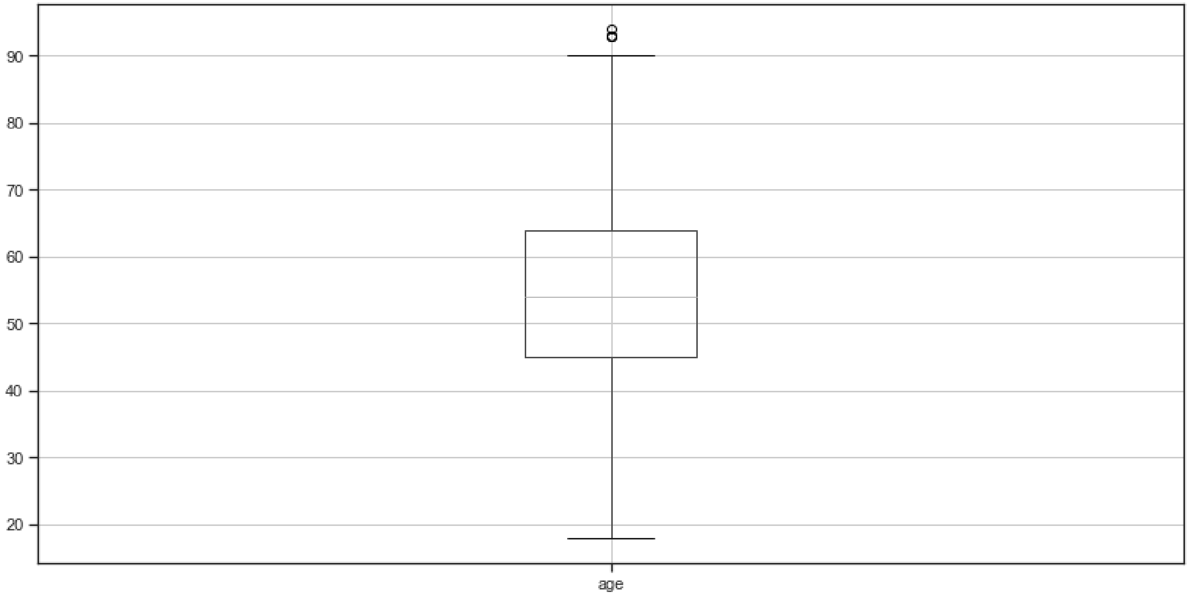
```
In [11]: data_1["age"] = data_1["age"].fillna(data_1["age"].median())
```

```
In [12]: data_1["age"].max()
```

```
Out[12]: 94.0
```

gestion des valeurs aberrantes

```
In [13]: data_1.boxplot(column=["age"])
         pass
```



```
In [14]: data_1['age'] = pd.cut(data_1['age'],
                                bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
                                labels=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

```
In [15]: data_1['age'].astype(int).dtype
```

```
Out[15]: dtype('int32')
```

Pour la colonne sex

```
In [16]: data_2 = data_1.copy(deep=True)
```

```
In [17]: data_2["sex"].isnull().sum()
```

```
Out[17]: 23
```

In [18]:
```
data_2['sex'] = data_2['sex'].fillna(data_2['sex'].value_counts().index[0])
```

In [19]:
```
data_2
```

Out[19]:

	age	sex	pO2_saturation	temperature	finding
0	70	M	NaN	NaN	Pneumonia/Viral/COVID-19
1	70	M	NaN	NaN	Pneumonia/Viral/COVID-19
2	70	M	NaN	NaN	Pneumonia/Viral/COVID-19
3	70	M	NaN	NaN	Pneumonia/Viral/COVID-19
4	60	F	NaN	NaN	Pneumonia/Viral/COVID-19
...	...	...	...	...	...
945	40	F	NaN	NaN	Pneumonia
946	40	F	NaN	NaN	Pneumonia
947	30	M	NaN	NaN	Pneumonia
948	50	M	NaN	NaN	Pneumonia
949	50	M	NaN	NaN	Pneumonia

893 rows × 5 columns

Pour la colonne `p02_saturation`

In [20]:
```
data_3 = data_2.copy(deep=True)
```

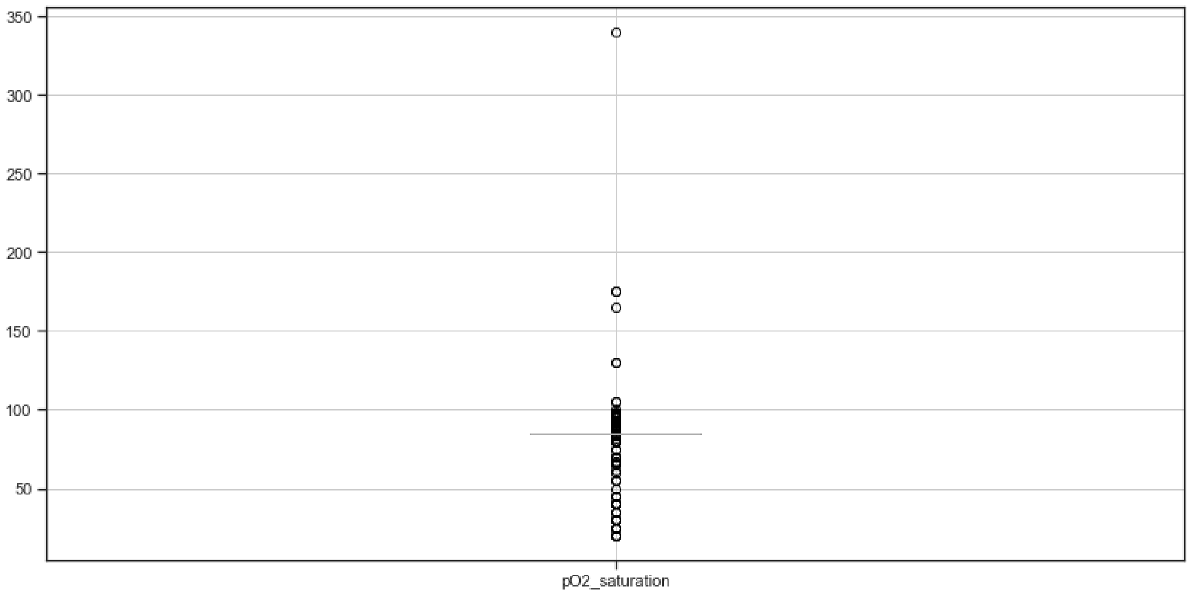
In [21]:
```
data_3["p02_saturation"].isnull().sum()
```

Out[21]:
```
774
```

In [22]:
```
data_3["p02_saturation"] = data_3["p02_saturation"].fillna(data_3["p02_saturat
ion"].median())
```

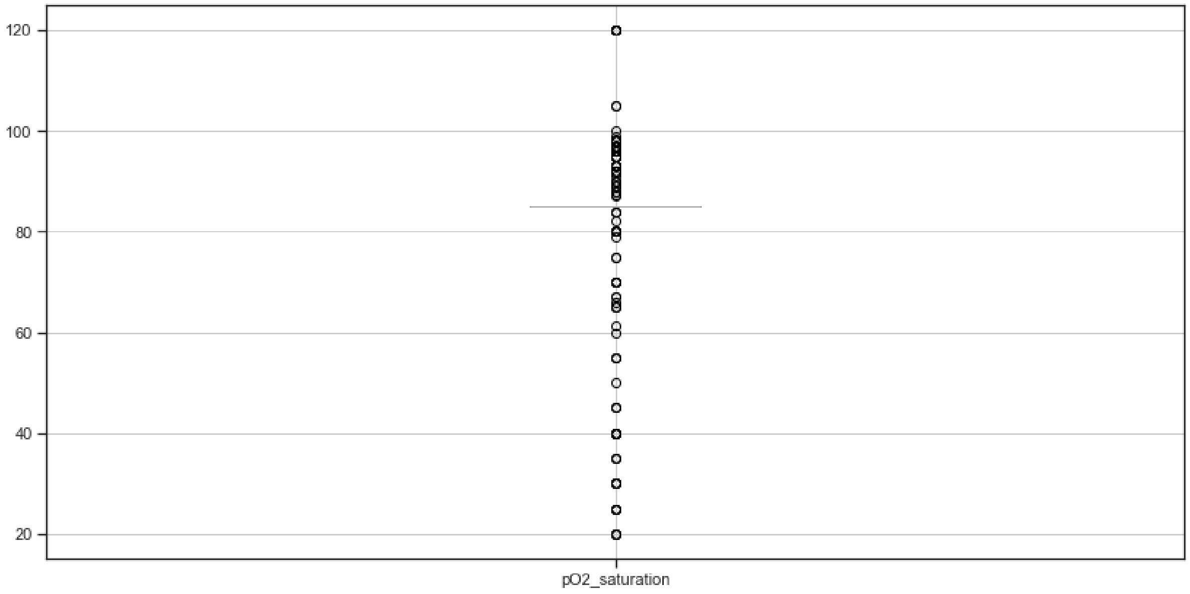
gestion des valeurs aberrantes

In [23]:
```
data_3.boxplot(column=["pO2_saturation"])
pass
```



In [24]:
```
threshold = 120
data_3.loc[data_3["pO2_saturation"] > threshold, "pO2_saturation"] = threshold
```

In [25]:
```
data_3.boxplot(column=["pO2_saturation"])
pass
```



Pour la colonne *temperature*

In [26]:
```
data_4 = data_3.copy(deep=True)
```

In [27]:

```
data_4["temperature"].isnull().sum()
```

Out[27]:

```
815
```

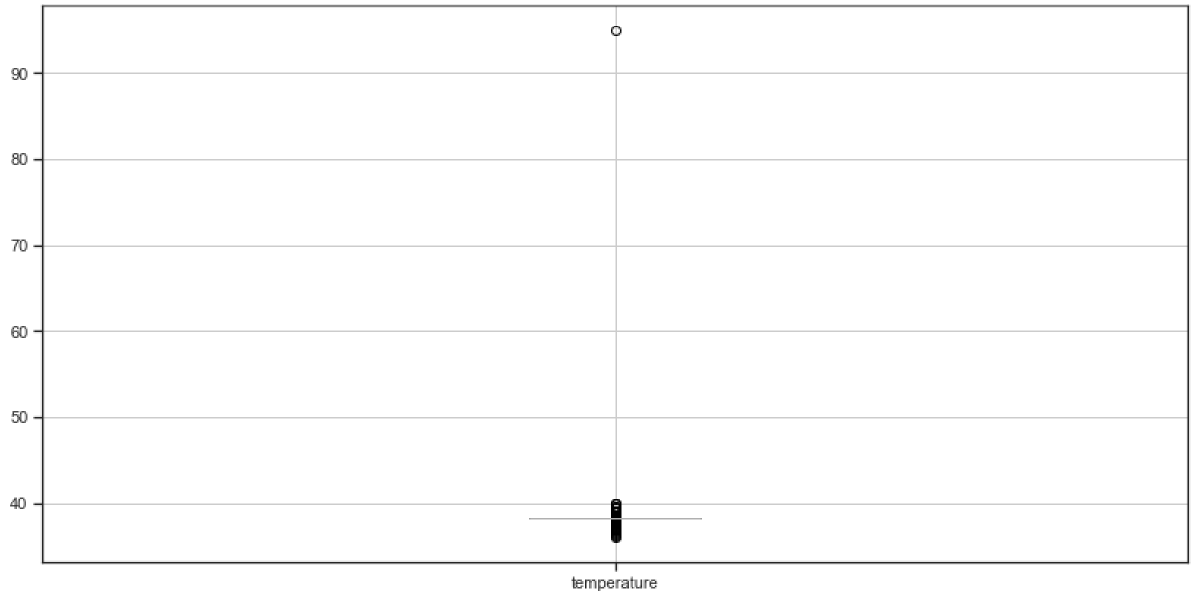
In [28]:

```
data_4["temperature"] = data_4["temperature"].fillna(data_4["temperature"].med
ian())
```

gestion des valeurs aberrantes

In [29]:

```
data_4.boxplot(column=["temperature"])
pass
```

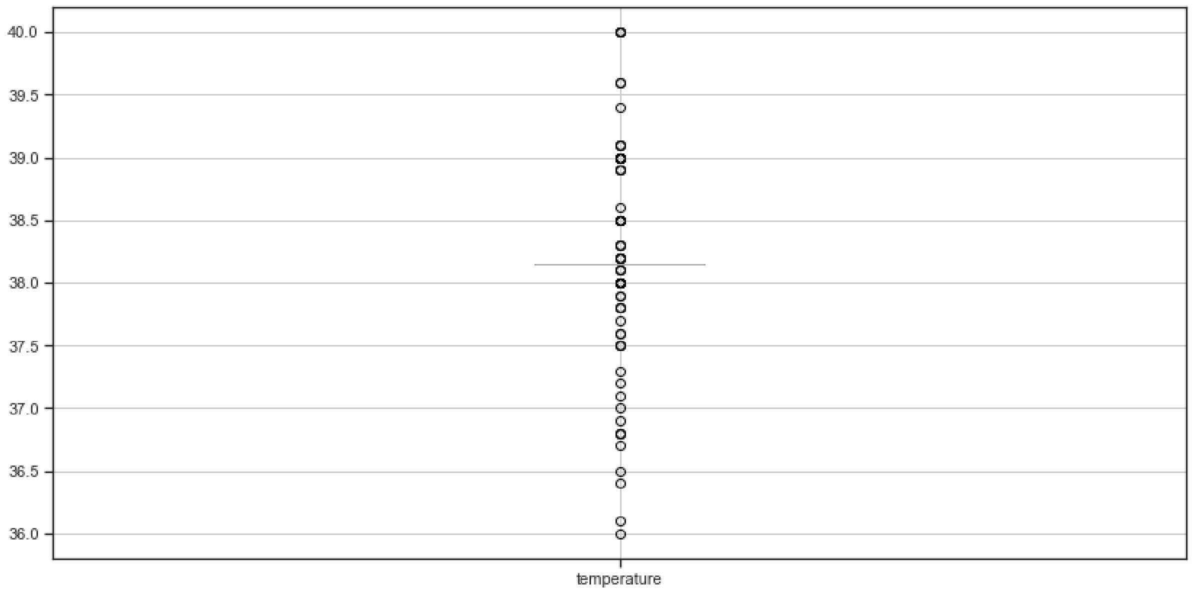


In [30]:

```
threshold = 40
data_4.loc[data_3["temperature"] > threshold, "temperature"] = threshold
```

In [31]:

```
data_4.boxplot(column=["temperature"])
pass
```



Pour la colonne finding

In [32]:

```
data_5 = data_4.copy(deep=True)
```

In [33]:

```
data_5 = data_5.loc[~data_5["finding"].str.contains("Unknown")]
```

In [34]:
```
data_5["finding"].value_counts()
```

Out[34]:
```
Pneumonia/Viral/COVID-19                    542
todo                                         82
Pneumonia                                    80
Pneumonia/Fungal/Pneumocystis                27
No Finding                                   22
Pneumonia/Bacterial/Streptococcus            22
Tuberculosis                                 18
Pneumonia/Viral/SARS                         16
Pneumonia/Lipoid                             13
Pneumonia/Bacterial/Mycoplasma               11
Pneumonia/Bacterial/Klebsiella               10
Pneumonia/Bacterial/Legionella               10
Pneumonia/Bacterial/Nocardia                  8
Pneumonia/Viral/Varicella                     6
Pneumonia/Viral/Influenza                     5
Pneumonia/Bacterial/E.Coli                    4
Pneumonia/Bacterial                           4
Pneumonia/Bacterial/Chlamydophila             3
Pneumonia/Viral/Herpes                        3
Pneumonia/Viral/Influenza/H1N1                2
Pneumonia/Fungal/Aspergillosis                2
Pneumonia/Aspiration                          1
Pneumonia/Bacterial/Staphylococcus/MRSA       1
Name: finding, dtype: int64
```

In [35]:
```
def split_finding(x):
    if "COVID-19" in x or "todo" in x:
        return 1
    return 0
```

In [36]:
```
data_5["covid-19"] = data_5["finding"].apply(lambda x: split_finding(x))
```

In [37]:
```
data_5 = data_5.drop(["finding"], axis=1)
```

In [38]:
```
data_5
```

Out[38]:

	age	sex	pO2_saturation	temperature	covid-19
0	70	M	85.0	38.15	1
1	70	M	85.0	38.15	1
2	70	M	85.0	38.15	1
3	70	M	85.0	38.15	1
4	60	F	85.0	38.15	1
...	...	...	...	...	...
945	40	F	85.0	38.15	0
946	40	F	85.0	38.15	0
947	30	M	85.0	38.15	0
948	50	M	85.0	38.15	0
949	50	M	85.0	38.15	0

892 rows × 5 columns

La correction des types des donnée

In [39]:
```
data_5.dtypes
```

Out[39]:
```
age                 category
sex                   object
pO2_saturation       float64
temperature          float64
covid-19               int64
dtype: object
```

In [40]:
```
data_5.astype({'age' : 'int32',
               'sex' : 'object',
               'pO2_saturation' : 'float64',
               'temperature': 'float64',
               'covid-19' : 'object'
              }).dtypes
```

Out[40]:
```
age                   int32
sex                  object
pO2_saturation      float64
temperature         float64
covid-19             object
dtype: object
```

E- Afficher un tableau décrivant le dataset après nettoyage.

In [41]:
```
data = data_5.copy(deep=True)
data
```

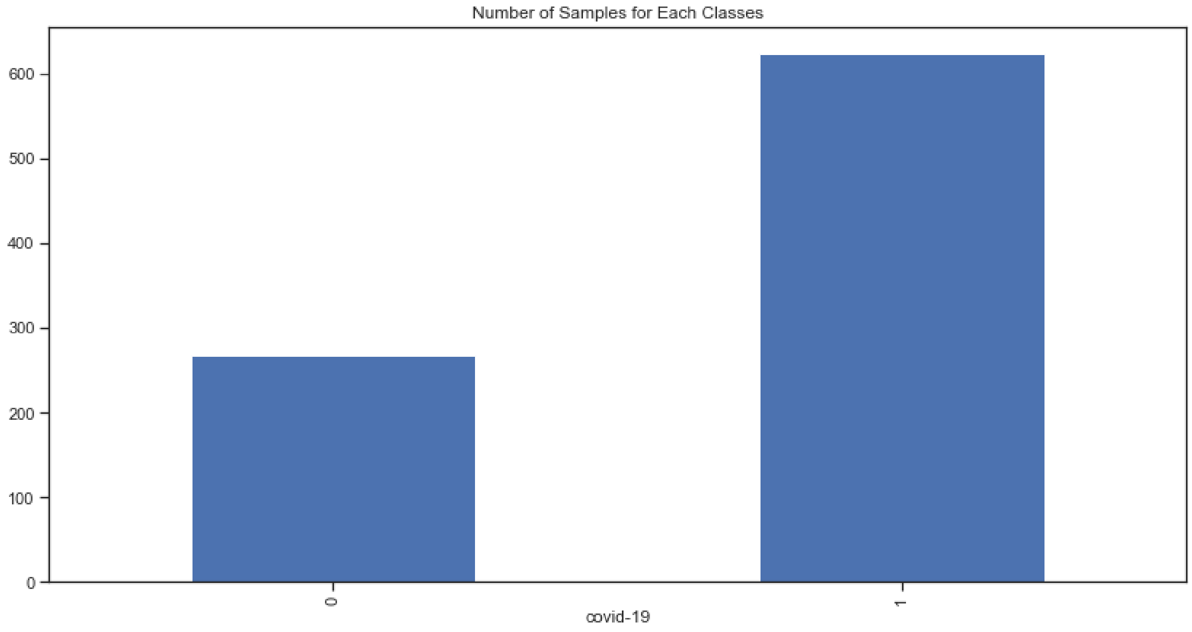
Out[41]:

	age	sex	pO2_saturation	temperature	covid-19
0	70	M	85.0	38.15	1
1	70	M	85.0	38.15	1
2	70	M	85.0	38.15	1
3	70	M	85.0	38.15	1
4	60	F	85.0	38.15	1
...	...	...	...	...	...
945	40	F	85.0	38.15	0
946	40	F	85.0	38.15	0
947	30	M	85.0	38.15	0
948	50	M	85.0	38.15	0
949	50	M	85.0	38.15	0

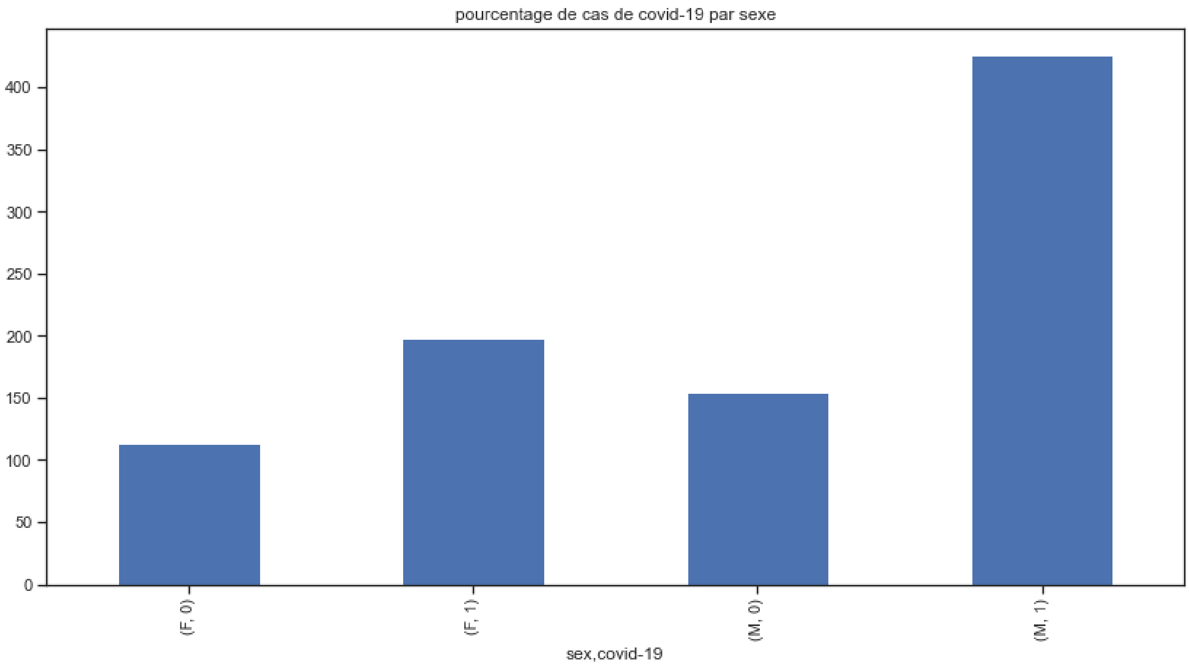
892 rows × 5 columns

F- Générer 4 plots pour visualiser convenablement chacune des variables dans ce nouveau dataset.

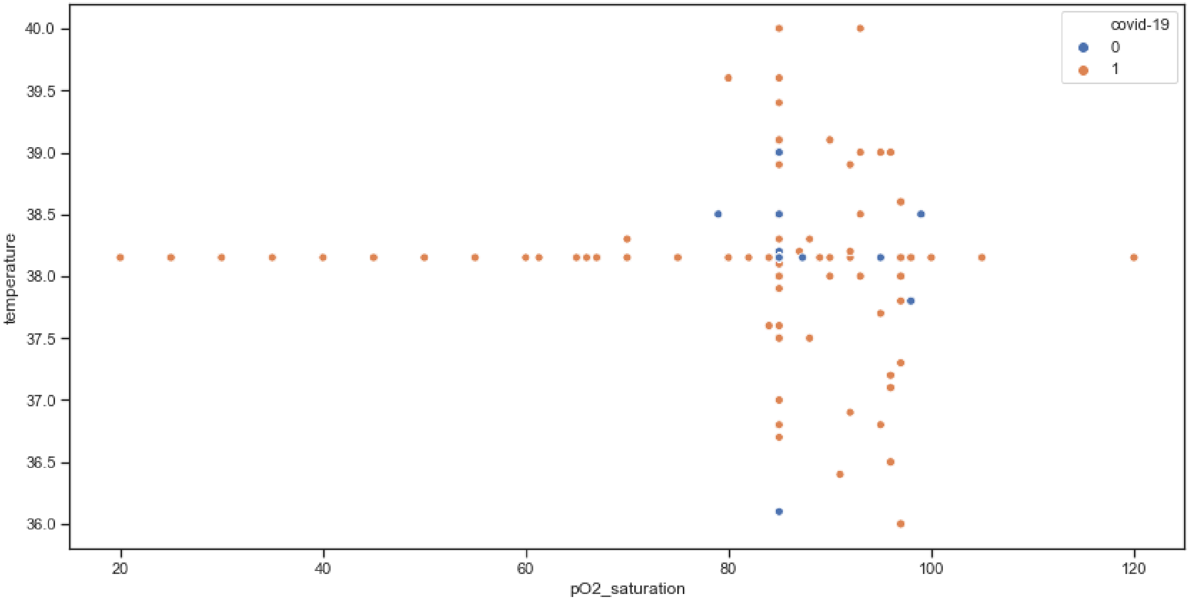
In [42]:
```
data.groupby(['covid-19']).size().plot.bar(title="Number of Samples for Each C
lasses")
pass
```



In [43]:
```
data.groupby(['sex', 'covid-19']).size().plot.bar(title="pourcentage de cas de covid-19 par sexe")
pass
```



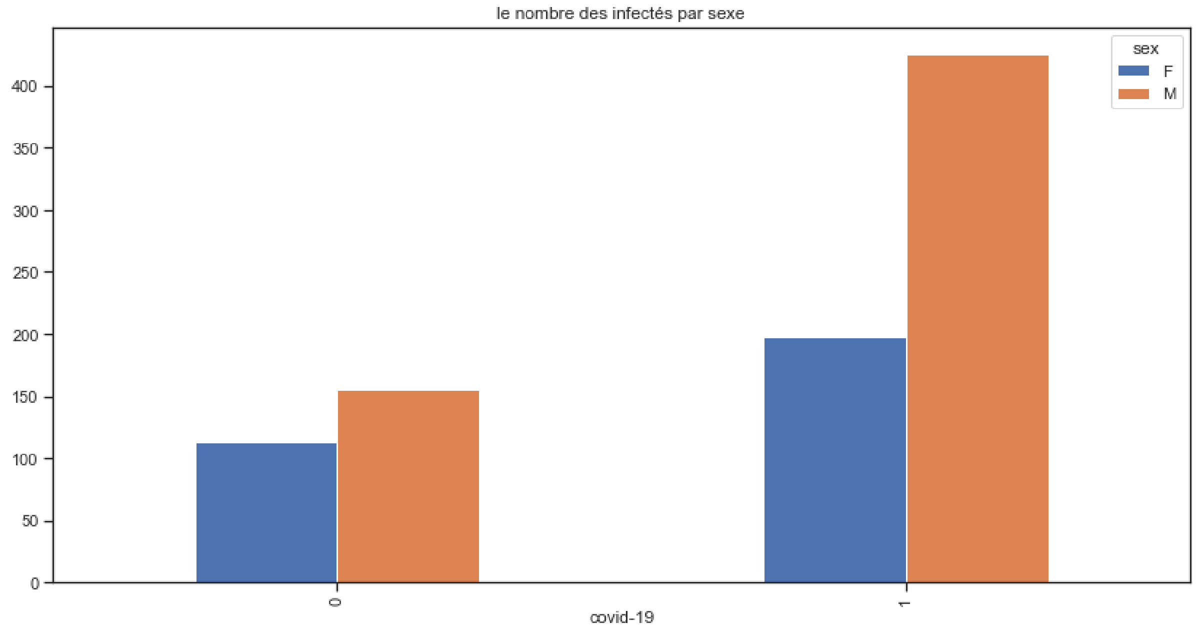
In [44]:
```
sns.scatterplot(data=data[["pO2_saturation", "temperature", "covid-19"]], x="pO2_saturation", y="temperature", hue="covid-19")
pass
```



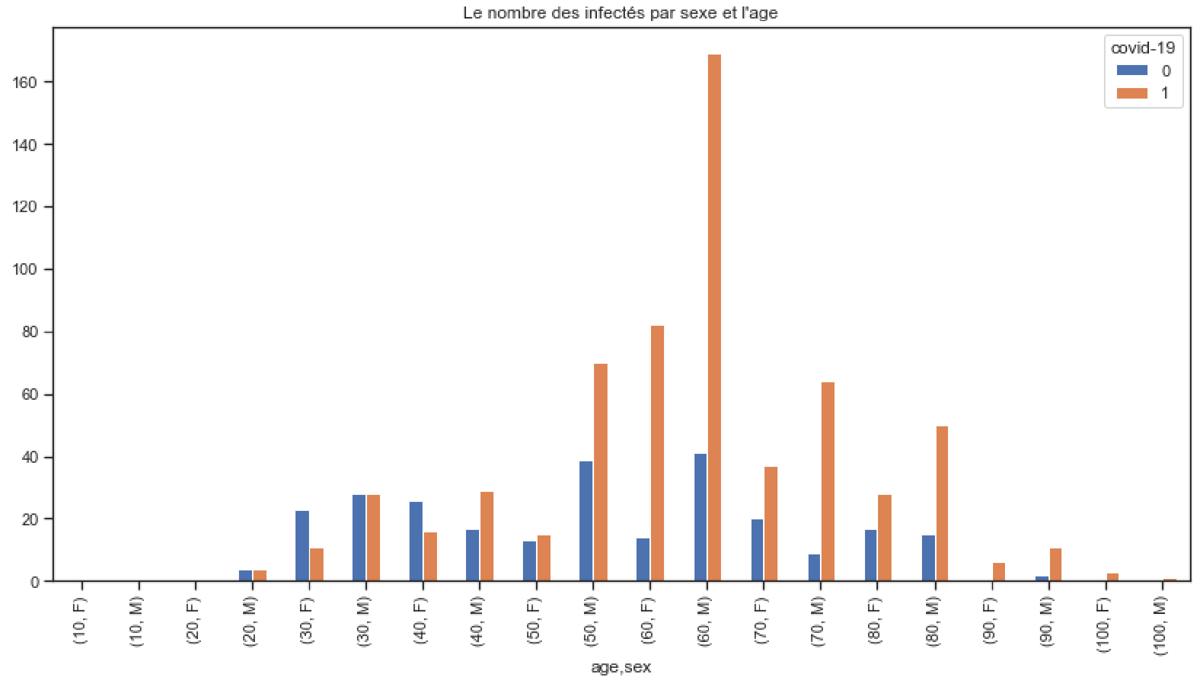
In [45]:
```
data.groupby(["covid-19", "sex"]).size()
```

Out[45]:
```
covid-19  sex
0         F      113
          M      155
1         F      198
          M      426
dtype: int64
```

```
In [46]: data.groupby(["covid-19", "sex"]).size().unstack().plot.bar(title="le nombre des infectés par sexe")
         pass
```



```
In [47]: data.groupby(["age", "sex", "covid-19"]).size().unstack().plot.bar(title="Le nombre des infectés par sexe et l'age")
         pass
```



G- Construire 5 modèles de prédictions du covid 19

In [48]:
```
def gender_bin(x):
    if x == "M":
        return 1
    elif x == "F":
        return 0

data["sex"] = data["sex"].apply(lambda x: gender_bin(x))

data.to_csv("dataset_cleaned.csv", index=False)
```

In [49]:
```
data
```

Out[49]:

	age	sex	pO2_saturation	temperature	covid-19
0	70	1	85.0	38.15	1
1	70	1	85.0	38.15	1
2	70	1	85.0	38.15	1
3	70	1	85.0	38.15	1
4	60	0	85.0	38.15	1
...	...	...	...	...	...
945	40	0	85.0	38.15	0
946	40	0	85.0	38.15	0
947	30	1	85.0	38.15	0
948	50	1	85.0	38.15	0
949	50	1	85.0	38.15	0

892 rows × 5 columns

In [50]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, 4].values
```

In [51]:
```
X
```

Out[51]:
```
array([[70.  ,  1.  , 85.  , 38.15],
       [70.  ,  1.  , 85.  , 38.15],
       [70.  ,  1.  , 85.  , 38.15],
       ...,
       [30.  ,  1.  , 85.  , 38.15],
       [50.  ,  1.  , 85.  , 38.15],
       [50.  ,  1.  , 85.  , 38.15]])
```

In [52]:
```
y[ : 10]
```

Out[52]:
```
array([1, 1, 1, 1, 1, 1, 0, 0, 0, 0], dtype=int64)
```

Train Test Split

In [53]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=109)
```

In [54]:
```
print("Number of examples in training set ",len(y_train))
print("Number of examples in test set", len(y_test))
```

```
Number of examples in training set  669
Number of examples in test set 223
```

Feature Scaling

In [55]:
```
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

1- K-Nearest Neighbors

In [56]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier_knn = KNeighborsClassifier(n_neighbors=58)
classifier_knn.fit(X_train, y_train)

y_knn_pred = classifier_knn.predict(X_test)
```

In [57]:
```
print(confusion_matrix(y_test, y_knn_pred))
```

```
[[ 26  46]
 [  7 144]]
```

In [58]:
```
print(classification_report(y_test, y_knn_pred, target_names=("covid-19", "non-covid-19")))
```

```
              precision    recall  f1-score   support

    covid-19       0.79      0.36      0.50        72
non-covid-19       0.76      0.95      0.84       151

   micro avg       0.76      0.76      0.76       223
   macro avg       0.77      0.66      0.67       223
weighted avg       0.77      0.76      0.73       223
```

In [59]:

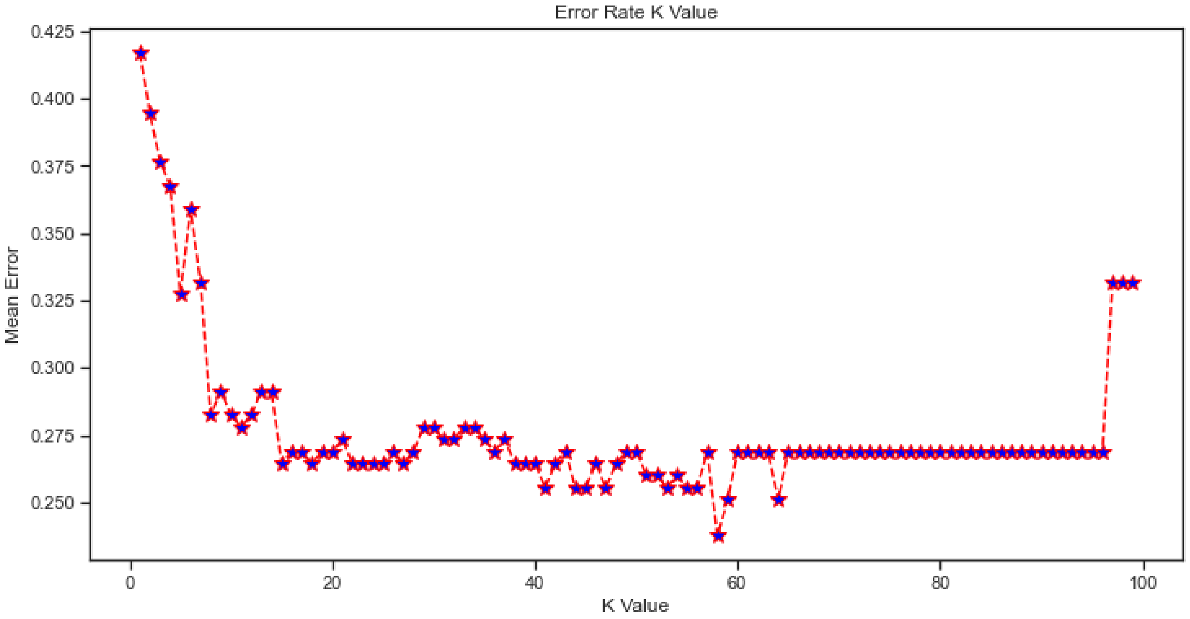
```
error = []

# Calculating error for K values between 1 and 100
for i in range(1, 100):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))

print("L'optimum K est {}".format(error.index(min(error))))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 100), error, color='red', linestyle='dashed', marker='*', ma
rkerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
pass
```

```
L'optimum K est 57
```



2- Support Vector Machines

In [60]:

```
from sklearn import svm

classifier_svm = svm.SVC(kernel='rbf', gamma=2)
classifier_svm.fit(X_train, y_train)

y_svm_pred = classifier_svm.predict(X_test)
```

In [61]:

```
print(confusion_matrix(y_test, y_svm_pred))
```

```
[[ 18  54]
 [  6 145]]
```

In [62]:

```
print(classification_report(y_test, y_svm_pred, target_names=("covid-19", "non-covid-19")))
```

```
              precision    recall  f1-score   support

    covid-19       0.75      0.25      0.38        72
non-covid-19       0.73      0.96      0.83       151

   micro avg       0.73      0.73      0.73       223
   macro avg       0.74      0.61      0.60       223
weighted avg       0.74      0.73      0.68       223
```

In [63]:

```
for kernel in ('linear', 'poly', 'rbf'):
    clf = svm.SVC(kernel=kernel, gamma=2)
    clf.fit(X_train, y_train)
    clf_accu = clf.predict(X_test)
    print("Confusion matrix for {} kernel :".format(kernel))
    print(confusion_matrix(y_test, clf_accu))
```

```
Confusion matrix for linear kernel :
[[  0  72]
 [  0 151]]
Confusion matrix for poly kernel :
[[  6  66]
 [  2 149]]
Confusion matrix for rbf kernel :
[[ 18  54]
 [  6 145]]
```

3- Decision Tree

In [64]:

```
from sklearn.tree import DecisionTreeClassifier

classifier_dt = DecisionTreeClassifier()
classifier_dt = classifier_dt.fit(X_train,y_train)

y_dt_pred = classifier_dt.predict(X_test)
```

In [65]:

```
print(confusion_matrix(y_test, y_dt_pred))
```

```
[[ 27  45]
 [ 13 138]]
```

```
In [66]: print(classification_report(y_test, y_dt_pred, target_names=("covid-19", "non-
         covid-19")))
```

```
                precision    recall  f1-score   support

    covid-19        0.68      0.38      0.48        72
non-covid-19        0.75      0.91      0.83       151

   micro avg        0.74      0.74      0.74       223
   macro avg        0.71      0.64      0.65       223
weighted avg        0.73      0.74      0.72       223
```

4- Random Forests

```
In [67]: from sklearn.ensemble import RandomForestClassifier

         classifier_rfc=RandomForestClassifier(n_estimators=100)
         classifier_rfc.fit(X_train, y_train)

         y_rfc_pred=clf.predict(X_test)
```

```
In [68]: print(confusion_matrix(y_test, y_rfc_pred))
```

```
[[ 18  54]
 [  6 145]]
```

```
In [69]: print(classification_report(y_test, y_rfc_pred, target_names=("covid-19", "non
         -covid-19")))
```

```
                precision    recall  f1-score   support

    covid-19        0.75      0.25      0.38        72
non-covid-19        0.73      0.96      0.83       151

   micro avg        0.73      0.73      0.73       223
   macro avg        0.74      0.61      0.60       223
weighted avg        0.74      0.73      0.68       223
```

La recherche des features importantes

```
In [70]: feature_imp = pd.Series(classifier_rfc.feature_importances_, index=data.column
         s[ : -1]).sort_values(ascending=False)
         feature_imp
```

```
Out[70]: age               0.458199
         temperature       0.244383
         pO2_saturation    0.215462
         sex               0.081955
         dtype: float64
```

5- Logistic Regression

In [71]:

```
from sklearn.linear_model import LogisticRegression

classifier_lr = LogisticRegression(solver='lbfgs')
classifier_lr.fit(X_train, y_train)

y_lr_pred=clf.predict(X_test)
```

In [72]:

```
print(confusion_matrix(y_test, y_lr_pred))
```

```
[[ 18  54]
 [  6 145]]
```

In [73]:

```
print(classification_report(y_test, y_lr_pred, target_names=("covid-19", "non-covid-19")))
```

```
              precision    recall  f1-score   support

    covid-19       0.75      0.25      0.38        72
non-covid-19       0.73      0.96      0.83       151

   micro avg       0.73      0.73      0.73       223
   macro avg       0.74      0.61      0.60       223
weighted avg       0.74      0.73      0.68       223
```

H- Comparer ces modèles en termes de performance en utilisant les métriques de classification habituelles.

Je montre déjà le rapport de classification pour chaque modèle. et le meilleur est KNN avec 50% pour la classe covid-19 et 84% pour la classe non-covid-19.

I- Utilisez une approche Ensemble Learning pour combiner les décisions de tous les modèles.

In [74]:

```
from collections import Counter

def most_frequent(List):
    occurence_count = Counter(List)
    return occurence_count.most_common(1)[0][0]

def all_models_in_one(X_test):
    y_all_models_in_one_test = []
    for example in enumerate(X_test):
        all_mdels_predects = []
        all_mdels_predects.append(classifier_knn.predict([example[1]])[0])
        all_mdels_predects.append(classifier_svm.predict([example[1]])[0])
        all_mdels_predects.append(classifier_dt.predict([example[1]])[0])
        all_mdels_predects.append(classifier_rfc.predict([example[1]])[0])
        all_mdels_predects.append(classifier_lr.predict([example[1]])[0])
        y_all_models_in_one_test.append(most_frequent(all_mdels_predects))
    return y_all_models_in_one_test
```

In [75]:

```
y_pred = all_models_in_one(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[ 25  47]
 [  9 142]]
```

In [76]:

```
print(classification_report(y_test, y_pred, target_names=("covid-19", "non-cov
id-19")))
```

```
              precision    recall  f1-score   support

    covid-19       0.74      0.35      0.47        72
non-covid-19       0.75      0.94      0.84       151

   micro avg       0.75      0.75      0.75       223
   macro avg       0.74      0.64      0.65       223
weighted avg       0.75      0.75      0.72       223
```

Released By El Houcine ES SANHAJI
Data Science Student

Thank You.